

## Chapter 2

# Perceptual Feature Extraction

**Abstract** In this chapter,<sup>†</sup> we present a deep model-based and data-driven hybrid architecture (DMD) for feature extraction. First, we construct a deep learning pipeline for progressively learning image features from simple to complex. We mix this deep model-based pipeline with a data-driven pipeline, which extracts features from a large collection of unlabeled images. Sparse regularization is then performed on features extracted from both pipelines in an unsupervised way to obtain representative patches. Upon obtaining these patches, a supervised learning algorithm is employed to conduct object prediction. We present how DMD works and explain why it works more effectively than traditional models from both aspects of neuroscience and computational learning theory.

**Keywords** Data driven · Deep learning · DMD · Feature extraction · Model-based

### 2.1 Introduction

Extracting useful features from a scene is an essential step of any computer vision and multimedia analysis tasks. Though progress has been made in past decades, it is still quite difficult for computers to accurately recognize an object or analyze the semantics of an image. In this chapter, we study two extreme approaches of feature extraction, *model-based* and *data-driven*, and then evaluate a hybrid scheme.

One may consider model-based and data-driven to be two mutually exclusive approaches. In practice; however, they are not. Virtually all *model* construction relies on some information from data; all data-driven schemes are built upon some *models*,

---

<sup>†</sup> © ACM, 2010. This chapter is a minor revision of the author's work with Zhiyu Wang and Dingyin Xia [1] published in VLS-MCMR'10. Permission to publish this chapter is granted under copyright license #2587600190581.

simple or complex. The key research questions for the feature-extraction task of an object recognition or image annotation application are:

1. *Can more data help a model?* Given a *model*, can the availability of more training data improve feature quality, and hence improve annotation accuracy?
2. *Can an improve model help data-driven?* Give some fixed amount of data, can a model be enhanced to improve feature quality, and hence improve annotation accuracy?

We first closely examine a model-based deep-learning scheme, which is neuroscience-motivated. Strongly motivated by the fact that the human visual system can effortlessly conduct these tasks, neuroscientists have been developing vision models based on physiological evidences. Though such research may still be in its infancy and several hypotheses remain to be validated, some widely accepted theories have been established. This chapter first presents such a model-based approach. Built upon the pioneer neuroscience work of Hubel [2], all recent models are founded on the theory that visual information is transmitted from the primary visual cortex (V1) over extrastriate visual areas (V2 and V4) to the inferotemporal cortex (IT). IT in turn is a major source of input to the prefrontal cortex (PFC), which involves in linking perception to memory and action [3]. The pathway from V1 to IT, which is called the *visual frontend* [4], consists of a number of simple and complex layers. The lower layers attain simple features that are invariant to scale, position and orientation at the pixel level. Higher layers detect complex features at the object-part level. Pattern reading at the lower layers are unsupervised; whereas recognition at the higher layers involves supervised learning. Computational models proposed by Lee [5] and Serre [6] show such a multi-layer generative approach to be effective in object recognition.

Our empirical study compared features extracted by a neuro-science-motivated deep-learning model and those extracted by a data-driven scheme through an application of image annotation. For the data-driven scheme, we employed features of some widely used pixel aggregates such as shapes and color/texture patches. These features construct a feature space. Given a previously unseen data instance, its annotation is determined through some nearest-neighbor scheme such as  $k$ -NN or kernel methods. The assumption of the data-driven approach is that if the features of two data instances are close (similar) in the feature space, then their target semantics would be same. For a data-driven scheme to work well, its feature space must be densely populated with training instances so that unseen instances can find sufficient number of nearest neighbors as their references.

We made two observations from the results of our experimental study. First, when the number of training instances is small, the model-based deep-learning scheme outperforms the data-driven. Second, while both feature sets commit prediction errors, each does better on certain objects. Model-based tends to do well on objects of a regular, rigid shape with similar interior patterns; whereas the data-driven model performs better in recognizing objects of variant perceptual characteristics. These observations establish three guidelines for feature design.

1. *Recognizing objects with similar details.* For objects that have regular features, *invariance* should be the top priority of feature extraction. A feature-extraction pipeline that can be invariant to scale, position and orientation requires only a handful of training instances to obtain a good set of features for recognizing this class of objects. For this class of objects, model-based works very well.
2. *Recognizing objects with different details.* Objects with variant features, such as strawberries in different orientations and environmental settings, or dalmatians with their varying patterns, do not have invariant features to extract. For recognizing such an object class, *diversity* is the design priority of feature extraction. To achieve diversity, a learning algorithm requires a large number of training instances to collect abundant samples. Therefore, data-driven works better for this class of objects.
3. *Recognizing objects within abstracts.* Classifying objects of different semantics such as whales and lions being mammals, or tea cups and beer mugs being cups, is remote to percepts. Abstract concept classification requires a WordNet-like semantic model.

The first two design principles confirm that feature extraction must consider both feature invariance and feature diversity; but how? A feedforward pathway model designed by Poggio's group [7] holds promises in obtaining invariant features. However, additional signals must be collected to enhance the diversity aspect. As Serre [5] indicates, feedback signals are transmitted back to V1 to pay attention to details. Biological evidences suggest that a feedback loop in visual system instructs cells to "see" local details such as color-based shapes and shape-based textures. These insights lead to the design of our hybrid model data-driven hybrid architecture (DMD), which combines a deep model-based pipeline with a data-driven pipeline to form a six-layer hierarchy. While the model-based pipeline faithfully models a deep learning architecture based on visual cortex's feedforward path [8], the data-driven pipeline extracts augmented features in a heuristic-based fashion. The two pipelines join at an unsupervised middle layer, which clusters low-level features into feature patches. This unsupervised layer is a critical step to effectively regularize the feature space [9, 10] for improving subsequent supervised learning, making object prediction both effective and scalable. Finally, at the supervised layer, DMD employs a traditional learning algorithm to map patches to semantics. Empirical studies show that DMD works markedly better than traditional models in image annotation. DMD's success is due to (1) its simple to complex deep pipeline for balancing invariance and selectivity, and (2) its model-based and data-driven hybrid approach for fusing feature specificity and diversity.

