Hierarchical Part Matching for Fine-Grained Visual Categorization

Lingxi Xie[1] Qi Tian[2] Richang Hong[3] Shuicheng Yan[4] Bo Zhang[5]
[1,5]State Key Laboratory of Intelligent Technology and Systems (LITS)
Tsinghua National Laboratory for Information Science and Technology (TNList)
Department of Computer Science and Technology, Tsinghua University, Beijing, China
[2]Department of Computer Science, University of Texas at San Antonio, Texas, USA
[3]School of Computer and Information, Hefei University of Technology, Hefei, China
[4]Department of Electrical and Computer Engineering, National University of Singapore, Singapore
[1]198808xc@gmail.com [2]qitian@cs.utsa.edu
[3]hongrc@hfut.edu.cn [4]eleyans@nus.edu.sg [5]dcszb@mail.tsinghua.edu.cn

Abstract

As a special topic in computer vision, fine-grained visual categorization (FGVC) has been attracting growing attention these years. Different with traditional image classification tasks in which objects have large inter-class variation, the visual concepts in the fine-grained datasets, such as hundreds of bird species, often have very similar semantics. Due to the large inter-class similarity, it is very difficult to classify the objects without locating really discriminative features, therefore it becomes more important for the algorithm to make full use of the part information in order to train a robust model.

In this paper, we propose a powerful flowchart named Hierarchical Part Matching (HPM) to cope with fine-grained classification tasks. We extend the Bag-of-Features (BoF) model by introducing several novel modules to integrate into image representation, including foreground inference and segmentation, Hierarchical Structure Learning (HSL), and Geometric Phrase Pooling (GPP). We verify in experiments that our algorithm achieves the state-of-the-art classification accuracy in the Caltech-UCSD-Birds-200-2011 dataset by making full use of the ground-truth part annotations.

1. Introduction

Classifying images according to their semantic meaning is a basic task in the computer vision community. It is a basic way towards image understanding and implies a wide range of commercial applications. Among them, fine-grained visual categorization (FGVC) is a special case, in which the visual concepts in different categories are very similar. Sometimes, it is even very difficult for a human to distinguish, say, hundreds of species of birds.

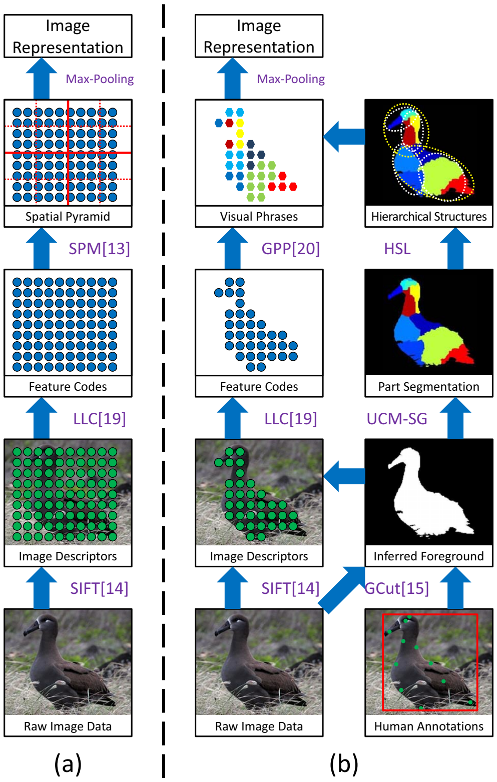


Figure 1. Comparison of the proposed HPM model (b) and the Bag-of-Features (BoF) framework (a) (best viewed in color PDF).

As one of the most popular algorithms for image classification, the Bag-of-Features (BoF) model [7] has been widely used in many image applications. In essential, the BoF model represents each image as a long vector, and adopts a machine learning algorithm to train a classifier. Based

1550-5499/13 $31.00 © 2013 IEEE
DOI 10.1109/ICCV.2013.206

on manufactural local features, it suffers from the well-known semantic gap between low-level features and high-level concepts. Although in recent years researchers have proposed new approaches to deal with the above problem [3] [4] [11] and verified that these modules help to boost the classification performance [20], the connection between image representation and visual concepts is still weak. It is also observed [18] that traditional classification model works poorly on the fine-grained tasks, due to the limited use of really discriminative features located on special parts of the objects.

In this paper, we propose a novel flowchart named Hierarchical Part Matching (HPM) to cope with fine-grained classification problems. We make full use of the ground-truth part annotation to help us obtain better image alignment and segmentation, and provide a much more descriptive image representation by building mid-level structures on local features as well as segmented regions. The new modules added in the HPM model (see Figure 1) could be summarized as follows. First, we use the ground-truth annotation to infer the object (foreground) on the image and segment it into semantic parts. Second, we propose the Hierarchical Structure Learning (HSL) algorithm to find mid-level concepts beyond basic parts. Third, we use the Geometric Phrase Pooling (GPP) algorithm to capture mid-level structures in the local feature groups. Integrating all the modules above gives a powerful model, which achieves the state-of-the-art classification performance in a challenging fine-grained image collection. The main contribution of this paper is to provide an intuitive, simple and efficient way of using ground-truth part annotations, and emphasize the importance of part detection in the fine-grained classification tasks with surprising boost in classification accuracy.

