Feature Engineering in Fine-Grained Image Classification

Shulin Yang

A dissertation submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2013

Reading Committee:

Linda G. Shapiro, Chair

Steve Seitz

Jue Wang

Program Authorized to Offer Degree:
Computer Science and Engineering

University of Washington

Abstract

Feature Engineering in Fine-Grained Image Classification

Shulin Yang

Chair of the Supervisory Committee:
Professor Linda G. Shapiro
Computer Science and Engineering

In most machine learning pipelines, feature engineering is an important module. The use of features has become the bottleneck for many learning related tasks, especially fo unstructured data like images and texts. This thesis researched on the use of features from several aspects, including feature design, feature selection, and feature learning, on a family of tasks called fine-grained image classification. Fine-grained classification refers to tasks in which the class differences are very subtle to observe, for example, to recognize sub-ordinate level classes like certain animal species, or to identify similar 3D shapes in medical images.

In particular, this thesis covers several different projects. The use of feature selection algorithm is first explored in analysing similar 3D shapes for a medical problem called craniosynostosis. Different sparse logistic regression models are investigated, and a new sparse logistic regression model called clustering lasso is proposed specifically for this problem. Next, on a specific fine-grained recognition problem – fast food recognition, an image representation called pairwise feature distribution (PFD) is proposed, which is focused on capturing the spatial information inside food images, using geometric pairwise features. The use of feature learning approaches is then explored on the general fine-grained object recognition problem, and a template model is proposed to improve the state-of-the-art object recognition framework by learning of mid-level feature representations for fine-grained tasks. The effectiveness of these algorithms proposed in this thesis is shown by comparison with the state-of-the-art algorithms on several publicly available benchmark datasets.

TABLE OF CONTENTS

LIST OF FIGURES

GLOSSARY

LOGISTIC REGRESSION: a type of regression analysis used for predicting the outcome of a categorical dependent variable (a dependent variable that can take on a limited number of categories) based on one or more predictor variables.

LASSO: a regression method that involves penalizing the absolute size of the regression coefficients.

FUSED LASSO: a generalization of lasso that is designed for problems with features that can be ordered in some meaningful way.

CLASSO: a generalization of lasso that is designed for problems with highly correlated features whose correlations are too complicated to specify.

PFD: pairwise feature distribution, an image representation that captures the geometric information of an image.

SVM: support vector machine, a family of supervised learning models with associated learning algorithms that analyze data and recognize patterns, used for classification and regression analysis.

FEATURE LEARNING: to automatically learn a good representation of the input from unlabeled data instead of hand-engineering feature representation.

FINE-GRAINED OBJECT RECOGNITION: a family of object recognition tasks in which class differences are subtle and hard to observe.

CRANIOSYNOSTOSIS: a condition in which one or more of the fibrous sutures in an infant skull prematurely fuses by turning into bone (ossification), thereby changing the growth pattern of the skull.

SIFT: scale-invariant feature transform, an algorithm in computer vision to detect and describe local features in images.

SPATIAL PYRAMID MATCHING: a method for recognizing scene categories based on approximate global geometric correspondence.

KERNEL DESCRIPTORS: a unified framework of feature descriptors that turn pixel level attributes into patch level features.

ACKNOWLEDGMENTS

The author wishes to express sincere appreciation to my advisor Linda G. Shapiro, my mentors Liefeng, Jue, Mei, Rahul, Ali, the members in the multimedia group Indri, Kasia, Jia, Sara, Irma, Ezgi, Joe, Jiunhung, Natalie, Bilge, and lots of other friends in the Computer Science department of University of Washington.

Chapter 1

INTRODUCTION

Nowadays, dealing with large amounts of data is a very hot topic. The world is continuously generating countless data, and how to process them effectively has become one of the biggest challenges of today. Images make up a large portion of this data, including natural images, medical images like CT and MRI, and a variety of sensor image data, like satellites images and infrared images. For example, in the category of natural images alone, there are 6 billion Flickr photos, 72 hours of videos uploaded to Youtube every minute, and over one billion smart phone users around the world who take photos every day of the food they eat, the place they go, and themselves.

However, how to effectively understand and use these data remains a major problem. Image classification serves as a basic way to organize the images. Specifically, typical image classification tasks include: object recognition, which classifies images based on the types of objects within them; scene recognition, which categorizes the scenery in the place where an image was taken; or classification of medical images based on the type and severity of the illness of the patients. Automatic systems that perform image classification tasks are very useful for understanding and processing these image data.