In this chapter, we show that a model-based pipeline encounters limitations. As we have explained, a data-driven pipeline is necessary for recognizing objects of different shapes and details. DMD employs both approaches of *deep* and *hybrid* to achieve improved performance for the following reasons:

1. *Balancing feature invariance and selectivity.* DMD implements Serre's method [8] to achieve a good balance between feature invariance and selectivity. To achieve feature selectivity, DMD conducts multi-band, multi-scale, and

multi-orientation convolutions. To achieve invariance, DMD only keeps signals of sufficient strengths via pooling operations.

2. *Properly using unsupervised learning to regularize supervised learning.* At the second, the third, and the fifth layers, DMD introduces unsupervised learning to reduce features so as to prevent the subsequent supervised layer from learning trivial solutions.
3. *Augmenting feature specificity with diversity.* Through empirical study, we identified that a model-based only approach cannot effectively recognize irregular objects or objects with diversified patterns; and therefore, fuse into DMD a data-driven pipeline. We point out subtle pitfalls in combining model-based and data-driven and propose a remedy for noise reduction.

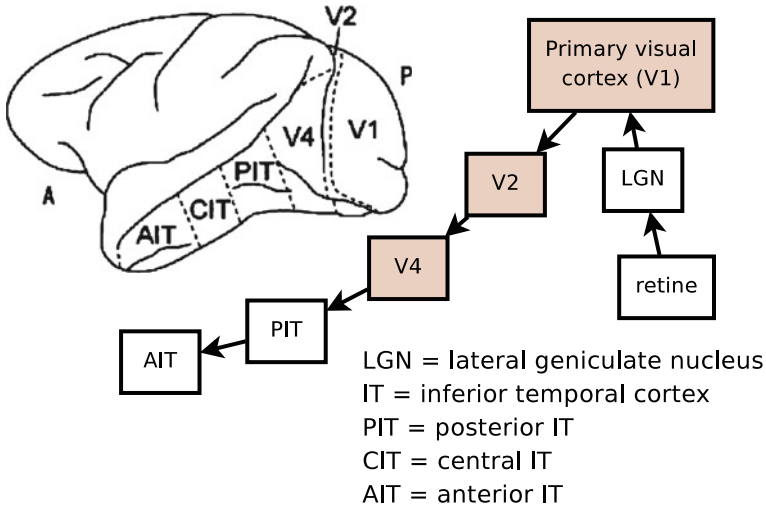
## 2.2 DMD Algorithm

DMD consists two pipelines of six steps. Given a set of training images, the *model-based* pipeline feeds training images to the edge detection step. At the same time, the *data-driven* pipeline feeds training images directly to the step of sparsity regularization. We first discuss the *model-based pipeline* of DMD in [Sect. 2.2.1](#), and then its *data-driven* pipeline in [Sect. 2.2.2](#).

### 2.2.1 Model-Based Pipeline

Figure 2.1 depicts that visual information is transmitted from the primary visual cortex (V1) over extrastriate visual areas (V2 and V4) to the IT. Physiological evidences indicate that the cells in V1 largely conduct *selection* operations, and cells in V2 and V4 *pooling* operations. Based on such, M. Riesenhuber and T. Poggio's theory of feedforward path of object recognition in the cortex [7] establishes a qualitative way to model the ventral stream in the visual cortex. Their model suggests that the visual system consists of multiple layers of computational units where *simple S* units alternate with *complex C* units. The *S* units deal with signal selectivity, whereas the *C* units deal with invariance. Lower layers attain features that are invariant to scale, position, and orientation at the pixel level. Higher layers detect features at the object-part level. Pattern reading at the lower layers are largely unsupervised, whereas recognition at the higher layers involves supervised models. Recent advancements in *deep learning* [10] have led to mutual justifications that a model-based, hierarchical model enjoys these computational advantages:

- Deep architectures enjoy advantages over shallow architectures (please consult [11] for details), and
- Unsupervised initiated deep architectures can enjoy better generalization performance [12].



**Fig. 2.1** Information flow in the visual cortex. (See the brain structure in [13])

Motivated by both physiological evidences [8, 14] and computational learning theories [5], we designed DMD’s model-based pipeline with six steps:

1. *Edge selection* (Sect. 2.2.1.1). This step corresponds to the operation conducted by cells in V1 and V2 [15], which detect edge signals at the pixel level.
2. *Edge pooling* (Sect. 2.2.1.2). This step also corresponds to cells in V1 and V2. The primary operation is to pool strong, representative edge signals.
3. *Sparsity regularization* (Sect. 2.2.1.3). To prevent too large a number of features, which can lead to dimensionality curse, or too low a level, which may lead to trivial solutions, DMD uses this unsupervised step to group edges into patches.
4. *Part selection* (Sect. 2.2.1.4). There is not yet strong physiological evidence, but it is widely believed that V2 performs part selection and then feeds signals directly to V4. DMD models this step to look for image patches matching those prototypes (patches) produced in the previous step.
5. *Part pooling* (Sect. 2.2.1.5). Cells in V4 [16], which have larger receptive fields than V1, deal with parts. Because of their larger receptive fields, V4’s selectivity is preserved over translation.
6. *Supervised learning* (Sect. 2.2.1.6). Learning occurs at all steps and certainly at the level of IT cortex and PFC. This top-most layer employs a supervised learning algorithm to map a patch-activation vector to some objects.

### 2.2.1.1 Edge Selection

In this step, computational units model the classical simple cells described by Hubel and Wiesel in the primary visual cortex (V1) [17]. A simple selective operation

is performed by V1 cells. To model this operation, Serre [8] uses Gabor filters to perform a 2D convolution, Lee [5] suggests using a convolutional restricted Boltzmann machine (RBM), and Ranzato [18] constructs an encoder convolution. We initially employed T. Serre’s strategy since T. Serre [6, 8] justifies model selection and parameter tuning based on strong physiological evidences, whereas computer scientists often justify their models through contrived experiments on a small set of samples. The input image is transmitted into a gray-value image, where only the *edge* information is of interest. The 2D convolution is a summation-like operation, whose convolution kernel is to model the receptive fields of cortical simple cells [19]. Different sizes of Gabor filters are applied as the convolution kernel to process the gray-value image  $\mathbf{I}$ , using this format:

$$\mathbf{F}_s(x, y) = \exp\left(-\frac{x_0^2 + \gamma^2 y_0^2}{2\sigma^2}\right) \times \cos\left(\frac{2\pi}{\lambda} x_0\right), \quad (2.1)$$

where

$$x_0 = x \cos \theta + y \sin \theta \text{ and } y_0 = -x \sin \theta + y \cos \theta.$$