The rest of this paper is organized as follows. Section 2 gives a survey of the related works. Section 3 introduces the fine-grained dataset and the baseline algorithm used in experiments. Next, we introduce the Hierarchical Part Matching (HPM) model by individually presenting three modules: foreground inference and segmentation in Section 4, the Hierarchical Structure Learning (HSL) algorithm in Section 5, and the Geometric Phrase Pooling (GPP) algorithm in Section 6. After experimental results are shown in Section 7, we draw the conclusions and summarize the future works in Section 8.

2. Related Works

2.1. The Bag-of-Features Model

The Bag-of-Features (BoF) model [7] is one of the most popular algorithms for image classification. The flowchart of the BoF model is illustrated in Figure 1(a). Starting from raw image data, we first extract SIFT [14] descriptors as local features. and train a visual vocabulary or codebook using K-Means clustering. Locality-sensitive Linear Coding (LLC) [19] is then used to quantize the local descriptors onto a sparse histogram in the feature space. We use max-pooling for a statistical summarization, and Spatial Pyramid Matching (SPM) [13] for a naive spatial context modeling. Finally, the representation vectors are fed into a linear SVM for training and testing.

2.2. Fine-Grained Visual Categorization

Fine-grained visual categorization (FGVC) is an emerging research area in computer vision, in which a dataset typically contains hundreds of categories sharing similar semantics. For example, the Caltech-UCSD Birds-200-2011 dataset [18] contains 200 bird species, and there are 120 different kinds of dogs in the Stanford Dogs dataset [12]. To cope with the fine-grained classification tasks, researchers have proposed many novel algorithms, such as visual attributes [8], random templates [22], hierarchical matching [6] and part-based one-vs-one features [2].

2.3. Foreground Inference and Segmentation

In the fine-grained image classification tasks, almost all the categories share similar background clutters, and objects often vary from each other only in some small regions named "parts". Therefore it is very important to distinguish the objects from background and segment them into semantic parts. For foreground inference, a popular tool is the Grab-Cut [15] algorithm, which iteratively infers the foreground region from an initial mask. For part-based segmentation, we can use the Ultrametric Contour Map (UCM) [1] as an unsupervised algorithm for to calculate the closed boundaries. There are also works proposing multi-label segmentation to combine above steps as one [5].

2.4. The Geometric Phrase Pooling Algorithm

The basic units in the BoF model are visual words, which are too far away from visual concepts. As a mid-level structure bridging low-level features and high-level semantics, visual phrases are verified useful in various image applications [23] [25]. The Geometric Phrase Pooling (GPP) algorithm [20] is an efficient phrase extraction and pooling approach, in which we define visual phrases as local groups of visual words, and perform an efficient pooling algorithm to enhance the similarity in both geometric and feature spaces.

3. Dataset and Baseline System

3.1. The Dataset

In our experiments, we use the Caltech-UCSD Birds-200-2011 (CUB-200-2011) dataset [18], which contains 200 bird species and 11788 images in total. Also, a manually labeled bounding box and at most 15 landmark points

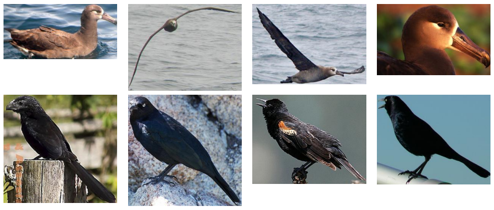


Figure 2. Samples from the Caltech-UCSD Birds-200-2011 dataset [18]. Upper: images from the same category (001. Black-footed Albatross) showing large intra-class variation. Lower: images of different species showing small inter-class variation.

are provided for each image. It is a very challenging dataset. For the sample images shown in Figure 2, it is even difficult for humans to recognize them accurately. The main reason for choosing this dataset is to use the ground-truth annotations which are very important to our approach.

3.2. The Baseline System

We use the Bag-of-Features (BoF) model as our baseline system. Here, we build a mathematical notation system for this model.

In the Birds dataset, a ground-truth **bounding box** is provided for each image. We use the image region within bounding box, and resize it into the same size, *i.e.*, width and height do not exceed 300. In this way we obtain the **raw image** denoted as **I**:

$$\mathbf{I} = (a_{ij})_{W \times H} \tag{1}$$

where a_{ij} is the **pixel** at position (i, j), and it is a 3-dimensional vector for RGB-images. Also, at most L **landmark points** are annotated for each image ($L = 15$ in the Birds dataset), Denote the points as $\{p_l\}$, where $l = 1, 2, \ldots, L$.

We extract SIFT descriptors [14] on the raw image and obtain a set of **local descriptors**:

$$\mathcal{D} = \{(\mathbf{d}_1, \mathcal{R}_1), (\mathbf{d}_2, \mathcal{R}_2), \ldots, (\mathbf{d}_M, \mathcal{R}_M)\} \tag{2}$$

where $\mathbf{d}_m$ and $\mathcal{R}_m$ denote the **description vector** and **occupied region** of the m-th descriptor, respectively. M is the total number of descriptors, which could be hundreds or even thousands under dense sampling. The description vector $\mathbf{d}_m$ is a D-dimensional vector, where $D = 3 \times 128 = 384$ using OpponentSIFT (OppSIFT) [16] on RGB-images.