1.1 Fine-grained image classification

This thesis is focused on a special group of problems called fine-grained image classification. First, let us define what "fine-grained" means.

In object recognition, for example, traditionally there are two types of object recognition tasks: the basic-level object recognition and the instance-level object recognition. The first category, the basic-level problem, usually refers to distinguishing very different object categories, like a bird category and an airplane category, where different bird species are considered the same class. The second category, the instance-level problem, refers to dis-

Figure 1.1: Examples of the basic-level recognition tasks.

tinguishing images containing different instances which belong to the same object type, for example face recognition and fingerprint recogntion.

If we use "granularity" to represent the level of subtlety of the cross-class differences for an image classification task, a significant difference between these two families of problems is the granularity level. For basic-level problems, such as distinguishing between bird images and aircraft images as in Figure 1.1, their difference is very salient. Because the object types are distinct from each other, it is relatively easy to pinpoint the differences of the images containing bird and aircraft in multiple aspects, including color, shape and texture. Therefore, for this family of problems, identifying the distinguishing features is usually not difficult. Instead, a bigger challenge is how to tolerate the intra-class variety. In such cases, the granuality of class differences is low, meaning the differences are easy to observe.

On the other hand, for the instance-level problems, for example when distinguishing the face images of two persons as in Figure 1.2, the differences are very subtle because these images contain the same object type. Objects from the same type have the identical structure and possibly very similar object shapes. Their color and texture are sometimes very close as well. For this type of problem, the class distinguishing features are observed in very subtle object details, and are usually relatively harder to localize and describe than in the basic-level tasks. In such cases, the granuality of class differences is high.

Figure 1.2: Examples of the instance-level recognition tasks.

There is another family of problems that connect these two families we just described. In the new family, the granularity level of class differences lies between the basic-level problems and the instance-level problems. The objective of classification is still an object category, but the class differences are not as large as the basic-level tasks. Instead, similarities are commonly shared among different object classes. This family of problems is called "fine-grained" object recognition, which usually refers to recognizing sub-ordinate level object classes, like recognizing bird species, dog species, etc. In this case, different bird species belong to different classes. When the image classification problem is extended to a fine-grained level, the number of classes increase significantly, and the task usually becomes more interesting and useful.

Fine-grained object recognition is an example of the fine-grained classification problems. A similar concept also applies to other computer vision tasks as well, such as 3D shape analysis. In comparison with a task of identifying rabbit shape and horse shape, identifying the shape of a human organ with different illness conditions involves more detailed measurements of the shape deformation. Such fine-grained tasks usually take place in 3D medical image analysis, for example when distinguishing CT images containing human skulls with different medical conditions.

While fine-grained tasks are challenging due to the subtlety of their class differences, they can be very useful when properly addressed. Fine-grained tasks cover a large variety of

problems in the real world, and automatic systems performing fine-grained classification can be used in many applications. The work presented in this thesis is focused on fine-grained image classification problems including the two types of fine-grained problems mentioned above: fine-grained object recognition and fine-grained 3D shape analysis.

1.1.1 Fine-grained object recognition

Object recognition is a major focus of research in computer vision and machine learning. Of all vision tasks, object recognition is key to understanding image data (or videos) because understanding what the objects are in an image is the foundation for deriving any deeper meaning. Object recognition has many direct applications if appropriately addressed, such as in vision systems for automobiles or mobile phones applications.

In the last decade, most of the existing work has been focused on basic recognition tasks: distinguishing different categories of objects, such as tables, computers and humans. Recently, there has been an increasing trend towards working on subordinate-level or fine-grained recognition that categorizes similar objects, such as different types of birds or dogs, into their subcategories. The subordinate-level recognition problem differs from the basic-level tasks in that the object differences are more subtle. Fine-grained recognition is generally more difficult than basic-level recognition for both humans and computers, but it will be widely helpful if successfully used in applications.

One typical fine-grained object recognition task is food recognition. For example, different types of sandwiches share many similarities, such as: the structure containing two buns, the shape, the color and texture of the buns. The differences are very subtle, such as the stuffing that lies between the buns. Good algorithms that recognize different food types can be used in health care applications, such as monitoring the diet of people with obesity problem.