In (2.1),  $\gamma$  is the aspect ratio and  $\theta$  is the orientation, which takes values  $0^\circ$ ,  $45^\circ$ ,  $90^\circ$ , and  $135^\circ$ . Parameters  $\sigma$  and  $\lambda$  are the effective width and wave length, respectively. The Gabor filter forms a 2D matrix with the value at position  $(x, y)$  to be  $\mathbf{F}_s(x, y)$ . The matrix size ( $s \times s$ ) or the Gabor filter size ranges from  $7 \times 7$  to  $37 \times 37$  pixels in intervals of two pixels. Thus there are 64 (16 scales  $\times$  4 orientations) different receptive field types in total. With different parameters, Gabor filters can cover different orientations and scales and hence increase selectivity. The output of the edge-selection step is produced by 2D convolutions (conv2) of the input image and  $n_b \times n_s \times n_f = 64$  Gabor filters of

$$\mathbf{I}_{S\_edge(i_b, i_s, i_f)} = \text{conv2}(\mathbf{I}, \mathbf{F}_{i_f}), \quad (2.2)$$

where

$$i_f = (i_b \times n_s + i_s) \times n_f + i_f.$$

### 2.2.1.2 Edge Pooling

In the previous step, several edge-detection output matrices are produced, which sufficiently support selectivity. At the same time, there is clearly some redundant or noisy information produced from these matrices. Physiological evidences on cats show that a MAX-like operation is taken in complex cells [15] to deal with redundancy and noise. To model this MAX-like operation, Serre’s, Lee’s, and Ranzato’s work all agree on applying a MAX operation on outputs from the simple cells. The response  $\mathbf{I}_{edge(i_b, i_f)}$  of a complex unit corresponds to the response of the strongest

**Fig. 2.2** DMD Model-based pipeline, steps 1 and 2Input: image data  $\mathbf{I}$ Output: edge feature  $\{\mathbf{I}_{edge(i_b, i_f)} | i_b, i_f\}$ **Initialization:**

- 1: Calculate normalize parameters  $\mathbf{I}_{Norm}$
- 2:  $n_b \leftarrow 8$  // total # of bands
- 3:  $n_s \leftarrow 2$  // total # of scales
- 4:  $n_f \leftarrow 4$  // total # of orientations

**Edge Selection:**

- 1: **for**  $i_b = 1$  to  $n_b$  **do**
- 2:     **for**  $i_s = 1$  to  $n_s$  **do**
- 3:         **for**  $i_f = 1$  to  $n_f$  **do**
- 4:              $\mathbf{I}_{S\_edge(i_b, i_s, i_f)} \leftarrow \text{conv2}(\mathbf{I}, \text{Gabor}(i_b, i_s, i_f))$
- 5:             **end for**
- 6:         **end for**
- 7:     **end for**
- 8: Normalize  $\mathbf{I}_{S\_edge}$  by  $\mathbf{I}_{Norm}$

**Edge Pooling:**

- 1: **for**  $i_b = 1$  to  $n_b$  **do**
- 2:     **for**  $i_f = 1$  to  $n_f$  **do**
- 3:         **for each**  $x, y$  in dimensions of  $\mathbf{I}_{edge(i_b, i_f)}$  **do**
- 4:              $\mathbf{I}_{edge(i_b, i_f)}(x, y) \leftarrow \max_{i_s} \mathbf{I}_{S\_edge(i_b, i_s, i_f)}(x, y)$
- 5:             **end for**
- 6:         **end for**
- 7:     **end for**

of all the neighboring units from the previous edge-selection layer. The output of this edge-pooling layer is as follows:

$$\mathbf{I}_{edge(i_b, i_f)}(x, y) = \max_{i_s \in \mathbf{V}_s, m \in \mathcal{N}(x, y)} \mathbf{I}_{S\_edge(i_b, i_s, i_f)}(x_m, y_m),$$

where  $(x_m, y_m)$  stands for edge-selection results at position  $(x, y)$ . The max is taken over the two scales within the same spatial neighborhood of the same orientation, justified by the experiments conducted by the work of Serre [20].

### 2.2.1.3 Sparsity Regularization

A subtle and important step of a deep architecture is to perform proper initialization between layers. The edge-pooling step may produce a huge number of edges. With such a large-sized output, the next layer may risk learning trivial solutions at the pixel level. Both Serre [8] and Ekanadham [21] suggest to sparsify the output of V2 (or input to V4).

To perform the sparsification, we form pixel *patches* via sampling. In this way, not only the size of the input to the part-selection step is reduced, but patches larger

than pixels can regularize the learning at the upper layers. The regularization effect is achieved by the fact that parts are formed by neighboring edges, not edges at random positions. Thus, there is no reason to conduct learning directly on the edges. A patch is a region of pixels sampled at a random position of a training image at four orientations. An object can be fully expressed if enough representative patches have been sampled. It is important to note that this sampling step can be performed incrementally when new training images are available. The result of this unsupervised learning step is  $n_p$  prototype patches, where  $n_p$  can be set initially to be a large value, and then trimmed back by the part-selection step.

In Sect. 2.2.2 we show that the data-driven pipeline also produces patches by sampling a large number of training instances. Two pipelines join at this unsupervised regularization step.

### 2.2.1.4 Part Selection

So far, DMD has generated patches via clustering and sampling. This part-selection step finds out which patches may be useful and of what patches an object part is composed. Part selection units describe a larger region of objects than the edge detection, by focusing on parts of the objects. Similar to our approach, Serre’s  $S_2$  units behave as radial basis function (RBF) units, Lee uses a convolutional deep belief network (CDBN), and Ranzato’s algorithm implements a convolutional operation for the decoder. All are consistent with well-known response properties of neurons in the primate inferotemporal cortex (IT).

Serre proposes using Gaussian-like Euclidean distance to measure similarity between an image and the pre-calculated prototypes (patches). Basically, we would like to find out what patches an object consists of. Analogically, we are constructing a map from object-parts to an object using the training images. Once the mapping has been learned, we can then classify an unseen image.