After descriptors have been extracted, they are quantized to be compact. For this purpose, we train a **codebook C** using descriptors from the whole dataset. **C** is a $B \times D$ matrix consisting of B vectors with dimension D, each of which is called a **codeword**. The number of codewords, or the **codebook size** B, is 2048 in our experiments.

Next, descriptors are represented using the codebook. This process is called **coding**, for we are encoding each descriptor into a sparse histogram on the feature space. Given a codebook with B codewords, the quantization vector or **feature vector** for a descriptor $\mathbf{d}_m$ would be a B-dimensional vector $\mathbf{w}_m$, which is named the corresponding **visual word** of **descriptor** $\mathbf{d}_m$. Denote $\mathcal{W}$ as the set of visual words:

$$\mathcal{W} = \{(\mathbf{w}_1, \mathcal{R}_1), (\mathbf{w}_2, \mathcal{R}_2), \ldots, (\mathbf{w}_M, \mathcal{R}_M)\} \tag{3}$$

Now, we aggregate the local visual words for global image representation. The **max-pooling** strategy calculates the maximal response on each codeword:

$$\mathbf{f} = \max_{1 \leqslant m \leqslant M} \mathbf{w}_m \tag{4}$$

where the notation $\max_m$ denotes the element-wise maximization. A 3-layer Spatial Pyramid Matching (SPM) [13] follows by dividing the image into hierarchical subregions for individual max-pooling and concatenating the pooled vectors as a **super-vector**. Finally, the super-vectors are fed into a linear SVM for classification.

4. Foreground Inference and Segmentation

4.1. Foreground Inference

A notable property of fine-grained datasets is the high similarity in backgrounds (non-object regions). Take the Birds dataset as an example. Water surfaces, trees and grasses appear in almost all the categories. As they introduce irrelevant features into the BoF model, it is reasonable to perform foreground inference and extract features only on the object (foreground regions).

We use the Grab-Cut algorithm [5] for foreground inference. The initial mask is constructed using ground-truth annotations, *i.e.*, the bounding box and landmark points. We set the pixels outside the bounding box as **definite background**, inside as **possible foreground** and the pixels around landmarks as **definite foreground**. We run the the Grub-Cut algorithm in no more than 10 iterations to produce the inference result. Figure 3 illustrates the inference process.

4.2. Energy Function and Segmentation

As the regular Spatial Pyramid often fails to align corresponding regions in the fine-grained tasks, we need a more accurate spatial segmentation to capture the semantic parts of the objects. The proposed segmentation algorithm starts with calculating the Ultrametric Contour Map (UCM) [1], which generates closed contours with decreasing boundary intensities to cut the image into smaller and smaller regions. Denote the intensity map as **U**:

$$\mathbf{U} = (u_{ij})_{W \times H} \tag{5}$$

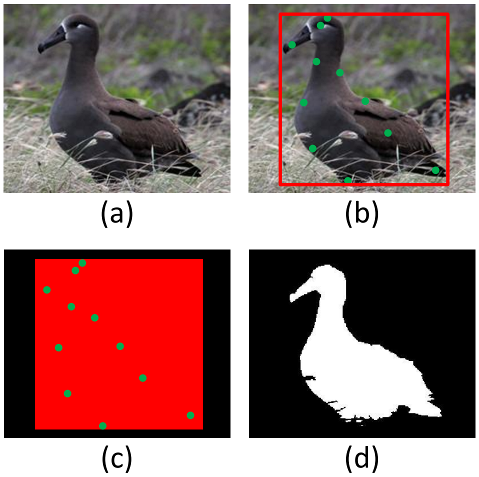
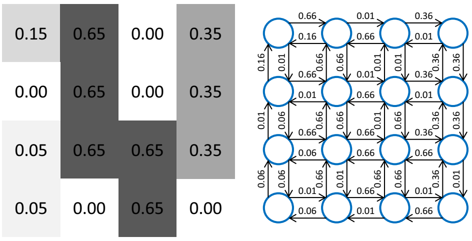


Figure 3. The foreground inference process (best viewed in color PDF). (a) the original image. (b) bounding box (red) and small areas around part locations (green). (c) the initial mask in Grab-Cut, in which black, red and green regions are definite BG, possible FG and definite FG, respectively. (d) the inferred foreground (white).



0.15	0.65	0.00	0.35
0.00	0.65	0.00	0.35
0.05	0.65	0.65	0.35
0.05	0.00	0.65	0.00

Figure 4. Construction of the graph $\mathcal{G}$. Left: a small patch on the UCM, where each grid is a pixel. The numbers on the grids are boundary intensities. Right: the constructed subgraph with edge weights shown on the arcs. The step penalty λ is 0.01 here.

where u_{ij} is the **boundary intensity** at position (i, j).

Based on the UCM, we construct a **directed graph** $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathcal{W}\}$, where $\mathcal{V} = \{v_{ij}\}$ consists of all the pixels, and $\mathcal{E}$ is composed of edges connecting adjacent pixels:

$$\mathcal{E} = \{(v_{ij} \rightarrow v_{i'j'}) \mid |i - i'| + |j - j'| = 1\} \tag{6}$$

The weight of an edge is determined by the boundary intensity at the tail node (pixel):

$$w(v_{ij} \rightarrow v_{i'j'}) = u_{i'j'} + \lambda \tag{7}$$

Here, λ is called the **step penalty**, which takes the geometric distance into consideration. Figure 4 shows a sample graph $\mathcal{G}$ constructed on a small patch.