To perform part selection, we have to examine if patches obtained in the regularization step appear frequently enough in the training images. If a patch appears frequently, that patch can be selected as a part; otherwise, that patch is discarded for efficiency. For each training image, we match its edge patches with the  $n_p$  prototyped patches generated in the previous step. For the  $i_b^{th}$  band of an image’s edge detection output, we obtain for the  $i_p^{th}$  patch a measure as follows:

$$\mathbf{I}_{S\_part(i_b, i_p)} = \exp(-\beta \|\mathbf{X}_{i_b} - \mathbf{P}_{i_p}\|^2), \quad (2.3)$$

where  $\beta$  is the sharpness of the tuning and  $\mathbf{P}_{i_p}$  is one of the  $n_p$  patches learned during sparsity regularization.  $\mathbf{X}_{i_b}$  is a transformation of the  $\mathbf{I}_{edge(i_b, i_f)}$  with all  $n_f$  orientations merged to fit the size of  $\mathbf{P}_{i_p}$ . We obtain  $n_b$  measurements of the image for each prototype patch. Hence the total number of measurements that this part-selection step makes is the number of patches times the number of bands, or  $n_p \times n_b$ .



**Fig. 2.3** DMD steps 4 and 5

Input: edge features  $\{\mathbf{I}_{edge(i_b, i_f)} | i_b, i_f\}$ , patches  $\{\mathbf{P}_{i_P} | i_P\}$   
 Output: part features  $\mathbf{v}_{part}$

**Initialization:**

- 1:  $n_P \leftarrow \text{size of } \{\mathbf{P}_{i_P} | i_P\}$  // total # of patches
- 2:  $n_b \leftarrow 8$  // total # of bands, the same as before

**Part Selection:**

- 1: **for**  $i_P = 1$  to  $n_P$  **do**
- 2:   merge  $\mathbf{I}_{edge(i_b, i_f)}$  for all  $i_f$  to get  $\mathbf{I}_{edge(i_b)}$
- 3:    $\mathbf{I}_{S\_part(i_b, i_P)} \leftarrow \text{distance}(\mathbf{I}_{edge(i_b)}, \mathbf{P}_{i_P})$
- 4: **end for**

**Part Pooling:**

- 1: **for**  $i_P = 1$  to  $n_P$  **do**
- 2:    $\mathbf{v}_{part(i_P)} \leftarrow \min_{i_b} \mathbf{I}_{S\_part(i_b, i_P)}$
- 3: **end for**

**2.2.1.5 Part Pooling**

Each image is measured against  $n_P$  patches, and for each patch,  $n_b$  measurements are performed. To aggregate  $n_b$  measurements into one, we resort to the part-pooling units. The part-pooling units 1 correspond to visual cortical V4 neurons. It has been discovered that a substantial fraction of the neurons takes the maximum input as output in visual cortical V4 neurons of rhesus monkeys (macaca mulatta) [16], or

$$\mathbf{v}_{part(i_P)} = \min_{i_b} \mathbf{I}_{S\_part(i_b, i_P)}. \quad (2.4)$$

The MAX operation (maximizing similarity is equivalent to minimizing distance) can not only maintain feature invariance, but also scale down feature-vector size. The output of this stage for each training image is a vector of  $n_P$  values as depicted by the pseudo code in Fig. 2.3.

**2.2.1.6 Supervised Learning**

At the top layer, DMD performs part-to-object mapping. At this layer, any traditional *shallow* learning algorithm can work reasonably well. We employ SVMs to perform the task. The input to SVMs is a set of vector representations of image patches produced by this model-based pipeline and by the data-driven pipeline, which we present next. Each image is represented by a vector of real values each depicting the image's perceptual strength matched by a prototype patch.

## 2.2.2 Data-Driven Pipeline

The key advantage of the model-based pipeline is feature invariance. For objects that have a rigid body of predictable patterns, such as a watch or a phone, the model-based pipeline can obtain invariant features from a small number of training instances. Indeed, our experimental results presented in Sect. 2.3 show that it takes just five training images to effectively learn the features of a watch and to recognize it. Unfortunately, for objects that can have various appearances such as pizzas with different toppings, the model-based pipeline runs into limitations. The features it learned from the toppings of one pizza cannot help recognize a pizza with different toppings. The key reason for this is that invariance may cause overfitting, and that hurts selectivity.

To remedy the problem, DMD adds a data-driven pipeline. The principal idea is to collect enough examples of an object so that feature selectivity can be improved. By collecting signals from a large number of training data, it is also likely to collect signals of different scales and orientations. In other words, instead of relying solely on a model-based pipeline to deal with invariance, we can collect enough examples to ensure with high probability that the collected examples can cover most transformations of features.

Another duty that the data-driven pipeline can fulfill is to augment a key shortcoming of the model-based pipeline, i.e., it considers only the feedforward pathway of the visual system. It is well understood that some complex recognition tasks may require recursive predictions and verifications. Backprojection models [22, 23] and attention models [24] are still in early stage of development, and hence there is no solid basis of incorporating feedback. DMD uses heuristic-based signal processing subroutines to extract patches for the data-driven pipeline. The extracted patches are merged with those learned in the sparse-regularization step of the model-based pipeline.

We extracted patches in multiple resolutions to improve invariance [25, 26]. We characterized images by two main features: color and texture. We consider shapes as attributes of these main features.<sup>1</sup>

### 2.2.2.1 Color Patches

Although the wavelength of visible light ranges from 400 to 700 nm, research effort [28] shows that the colors that can be named by all cultures are generally limited to eleven. In addition to *black* and *white*, the discernible colors are *red*, *yellow*, *green*, *blue*, *brown*, *purple*, *pink*, *orange* and *gray*.

---

<sup>1</sup> *Disclaimer:* We do not claim these heuristic-based features to be novel. Other heuristic-based features [27] may also be useful. What we consider to be important is that these features can augment model-based features to improve diversity before a principled theory can be formulated by neuroscientists to model cortex feedback/feedforward recursive signals.

We first divided color into 12 color bins including 11 bins for culture colors and one bin for outliers [26]. At the coarsest resolution, we characterized color using a color mask of 12 bits. To recorded color information at finer resolutions, we record eight additional features for each color. These eight features are color histograms, color means (in H, S and V channels), color variances (in H, S and V channel), and two shape characteristics: elongation and spread. Color elongation characterizes the shape of a color and spreadness characterizes how that color scatters within the image [29]. We categorize color features by coarse, medium and fine resolutions.

### 2.2.2.2 Texture Patches

Texture is an important cue for image analysis. Studies [30–33] have shown that characterizing texture features in terms of structuredness, orientation, and scale (coarseness) fits well with models of human perception. A wide variety of texture analysis methods have been proposed in the past. We chose a discrete wavelet transformation (DWT) using quadrature mirror filters [31] because of its computational efficiency.

Each wavelet decomposition on a 2D image yields four subimages: a  $\frac{1}{2} \times \frac{1}{2}$  scaled-down image of the input image and its wavelets in three orientations: horizontal, vertical and diagonal. Decomposing the scaled-down image further, we obtain a tree-structured or wavelet packet decomposition. The wavelet image decomposition provides a representation that is easy to interpret. Every subimage contains information of a specific scale and orientation and also retains spatial information. We obtain nine texture combinations from subimages of three scales and three orientations. Since each subimage retains the spatial information of texture, we also compute elongation and spreadness for each texture channel.

### 2.2.2.3 Feature Fusion

Now, given an image, we can extract the above color and texture information to produce some clusters of features. These clusters are similar to those patches generated by the model-based pipeline. All features, generated by the model-based or data-driven pipeline are inputs to the sparsity regularization step, depicted in Sect. 2.2.1.3, to conduct subsequent processing. In Sect. 2.3.3 we discuss where fusion can be counter-productive and propose a remedy to reduce noise.

## 2.3 Experiments

Our experiments were designed to answer three questions:

1. How does a model-based approach compare to a data-driven approach? Which model performs better? Where and why?

**Table 2.1** Model-based versus data-driven

Training Size	Model-based (%)	Data-driven (%)
15	22.53	16.32
30	28.06	20.61

2. How does DMD perform compared to an individual approach, model-based or data-driven?
3. How much does the unsupervised regularization step help?

To answer these questions we conducted three experiments:

1. Model-based versus data-driven model.
2. DMD versus individual approaches.
3. Parameter tuning at the regularization step.

### 2.3.1 Dataset and Setup

To ensure that our experiments can cover object appearances of different characteristics (objects of similar details, different details, and within abstracts, as depicted in Sect. 2.1), we collected training and testing data from ImageNet [34]. We selected 10,885 images of 100 categories to cover the above characteristics. We followed the two pipelines of DMD to extract model-based features and data-driven features. For each image category, we cross-validated by using 15 or 30 images for training, and the remainder for testing to compute annotation accuracy. Because of the small training-data size, using linear SVMs turned out to be competitive to using advanced kernels. We thus employed linear SVMs to conduct all experiments.

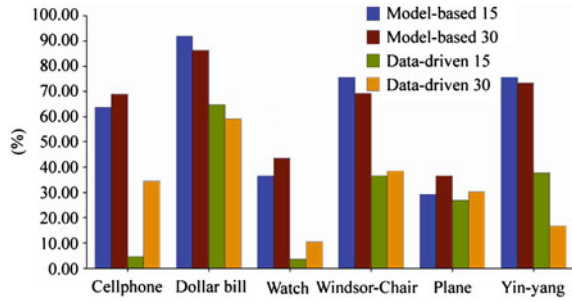
### 2.3.2 Model-Based versus Data-Driven

This experiment was designed to evaluate individual models and explain where and why model-based or data-driven is more effective.

#### 2.3.2.1 Overall Accuracy

Table 2.1 summarizes the average annotation accuracy of the model-based-only versus data-driven-only method with 15 and 30 training images, respectively. The table shows that the model-based method to be more effective. (We will shortly explain this result to be not-so-meaningful.) We next looked into individual categories to examine the reasons why.

**Fig. 2.4** Model-based outperforms data-driven (see color insert)



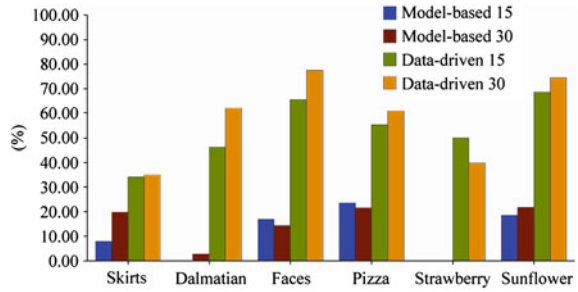
**Table 2.2** Images with a rigid body (see color insert)

Category	Images		
Cellphone			
Watch			
Yin-yang			
Plane			
Windsor chair			

**2.3.2.2 Where Model-Based Works Better**

Figure 2.4 shows a set of six categories (example images shown in Table 2.2) where the model-based method performs much better than the data-driven in class-prediction accuracy (on the y-axis).

**Fig. 2.5** Data-driven outperforms model-based (see color insert)



First, on some categories such as ‘dollar bills’, ‘Windsor-chair’ and ‘yin-yang’, increasing training data from 15 to 30 does not improve annotation accuracy. This is because these objects exhibit precise features (e.g., all dollar bills are the same), and as long as a model-based pipeline can deal with scale/position/orientation invariance, the feature-learning process requires only a small number of examples to capture their idiosyncrasies. The data-driven approach on these six categories performs quite poorly because its lacking the ability to deal with feature invariance. For instance, the data-driven pipeline cannot recognize watches of different sizes and colors, or Windsor-chairs of different orientations.

### 2.3.2.3 Where Data-Driven Works Better

Data-driven works better than model-based when objects exhibit patterns that are similar but not necessarily identical nor scale/position/orientation invariant. Data-driven can work effectively when an object exhibit some consistent characteristics such as apples are red or green, and dalmatians have black patches of irregular shapes.













Figure 2.5 shows a set of six categories where data-driven works substantially better than model-based in class-prediction accuracy.

Table 2.3 display example images from four categories, ‘strawberry’, ‘sunflower’, ‘dalmatian’, and ‘pizza’. Both ‘strawberry’ and ‘sunflower’ exhibit strong color characteristics, whereas ‘dalmatian’ and ‘pizza’ exhibit predictable texture features. For these categories, once after sufficient amount of samples can be collected, semantic-prediction can be performed with good accuracy. The model-based pipeline performs poorly in this case because it does not consider color nor can it find invariant patterns (e.g., pizzas have different shapes and toppings).