After the graph is complete, we take the landmark points as source nodes, and calculate their **shortest paths** to other nodes (pixels). The Dijkstra algorithm gives a solution within $O(LN\log(N))$ time, where L is the number of points and $N = W \times H$ is the image size. Denote the distances as $\{d(p_l, v_{ij})\}$, where p_l is the l-th part location, and v_{ij} is an arbitrary node in graph $\mathcal{G}$. For later convenience, we define $\{d(0, v_{ij})\}$ as the **background distances**, which takes 0 if v_{ij} is a background node and $+\infty$ otherwise.

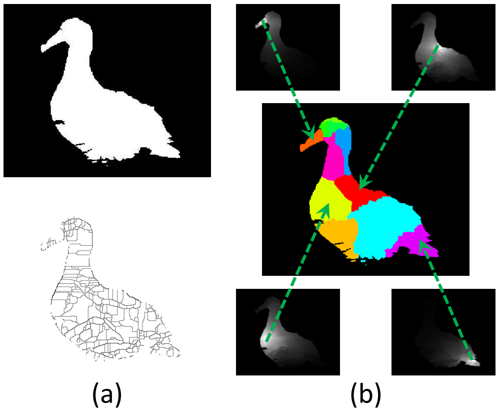


Figure 5. Segmentation illustration. (a) Inferred foreground and UCM (darker pixel, larger intensity). (b) Heatmap of distance from 'beak' (upper left), 'breast' (bottom left), 'back' (upper right) and 'tail' (bottom right), respectively. Pixel-wise minimization on all the distances yields the segmentation results (centered).

The segmentation process is to assign each node (pixel) to one of the landmarks. Denote an assignment as $\mathbf{S}$:

$$\mathbf{S} = (s_{ij})_{W \times H} \tag{8}$$

where s_{ij} is the index of assigned landmark of node v_{ij}, $0 \leqslant s_{ij} \leqslant L$, and $s_{ij} = 0$ implies assigning v_{ij} into background. To judge the quality of the segmentation, we define an **energy function** which is the summation of distance between each pixel and the assigned node:

$$f(\mathbf{S}) = \sum_{i,j} d(p_{s_{ij}}, v_{ij}) \tag{9}$$

(9) is easily solved using independent minimization:

$$s^{\star}_{ij} = \arg\min_{0 \leqslant l \leqslant L} d(p_l, a_{ij}) \tag{10}$$

The segmentation process is illustrated in Figure 5.

5. Hierarchical Structure Learning

Denote the set of all pixels as $\mathcal{I}$. Segmentation algorithm divides $\mathcal{I}$ into at most L foreground parts and one background region:

$$\mathcal{I} = \bigcup_{l=0}^{L} \mathcal{I}_l \tag{11}$$

where $\mathcal{I}_l$ represents the l-th segmented region for $l > 0$, and $\mathcal{I}_0$ is the background. All the segmented regions are exclusive, *i.e.*, $\mathcal{I}_{l_1} \cap \mathcal{I}_{l_2} = \varnothing$ for $l_1 \neq l_2$.

It is worth noting that the segmented regions are basic body parts, *i.e.*, 'nape', 'left eye', 'right leg', etc. Semantically, there exist mid-level concepts consisting of basic parts. For example, the concept 'eyes' consists of 'left eye' and 'right eye', and 'head' is composed of 'forehead', 'crown', 'beak', and 'eyes'. To combine the basic parts, close geometric locations and similar appearance features are the necessary conditions. Therefore, we need to quantize the geometric and feature **distances** for pairwise parts.

We use the set of descriptors $\{\mathbf{d}_m, \mathcal{R}_m\}$ as defined in Equation (2), the landmark points $\{p_l\}$ and the segmented regions $\{\mathcal{I}_l\}$ to calculate the distances. For an image in which $\mathcal{I}_{l_1}$ and $\mathcal{I}_{l_2}$ are both non-empty, the **geometric distance** between them is formulated as:

$$\mathrm{dist}_{\mathrm{g}}(\mathcal{I}_{l_1}, \mathcal{I}_{l_2}) = \left[\left(p_{l_1}^{\mathrm{X}} - p_{l_2}^{\mathrm{X}}\right)^2 + \left(p_{l_1}^{\mathrm{Y}} - p_{l_2}^{\mathrm{Y}}\right)^2\right]^{1/2} \quad (12)$$

and the **feature distance** is calculated as:

$$\mathrm{dist}_{\mathrm{f}}(\mathcal{I}_{l_1}, \mathcal{I}_{l_2}) = \left\| \operatorname*{avg}_{\mathcal{R}_m \cap \mathcal{I}_{l_1} \neq \varnothing} \mathbf{d}_m - \operatorname*{avg}_{\mathcal{R}_m \cap \mathcal{I}_{l_2} \neq \varnothing} \mathbf{d}_m \right\|_2 \quad (13)$$