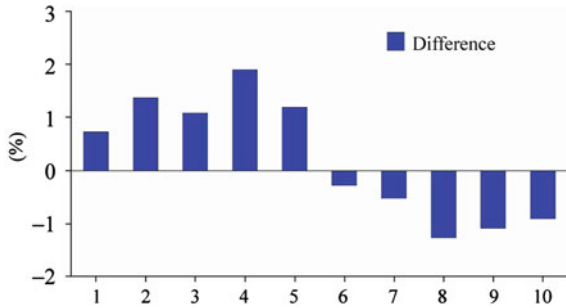
### 2.3.2.4 Accuracy versus Training Size

We varied the number of training instances for each category from one to ten to further investigate the strengths of model-based and data-driven. The  $x$ -axis of Fig. 2.6 indicates the number of training instances, and the  $y$ -axis the accuracy

**Table 2.3** Images with a rigid body (see color insert)













Class	Category	Images		
<i>Visual words (color)</i>	Strawberry			
	Sunflower			
<i>Visual words (texture)</i>	Dalmatian			
	Pizza			

**Fig. 2.6** Accuracy comparison (x-axis for the training numbers and y-axis for the accuracy model-based minus data-driven)



of model-base subtracted by the accuracy of data-driven. A positive margin means that the model-based outperforms the data-driven in class prediction, whereas a negative means that the data-driven outperforms. When the size of training data is below five, the model-based outperforms data-driven. As we have observed from the previous results, model-based can do well with a small number of training instances on objects of invariant features. On the contrary, a data-driven approach cannot be productive when the number of training instances is scarce, as its name suggests. Also as we have explained, even the features of an object class can be quite predictable, varying camera and environmental parameters can produce images of different scales, orientations, and colors. A data-driven pipeline is not expected to do well unless it can get ample samples for capturing these variations.

**Table 2.4** Images with a rigid body (see color insert)

Class	Category	Images		
<i>Rigid body</i>	Watch			
	Windsor chair			
<i>Texture</i>	Dalmatian			
	Pizza			

### 2.3.2.5 Discussion

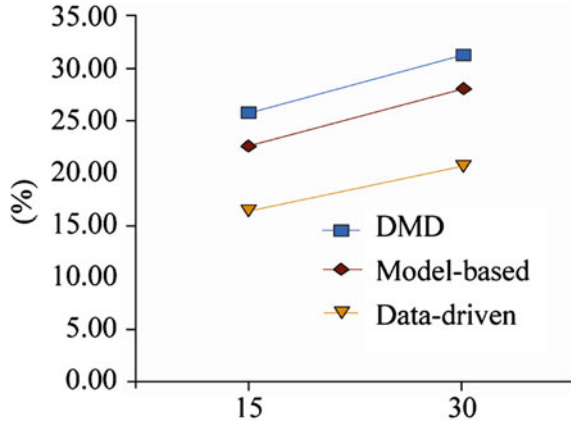
Table 2.4 displays patches of selected categories to illustrate the strengths of the model-based and data-driven, respectively. The patches of ‘watch’ and ‘Windsor-chair’ show patterns of almost identical edges. Model-based thus works well with these objects.

The ‘dalmatian’ and ‘pizza’ images tell a different story. The patches of these objects are similar but not identical. Besides, these patterns do not have strong edges. The color and texture patterns extracted by the data-driven pipeline can be more productive for recognizing these objects when sufficient samples are available.

*Note* Let us revisit the result presented in Table 2.1. It is easy to make the data-driven look better by adding more image categories favoring data-driven to the testbed. Or the bias of a dataset can favor one approach over the other. Thus, evaluating which model works better cannot be done by looking only at the average accuracy. On a dataset of a few hundred or thousand categories, evaluation should be performed on individual categories.



**Fig. 2.7** DMD versus individual models



**Table 2.5** Fusion accuracy with large training pool

Category	Model-based (%)		Data-driven (%)		DMD (%)	
	30	Large <sup>2</sup>	30	Large <sup>2</sup>	30	Large <sup>2</sup>
Plane	36.60	<b>97.17</b>	30.41	90.57	29.51	96.23
Watch	33.97	<b>60.87</b>	8.61	43.48	24.40	55.07
Skirts	19.82	53.66	35.14	58.54	29.73	<b>65.85</b>
Sunflower	18.18	25.00	60.00	65.00	63.64	<b>70.00</b>

### 2.3.3 DMD versus Individual Models

Fusing model-based and data-driven seems like a logical approach. Figure 2.7 shows that DMD outperforms individual models in classification accuracy on training sizes of both 15 and 30. Table 2.5 reports four selected categories on which we also performed training on up to 700 training instances.

#### 2.3.3.1 Where Fusion Helps

For objects that the data-driven model works better, adding more features from the model-based pipeline can be helpful. The data-driven model needs a large number of examples to characterize an object (typically deformable as we have discussed), and those features produced by the model-based pipeline can help. On ‘skirts’ and ‘sunflower’, DMD outperforms data-driven when 100 images were used for training ‘skirts’, and 65 for ‘sunflower’. (These are the maximum amount of training data that can be obtained from ImageNet for these categories.) On these deformable objects, we have explained why the data-driven approach is more productive. Table 2.5 shows that the model-based features employed by DMD can help further improve class-prediction accuracy.

### 2.3.3.2 Where Fusion May Hurt

For objects that the model-based can achieve better prediction accuracy, adding more features from the data-driven pipeline can be counter-productive. For all object categories in Fig. 2.4 with 30 training instances, adding data-driven features reduces classification accuracy. This is because for those objects where the model-based performs well, additional features may contribute noise. For instance, the different colors of watches can make watch recognition harder when color is considered as a feature. Table 2.5 shows that when 30 training instances were used, DMD’s accuracy on ‘watch’ was degraded by 25 percentile. However, the good news is that when ample samples were provided to capture almost all possible colors of watches, the prediction accuracy improved. When we used 170 watch images to train the ‘watch’ predictor, the accuracy of DMD trails data-driven by just 5 category where we were able to get a hold of 700 training instances, both DMD and model-base enjoy the same degree of accuracy.

### 2.3.3.3 Strength-Dominant Fusion

One key lesson learned from this experiment is that the MAX operator in the pooling steps of DMD is indeed effective in telling features from noise. Therefore, DMD can consider amplifying the stronger signals from either the model-based or the data-driven pipeline. When the signals extracted from an image well match some patches generated by the model-based pipeline, the model-based classifier should dominate the final class-prediction decision. This simple adjustment to DMD can improve its prediction accuracy on objects of rigid bodies, and hence the average prediction accuracy. Another key lesson learned (though obvious) is that the amount of training instances must be large to make the data-driven pipeline effective. When the training size is small and model-based can be effective, the data-driven pipeline should be turned off.