Integrating (12) and (13) yields the **total distance**:

$$\mathrm{dist}(\mathcal{I}_{l_1}, \mathcal{I}_{l_2}) = \frac{\mathrm{dist}_{\mathrm{g}}(\mathcal{I}_{l_1}, \mathcal{I}_{l_2})}{\max \mathrm{dist}_{\mathrm{g}}(\mathcal{I})} + \frac{\mathrm{dist}_{\mathrm{f}}(\mathcal{I}_{l_1}, \mathcal{I}_{l_2})}{\max \mathrm{dist}_{\mathrm{f}}(\mathcal{I})} \quad (14)$$

where both distances are normalized. Finally, we average the total distance over all the images and obtain the **part distance** for l_1 and l_2.

$$\mathrm{dist}(l_1, l_2) = \operatorname*{avg}_{\mathcal{I}_{l_1}, \mathcal{I}_{l_2} \neq \varnothing} \mathrm{dist}(\mathcal{I}_{l_1}, \mathcal{I}_{l_2}) \quad (15)$$

We define a mid-level part $\mathcal{L}_{\mathrm{s}}$ as a set of basic parts:

$$\mathcal{L}_s = \{l_{s_1}, l_{s_2}, \ldots, l_{s_T}\} \quad (16)$$

where $2 \leqslant T \leqslant L$, and the cost for constructing $\mathcal{L}_{\mathrm{s}}$ as:

$$\mathrm{cost}(\mathcal{L}_s) = \frac{\sum_{1 \leqslant i < j \leqslant T} \mathrm{dist}\left(l_{s_i}, l_{s_j}\right)}{\frac{1}{2} T\left(T-1\right)} \quad (17)$$

Now, the **Hierarchical Structure Learning** (HSL) algorithm is very easy to implement.

1. **Initialization.** Start from the original part set $\mathcal{P} = \{1, 2, \ldots, L\}$ and a pre-defined learning parameter μ.

2. **Learning.** Enumerate all the subsets $\mathcal{L}_s \subseteq \mathcal{P}$, and calculate the cost function $c = \mathrm{cost}(\mathcal{L}_s)$. If $c \leqslant \mu$, then $\mathcal{L}_s$ is accepted as a mid-level part.

3. **Construction.** Organize all the original and learned parts as a hierarchical structure.

No.	μ	Learned Mid-Level Parts
#0	0.0	No mid-level parts are learned.
#1	0.1	**eyes** (left/right eye), **legs** (left/right leg), **wings** (left/right wing).
#2	0.3	**eyes**, **legs**, **wings**, **neck** (nape/throat).
#3	0.5	**eyes**, **legs**, **wings**, **neck**, **head** (beak/crown/forehead/eyes), **body** (back/belly/breast/tail).
#4	1.0	**eyes**, **legs**, **wings**, **neck**, **head**, **body**, (**wings/legs**), (**body/wings/legs**), **ALL**.

Table 1. The learning parameter μ and the learned mid-level parts. Bolded are the semantic name of the learned parts.

Denote $\widetilde{\mathcal{P}} = \left\{1, 2, \ldots, L, L+1, \ldots, \widetilde{L}\right\}$ as the set of all the original and learned parts, where $\widetilde{L} \geqslant L$.

Obviously, as the **learning parameter** μ increases, the learned structure will become more and more complex. We list some values of μ and the learned structures in Table 1. We can observe that the learned mid-level parts are semantically nameable (bolded in the table). Also, we can learn a hierarchical structure (more than 2 layers) when $\mu = 0.5$ and $\mu = 1.0$. Both the above observations reveal the effectiveness of our algorithm.

The learning process of the HSL algorithm requires to enumerate all the subsets $\mathcal{L}_s \subseteq \mathcal{P}$, resulting in a $O\left(L \times 2^L\right)$ time complexity, which grows exponentially with the number of basic parts, L. Fortunately, L is often very small, *e.g.*, no more than 20, in common objects. Our algorithm requires about 100 seconds in the case of $L = 20$, and less than 3s in the birds dataset ($L = 15$). Considering that the hierarchical structure is calculated only once, we can claim that our algorithm is very efficient.

6. Geometric Pooling Strategy

We return to the original image $\mathbf{I}$. After descriptor extraction and feature encoding, we obtain the set of descriptors $\mathcal{D}$ defined in (2) and the corresponding set of visual words $\mathcal{W}$ defined in (3), respectively. Also, we have at most $\widetilde{L}$ regions $\left\{\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_{\widetilde{L}}\right\}$ for each image.

The **Naive Pooling** (NP) strategy summarizes each region by finding all the related features:

$$\mathcal{W}_l = \{(\mathbf{w}_m, \mathcal{R}_m) \mid \mathcal{R}_m \cap \mathcal{I}_l \neq \varnothing\} \quad (18)$$

and performing max-pooling:

$$\mathbf{f}_l^{(\mathrm{W})} = \max_{(\mathbf{w}_m, \mathcal{R}_m) \in \mathcal{W}_l} \mathbf{w}_m \quad (19)$$

Despite its simplicity, the Naive Pooling strategy fails to capture the geometric information, which could be very useful for fine-grained recognition. Figure 6 shows such examples with dominant geometric features, such as 'crown'

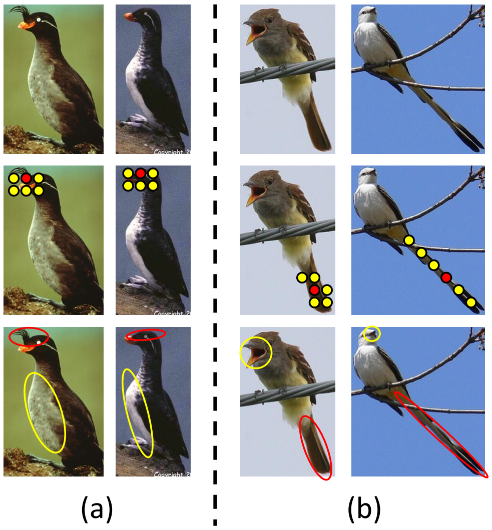


Figure 6. Examples illustrating Geometric Phrase Pooling (best viewed in color PDF). Upper: bird species varying from each other mainly in 'crown' shape (left pair) and 'tail' length (right pair). Middle: examples of Geometric Visual Phrases (GVP), in which red circles are central words and yellows are side words. The GVP in the last case is irregular, for the definition limits the side words on the same region (the long 'tail' here) as the central word. Bottom: the regions (of same color) with largest discriminativity increases. The differences in 'crown' and 'tail' are detected.

shape and 'tail' length. In this respect, we adopt the Geometric Phrase Pooling (GPP) algorithm [20], which defines visual phrases as neighboring word groups, and performs an efficient pooling algorithm to enhance the correlation between local word pairs.

In precise, GPP is performed within each segmented region $\mathcal{I}_l$ and the corresponding set $\mathcal{W}_l$. For each visual word $(\mathbf{w}_m, \mathcal{R}_m)$ in $\mathcal{W}_l$, we search for its K nearest neighbors in $\mathcal{W}_l$ and form a word group:

$$\mathcal{P}_{l,m} = \left\{\left(\mathbf{w}_{l,m,0}, \mathbf{l}_{l,m,0}\right), \ldots, \left(\mathbf{w}_{l,m,K}, \mathbf{l}_{l,m,K}\right)\right\} \quad (20)$$

$\mathcal{P}_{l,m}$ is the m-th **Geometric Visual Phrase** (GVP) in $\mathcal{W}_l$, in which $\mathbf{w}_{l,m,0} = \mathbf{w}_m$ is the **central word**, and others are **side words**. K is the **order** of $\mathcal{P}_{l,m}$, which is 20 as in [20]. Figure 6 illustrates some examples of visual phrases.

The **Geometric Phrase Pooling** (GPP) algorithm calculates the following representation vector on $\mathcal{P}_{l,m}$:

$$\mathbf{p}_{l,m} = \mathbf{w}_{l,m,0} + \max_{1 \leqslant k \leqslant K} \mathbf{w}_{l,m,k} \quad (21)$$

and summarizes all the phrases using max-pooling:

$$\mathbf{f}_l^{(\mathrm{P})} = \max_{(\mathbf{w}_m, \mathcal{R}_m) \in \mathcal{W}_l} \mathbf{p}_{l,m} \quad (22)$$

Here, we conduct an experiment to show the effectiveness of GPP. On the image pairs shown in Figure 6, we calculate the distance ϕ_l between the corresponding regions using visual words or phrases, respectively, and consider ϕ_l as the model's **discriminativity metric**:

$$\phi_l^{(\mathrm{W})} = \left\|\mathbf{f}_l^{(\mathrm{W})}(\mathbf{I}_1) - \mathbf{f}_l^{(\mathrm{W})}(\mathbf{I}_2)\right\|_2^2 \quad (23)$$

$$\phi_l^{(\mathrm{P})} = \left\|\mathbf{f}_l^{(\mathrm{P})}(\mathbf{I}_1) - \mathbf{f}_l^{(\mathrm{P})}(\mathbf{I}_2)\right\|_2^2 \quad (24)$$

We calculate the discriminativity increase $\phi_l^{(\mathrm{P})} - \phi_l^{(\mathrm{W})}$, and circle out the regions with largest values in Figure 6. The results reveal that GPP actually discovers useful geometric properties and combines them with the texture features.

7. Experiments

This section gives **classification results** on the Caltech-UCSD Birds-200-2011 dataset [18]. To make comparison, we keep the same settings as the baseline algorithms:

- **Local descriptors.** We use the VLFeat [17] library to extract OppSIFT descriptors [16]. The spatial stride and window size for dense sampling are 5 and 6, respectively.
- **Codebook learning.** We train a 2048-entry codebook with K-Means clustering. The number of descriptors collected for training is around 2 million.
- **Feature encoding.** We use LLC [19] for sparse coding with $K = 5$ as in the same literature.
- **Feature construction.** The feature vectors in the basic and mid-level regions are individually computed using max-pooling, and then concatenated as a super-vector. We normalize the super-vector using the separate normalization strategy proposed in [21].
- **Classification.** We use LibLINEAR [9], a scalable SVM implementation for training and testing.
- **Accuracy evaluation.** We select fixed numbers (5, 10, 20, 30) of images for training the classification model, and test it on the remaining images to calculate the average classification accuracy by category. A fixed training/testing split [18] is also used.