### 2.3.4 Regularization Tuning

This experiment examined the effect of the realization step. Recall that regularization employs unsupervised schemes to achieve both feature selection and feature reduction. Table 2.6 shows the effect of the number of prototype patches  $n_p$  on the final prediction accuracy of DMD. When  $n_p$  is set between  $2,000 \times 4$  and  $2,500 \times 4$ , the prediction accuracy is the best. A small  $n_p$  may cause underfitting and too large an  $n_p$  may cause overfitting. Parameter  $n_p$  decides selectivity, and its best setting must be tuned through an empirical process like this.

**Table 2.6** Accuracy of different number of patches

$n_p/4$	Training size (%)		$n_p/4$	Training size (%)	
	15	30		15	30
200	21.01	26.09	1500	23.62	29.04
300	22.08	27.82	2000	<b>23.75</b>	29.21
500	22.53	28.54	2500	23.30	<b>29.61</b>
1000	23.00	28.82	3000	23.27	29.42

### 2.3.5 Tough Categories


















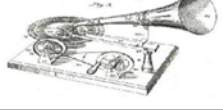
Table 2.7 presents six categories where DMD cannot perform effectively with a small amount of training data. For instance, the best prediction accuracy with 30 training images on ‘barrel’ and ‘cup’ is 12 and 17%, respectively. These objects exhibit such diversified perceptual features, and neither model-based nor data-driven with a small set of training data can capture their characteristics. Furthermore, the ‘cup’ category is more than percepts, as liquid containers of different shapes and colors can be classified as a cup. To improve class-prediction accuracy on these challenging categories, more training data ought to be collected to make DMD effective.

## 2.4 Related Reading

The CBIR community has been striving to bridge the *semantic gap* [35] between low-level features and high-level semantics for over a decade. (For a comprehensive survey, please consult [27].) Despite the progress has been made on both feature extraction and computational learning, these algorithms are far from being completely successful. On feature extraction, scale-invariant feature transform (SIFT) was considered a big step forward in the last decade for extracting scale-invariant features. SIFT may have improved feature invariance, but it does not effectively deal with feature selectivity. Indeed, SIFT has shown to be effective in detecting near-replicas of images [36], but it alone has not been widely successful in more general recognition tasks. As for computational learning, discriminative models such as linear SVMs [37] and generative models such as latent dirichlet allocation (LDA) [38] have been employed to map features to semantics. However, applying these and similar models directly to low-level features are considered to be *shallow learning*, which may be too limited for modeling complex vision problems. Yoshua Bengio [9] argues in theory that some functions simply cannot be represented by architectures that are too shallow.

If shallow learning suffers from limitations, then why have’t deep learning been widely adopted? Indeed, neuroscientists have studied how the human vision system works for over 40 years [2]. Ample evidences [39–41] indicate that the human visual system is a pipeline of multiple layers: from sensing pixels and detecting edges to forming patches, recognizing parts, and then composing parts into objects.

**Table 2.7** Images with a rigid body (see color insert)

Category	Images		
Beaver			
Barrel			
Cup			
Mayfly			
Wild cat			
Gramophone			

Physiological evidences strongly suggest that *deep* learning, rather than *shallow*, is appropriate. Unfortunately, before the work of Hinton in 2006 [10] deep models were not fully embraced partly because of their high intensity of computation and partly because of the well known problem of local optima. Recent advancements in neuroscience [3] motivated computer scientists to revisit deep learning in two respects:

1. *Unsupervised learning in lower layers.* The first couple of layers of the visual system are unsupervised, whereas supervised learning is conducted at the latter layers.
2. *Invariance and selectivity tradeoff.* Layers in the visual system deal with invariance and selectivity alternately.

These insights have led to recent progress in deep learning. First, Salakhutdinov et al. [42] show that using an unsupervised learning algorithm to conduct pre-training at each layer, a deep architecture can achieve much better results. Second, Serre [8] shows that by alternating pooling and sampling operations between layers, a balance

between feature invariance and selectivity can be achieved. The subsequent work of Lee [5] confirms these insights.

## 2.5 Concluding Remarks

In this chapter, we first conducted empirical study to compare the model-based and the data-driven approach to image annotation. From our experimental results, we learned insights to design DMD, a hybrid architecture of deep model-based and data-driven learning. We showed the usefulness of unsupervised learning at three steps: edge-pooling, sparse regularization, and part-pooling. Unsupervised learning plays a pivotal role in making good tradeoffs between invariance and selectivity, and between specificity and diversity. We also showed that the data-driven pipeline can always be helped by the model-based. However, the other way may introduce noise when the amount of training data is scarce. DMD makes proper adjustments in making model-based and data-driven complement each other to achieve good performance.

Besides perceptual features that can be directly extracted from images, researchers have also considered camera parameters, textual information surrounding images, and social metadata, which can be characterized as contextual features. Nevertheless, perceptual feature extraction remains to be a core research topic of computer vision and image processing. Contextual features can complement image content but cannot substitute perceptual features. How to combine context and content belongs to the topic of multimodal fusion, which we address in [Chaps. 7, 8 and 9](#).

## References

1. Z. Wang, D. Xia, E. Y. Chang, A deep model-based and data-driven hybrid architecture for image annotation, in *Proceedings of ACM International Workshop on Very-Large-Scale Multimedia Corpus, Mining and Retrieval*, pp. 13–18, 2010
2. D.H. Hubel, T.N. Wiesel, Receptive fields and functional architecture of monkey striate cortex. *J. Physiol.* **195**(1), 215–243 (1968)
3. E. Miller, The prefrontal cortex and cognitive control. *Nat. Rev. Neurosci.* **1**(1), 59–66 (2000)
4. G. Potamianos, C. Neti, J. Luetten, I. Matthews, Audio–visual automatic speech recognition: An overview. in *Issues in Visual and Audio–Visual Speech Processing* (MIT Press, Cambridge, 2004)
5. H. Lee, R. Grosse, R. Ranganath, A. Ng, Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations, in *Proceedings of International Conference on Machine Learning (ICML)*, 2009
6. T. Serre, Learning a dictionary of shape-components in visual cortex: comparison with neurons, humans and machines. Ph.D. Thesis, Massachusetts Institute of Technology, 2006
7. M. Riesenhuber, T. Poggio, Are cortical models really bound by the binding problem. *Neuron* **24**(1), 87–93 (1999)
8. T. Serre, L. Wolf, S. Bileschi, M. Riesenhuber, T. Poggio, Robust object recognition with cortex-like mechanisms. *IEEE Trans. Pattern Anal. Mach. Intell.* **29**(3), 411–426 (2007)