7.1. Annotation: Manual vs. Automatic

Many people have been debating on whether to use human annotations in fine-grained visual categorization (FGVC) tasks [18] [24] [6] [22]. Here we provide twofold clues by testing our model on automatically annotated parts and lighter manually annotated parts.

We use DPM [10], a part-based object detection model, for automatic part annotation. The templates of birds with 9 unnameable parts are trained on the PascalVOC 2007 and PascalVOC 2010 databases, respectively. Also we construct

Algorithm	Detail Explanation	Accuracy
Baseline	BoF model without parts	13.64%
VOC07	trained in VOC 2007 (9 parts)	9.43%
VOC11	trained in VOC 2011 (9 parts)	11.09%
3 parts	preserving 3 out of 15 parts	21.37%
6 parts	preserving 6 out of 15 parts	23.91%

Table 2. Comparison of **classification accuracies** (%) with 5 training samples per category. The automatic part-based model produces even worse results than the baseline, while it is possible to obtain better results with few manual annotations.

# training	5	10	20	30
Baseline	13.64	20.25	28.36	33.63
FG Inference	19.25	27.66	37.08	43.06
Part Seg.	**28.55**	**40.46**	**52.52**	**58.09**

Table 3. **Classification accuracies** (%) with foreground inference and segmentation. **FG Inference**: a 3-layer SPM on the foreground. **Part Segmentation**: spatial pooling on the segmented regions.

# training	5	10	20	30
Structure #0	28.55	40.46	52.52	58.09
Structure #1	29.29	41.62	53.36	59.24
Structure #2	29.75	42.03	53.55	59.32
Structure #3	**30.33**	**42.66**	**53.94**	**59.86**
Structure #4	27.38	38.64	50.22	56.11

Table 4. The mid-level parts help to improve the **classification accuracies** (%). See Table 1 for the structure details.

# training	5	10	20	30
No Phrase	30.33	42.66	53.94	59.86
GPP(5,5)	31.69	43.80	55.26	60.80
GPP(5,10)	32.23	45.10	56.11	61.93
GPP(5,20)	34.13	47.29	58.60	64.01
GPP(5,40)	**36.09**	**48.87**	**60.56**	**65.62**

Table 5. **Classification accuracies** (%) with and without Geometric Phrase Pooling. In parentheses are the numbers of coding bases for central and side words.

# training	5	10	20	30
Wah [18]	10.05	-	-	-
Wang [19]	13.64	20.25	28.36	33.63
Xie [20]	15.34	22.91	31.01	36.17
Ours (Mean)	**36.09**	**48.87**	**60.56**	**65.62**
Ours (StdDev)	±0.31	±0.60	±0.50	±0.46

Table 6. **Classification accuracies** (%) on the Birds dataset using random data split. LLC [19] is the baseline system.

Wah [18]	Zhang [24]	Wang [19]	Berg [2]	Ours
17.31	24.21	33.91	73.30	66.35

Table 7. **Classification accuracies** (%) on the Birds dataset using the fixed training/testing data split provided by [18]. In the fixed split, there are about 30 images per category for training the model.

two lighter sets of manual annotations by preserving 3 or 6 landmark points and discarding others. The classification results are listed in Table 2.

It is observed that the parts detected by the DPM model [10] provide limited help to fine-grained concept recognition. On the other hand, even partial manual annotations (3 or 6 parts) is valuable for visual categorization. Therefore we propose to use the ground-truth annotations temporarily in the fine-grained recognition, and improve the quality of object detection using the clues learned in the classification tasks.

7.2. Model and Parameters

First, we test the effectiveness of foreground inference and segmentation and list the results in Table 3. The classification accuracy is highly boosted when better alignment is provided on the objects.

Second, we test the Hierarchical Structure Learning (HSL) algorithm. We use the learned structures in Table 1 and list the corresponding results in Table 4. Using mid-level parts as extra bins, we improve the classification accuracy by a margin. Exceptions come from the last case (Structure #4) which is much too complex. Therefore, we preserve Structure #3 for the next experiments.

Finally, we test the accuracy gain from the Geometric Phrase Pooling algorithm. Results are listed in Table 5. We see that GPP indeed provides a better solution for spatial context modeling. We choose 5 and 40 as the numbers of coding bases for central and side words, respectively.

Both the SPM algorithm and our model (HPM) build hierarchical structures for feature pooling. We make full use of the ground-truth annotations, and extract semantic parts as spatial pooling bins. In this way, our algorithm significantly outperforms the baseline system in fine-grained image classification. HPM is not only a supporter of SPM which verifies that well-aligned parts and well-organized structures are useful for discrimination, and also a contradictor showing that global feature is not always better than local ones (See Structure #3 and #4 in Table 1 and 4. #4 uses global features and produces much lower accuracies than #3).

7.3. Comparison to Previous Works

Finally, we compare HPM with existing works **using ground truth part annotations**. We inherit the best parameters from the previous section, *i.e.*, Part Segmentation, Structure #3 in HSL, and numbers of coding bases (5,40) for GPP. Table 6 and 7 show the results on random and fixed data splits, respectively. HPM overwhelmingly outperforms the previous algorithms except the POOF proposed in [2]. The great improvement in classification accuracy comes from the full use of ground-truth part annotations.