9. Y. Bengio, *Learning Deep Architectures for AI* (Now Publishers, 2009)
10. G. Hinton, S. Osindero, Y. Teh, A fast learning algorithm for deep belief nets. *Neural Comput.* **18**(7), 1527–1554 (2006)
11. Y. Bengio, Y. LeCun, Scaling learning algorithms towards AI, in *Large-Scale Kernel Machines* (MIT Press, Cambridge, 2007), pp. 321–360
12. G. Loosli, S. Canu, L. Bottou, Training invariant support vector machines using selective sampling, in *Large Scale Kernel Machines* (MIT Press, Cambridge, 2007) pp. 301–320
13. M. Yasuda, T. Banno, H. Komatsu, Color selectivity of neurons in the posterior inferior temporal cortex of the macaque monkey. *Cereb. Cortex* **20**(7), 1630–1646 (2009)
14. K. Tsunoda, Y. Yamane, M. Nishizaki, M. Tanifuji, Complex objects are represented in macaque inferotemporal cortex by the combination of feature columns. *Nat. Neurosci.* **4**, 832–838 (2001)
15. I. Lampl, D. Ferster, T. Poggio, M. Riesenhuber, Intracellular measurements of spatial integration and the MAX operation in complex cells of the cat primary visual cortex. *J. Neurophysiol.* **92**(5), 2704 (2004)
16. T. Gawne, J. Martin, Responses of primate visual cortical V4 neurons to simultaneously presented stimuli. *J. Neurophysiol.* **88**(3), 1128 (2002)
17. D. Hubel, T. Wiesel, Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *J. Physiol.* **160**(1), 106 (1962)
18. M. Ranzato, F. Huang, Y. Boureau, Y. LeCun, Unsupervised learning of invariant feature hierarchies with applications to object recognition, in *Proceedings of IEEE CVPR*, 2007
19. J. Jones, L. Palmer, An evaluation of the two-dimensional Gabor filter model of simple receptive fields in cat striate cortex. *J. Neurophysiol.* **58**(6), 1233 (1987)
20. T. Serre, M. Riesenhuber, Realistic modeling of simple and complex cell tuning in the hmax model, and implications for invariant object recognition in cortex. MIT technical report, 2004
21. C. Ekanadham, S. Reader, H. Lee, Sparse deep belief net models for visual area V2, in *Proceedings of NIPS*, 2008
22. B. Olshausen, C. Anderson, D. Van Essen, A neurobiological model of visual attention and invariant pattern recognition based on dynamic routing of information. *J. Neurosci.* **13**(11), 4700 (1993)
23. D. Walther, T. Serre, T. Poggio, C. Koch, Modeling feature sharing between object detection and top-down attention. *J. Vis.* **5**(8), 1041 (2005)
24. S. Chikkerur, T. Serre, T. Poggio, A Bayesian inference theory of attention: neuroscience and algorithms. MIT technical report MIT-CSAIL-TR-2009-047, 2009
25. E. Chang, B. Li, C. Li, Toward perception-based image retrieval. *IEEE Content-Based Access of Image and Video Libraries*, pp. 101–105, 2000
26. S. Tong, E. Y. Chang, Support vector machine active learning for image retrieval, in *Proceedings of ACM International Conference on Multimedia* (ACM, New York, 2001) pp. 107–118
27. R. Datta, D. Joshi, J. Li, J. Wang, Image retrieval: ideas, influences, and trends of the new age. *ACM Comput. Surv. (CSUR)* **40**(2), 1–60 (2008)
28. S. Coren, L.M. Ward, J.T. Enns, *Sensation and Perception*, 6th edn. (Wiley, New York, 2003)
29. J. Leu, Computing a shape's moments from its boundary. *Pattern Recognit.* **24**(10), 949–957 (1991)
30. H. Tamura, S. Mori, T. Yamawaki, Textural features corresponding to visual perception. *IEEE Trans. Syst. Man Cybern.* **8**(6), 460–473 (1978)
31. J. Smith, S. Chang, Automated image retrieval using color and texture. *IEEE Trans. Pattern Anal. Mach. Intell.* 1996
32. P. Wu, B. Manjunath, S. Newsam, H. Shin, A texture descriptor for browsing and similarity retrieval. *Sig. Process. Image Commun.* **16**(1-2), 33–43 (2000)
33. W. Ma, H. Zhang, Benchmarking of image features for content-based retrieval, in *Proceedings of Asilomar Conference on Signals Systems and Computers*, pp. 253–260, 1998
34. J. Deng, W. Dong, R. Socher, L. Li, K. Li, F.F. Li, Imagenet: a large-scale hierarchical image database, in *Proceedings of IEEE CVPR*, pp 156–161, 2009

35. A. Smeulders, M. Worring, S. Santini, A. Gupta, R. Jain, Content-based image retrieval at the end of the early years. *IEEE Trans. Pattern. Anal. Mach. Intell.* **22**(12), 1349–1380 (2000)
36. Y. Ke, R. Sukthankar, PCA-SIFT: a more distinctive representation for local image descriptors, in *Proceedings of IEEE CVPR*, pp. 506–513, 2004
37. O. Chapelle, P. Haffner, V. Vapnik, SVMs for histogram-based image classification. *IEEE Trans. Neural Netw.* **10**(5), 1055 (1999)
38. D. Blei, A. Ng, M. Jordan, Latent dirichlet allocation. *J. Mach. Learn. Res.* **3**, 993–1022 (2003)
39. T.L.W. Serre, T. Poggio, Object recognition with features inspired by visual cortex, in *Proceedings of IEEE CVPR*, 2005
40. C. Gross, Visual functions of inferotemporal cortex. *Handbook of Sensory Physiology*, vol 7(3), 1973
41. C. Gross, C. Rocha-Miranda, D. Bender, Visual properties of neurons in inferotemporal cortex of the macaque. *J. Neurophysiol.* **35**(1), 96–111 (1972)
42. R. Salakhutdinov, A. Mnih, G. Hinton, Restricted Boltzmann machines for collaborative filtering, in *Proceedings of International Conference on Machine Learning (ICML)*, pp. 791–798, 2007