8. Conclusions and Future Works

In this paper, we present a novel flowchart named Hierarchical Part Matching (HPM) for fine-grained visual categorization (FGVC). HPM contains three modules to enhance the BoF model. First, using the Grab-Cut algorithm and the Ultrametric Contour Map, we develop an effective algorithm for foreground inference and segmentation, generating more accurate object alignment. Second, we propose the Hierarchical Structure Learning (HSL) algorithm for finding mid-level concepts beyond basic parts. The learned parts are semantically nameable. Third, we use the Geometric Phrase Pooling (GPP) algorithm for spatial context modeling. Integrating all the modules makes HPM a powerful model, which shows notable improvements over the existing works on the Birds dataset.

Despite the leap in classification accuracies, our HPM model is still imperfect. For example, biological taxonomy provides a scientific classification system for all the bird species (available on Wikipedia). It implies a hierarchical classifier, on which we could apply various techniques such as transfer learning for fine-grained understanding. We will investigate these problems in our future works and look forward to a better model for fine-grained recognition.

9. Acknowledgements

This work was supported by the National Basic Research Program (973 Program) of China under Grant 2012CB316301, and Basic Research Foundation of Tsinghua National Laboratory for Information Science and Technology (TNList). This work was also supported in part to Dr. Qi Tian by ARO grant W911NF-12-1-0057, NSF IIS 1052851, Faculty Research Awards by Google, NEC Laboratories of America and FXPAL, UTSA START-R award and NSFC 61128007, respectively. This work was also supported by the Singapore National Research Foundation under its International Research Centre at Singapore Funding Initiative and administered by the IDM Programme Office.

References

[1] P. Arbeláez, M. Maire, C. Fowlkes, and J. Malik. From Contours to Regions: An Empirical Evaluation. *CVPR*, 2009.

[2] T. Berg and P. Belhumeur. POOF: Part-Based One-vs-One Features for Fine-Grained Categorization, Face Verification, and Attribute Estimation. *CVPR*, 2013.

[3] A. Bosch, A. Zisserman, and X. Muoz. Image Classification using Random Forests and Ferns. *ICCV*, 2007.

[4] Y. Boureau, F. Bach, Y. LeCun, J. Ponce, et al. Learning Mid-Level Features for Recognition. *CVPR*, 2010.

[5] Y. Boykov, O. Veksler, and R. Zabih. Fast Approximate Energy Minimization via Graph Cuts. *PAMI*, 2001.

[6] Q. Chen, Z. Song, Y. Hua, Z. Huang, and S. Yan. Hierarchical Matching with Side Information for Image Classification. *CVPR*, 2012.

[7] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray. Visual Categorization with Bags of Keypoints. *Workshop on Statistical Learning in Computer Vision, ECCV*, 2004.

[8] K. Duan, D. Parikh, D. Crandall, and K. Grauman. Discovering Localized Attributes for Fine-Grained Recognition. *CVPR*, 2012.

[9] R. Fan, K. Chang, C. Hsieh, X. Wang, and C. Lin. LIBLINEAR: A Library for Large Linear Classification. *JMLR*, 2008.

[10] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object Detection with Discriminatively Trained Part-Based Models. *PAMI*, 2010.

[11] J. Feng, B. Ni, Q. Tian, and S. Yan. Geometric Lp-Norm Feature Pooling for Image Classification. *CVPR*, 2011.

[12] A. Khosla, N. Jayadevaprakash, B. Yao, and F. Li. Novel Dataset for Fine-Grained Image Categorization. *First Workshop on FGVC, CVPR*, 2011.

[13] S. Lazebnik, C. Schmid, and J. Ponce. Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories. *CVPR*, 2006.

[14] D. G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *IJCV*, 2004.

[15] C. Rother, V. Kolmogorov, and A. Blake. GrabCut: Interactive Foreground Extraction Using Iterated Graph Cuts. *ACM Transactions on Graphics*, 2004.

[16] K. Van De Sande, T. Gevers, and C. Snoek. Evaluating Color Descriptors for Object and Scene Recognition. *PAMI*, 2010.

[17] A. Vedaldi and B. Fulkerson. VLFeat: An Open and Portable Library of Computer Vision Algorithms. *ACM Multimedia*, 2010.

[18] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. *Technical Report*, 2011.

[19] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong. Locality-Constrained Linear Coding for Image Classification. *CVPR*, 2010.

[20] L. Xie, Q. Tian, and B. Zhang. Spatial Pooling of Heterogeneous Features for Image Applications. *ACM Multimedia*, 2012.

[21] L. Xie, Q. Tian, and B. Zhang. Feature Normalization for Part-Based Image Classification. *ICIP*, 2013.

[22] S. Yang, L. Bo, J. Wang, and L. Shapiro. Unsupervised Template Learning for Fine-Grained Object Recognition. *NIPS*, 2012.

[23] J. Yuan, M. Yang, and Y. Wu. Mining Discriminative Cooccurrence Patterns for Visual Recognition. *CVPR*, 2011.

[24] N. Zhang, R. Farrell, and T. Darrell. Pose Pooling Kernels for Sub-Category Recognition. *CVPR*, 2012.

[25] S. Zhang, Q. Tian, G. Hua, Q. Huang, and S. Li. Descriptive Visual Words and Visual Phrases for Image Applications. *ACM Multimedia*, 2009.