Another example of fine-grained object recognition is clothes recognition. This task has useful applications for commecial websites, such as helping customers to recognize clothing or shoe styles from mobile phone images. Plant recognition covers a family of fine-grained recognition problems, such as: flower recognition, leaf recognition and tree recognition.

These tasks can make very interesting and educational iPhone applications. In fact there is an existing leaf recognition application on the iPhone call "Leafsnap".

Last but not least, recognizing animal subspecies is useful in industries, including argriculture, fisheries, etc. For example in Seattle, lots of fish are caught each year. However there are restrictions on certain species, which need to be protected because they are endangered. In this scenario, an automatic system to identify the fish species from images and videos is very useful. In argriculture, identifying sick farm animals at an early stage is very important. A sick cow, if not identified early, may die or may be expensive to cure. Therefore automatic system to recognize sick farm animals is helpful in this scenario.

In practice when humans are perfoming such fine-grained tasks, a lot of them require expert knowledge to be addressed, such as fish recognition and flower recognition. Therefore, effective algorithms for these tasks will be especially helpful in comparison with the algorithms for the basic-level tasks, which are relatively easier for humans.

1.1.2 Fine-grained 3D shape analysis

As we mentioned previously, shape analysis of human anatomy for a particular medical condition is also a typical fine-grained task. For example, the shapes of human skulls are very similar, and the differences can only be observed by quantifying the shape details. In particular, illnesses that cause deformation of human skulls are reflected in the detailed deformation of the skull shapes.

Craniosynostosis is a common congenital condition in which one or more of the fibrous sutures in an infant's calvaria fuse prematurely, resulting in restricted skull and brain growth. Because the brain cannot expand perpendicular to the fused suture, it redirects growth in the direction of the open sutures, resulting in abnormal head shape and in some cases, facial features. Craniosynostosis results in head deformity that can be severe if it is not corrected surgically. This condition may result in increased intracranial pressure on the brain and is correlated with developmental delays, although the cause of such delays is not currently known [111]. It is estimated that the fusion of any one or more sutures occurs in approximately 1 in $2,000$ live births [108]. In clinical practice, craniosynostosis is diagnosed by a

physician on the basis of head shape and confirmatory CT scan. Automatic analysis of CT scans, including a measure of shape deformation, would be of great help to both doctors and medical researchers.

1.2 Existing methods for fine-grained problems

There is relatively limited work focusing specifically on fine-grained tasks. A straightforward idea is to simply apply the methods used for a general task regardless of the granularity of the class differences. However, the classical algorithms for the basic-level tasks and the instance-level tasks are quite different.

At the basic level, because the class difference is very large, classical pipelines are usually based on the Bag-of-Words model, which accumulates all features to capture the global difference. At the instance level, because the class differences are usually hard to identify and the majority of image features are similar, the commonly used methods are focused on finding an accurate registration of each part of the image. Therefore the detailed differences can be precisely pinpointed. However, at the fine-grained level where the granularity level lies in-between, it is hard to apply either of the above approaches. When the basic-level approaches are applied to the fine-grained tasks, it is very easy to ignore the subtle differences because of the large number of similar features across different categories. Simply accumulating all features involves too much noise. When the instance level approaches are applied, however, an accurate registration is almost impossible because the image structures from different classes are usually not identical. Missing parts for registration become the biggest obstacle of these approaches. Therefore, new algorithms need to be developed specifically for the fine-grained tasks.

1.3 Feature engineering

From a high-level perspective, the computer vision research community has made enormous progress in recent years. There are many systems and pipelines for object recognition, object detection and medical image analysis. For example in object recognition, the combination of using SIFT features and Bag-of-Words models has become a standard pipeline that is successfully used in many applications. Researchers have extended the use of SIFT features

to other feature designs or even machine-learned features; and extended the use of Bag-of-Words model to other feature pooling approaches, such as spatial pyramid matching algorithm.

Of all these computer vision systems, one of the most important modules is the exploration of features, which can be called feature engineering. Traditionally, researchers have been using design features and image representations. However, manual design is time cosuming and hard to generalize. One way to automate this process is by automatically generating large numbers of candidate features and selecting the best. More recently, there is a trend toward using feature learning algorithms to generate effective image features and representations, such as kernel descriptors (KDES) and deep belief network (DBN).

The use of automatic feature engineering can be particularly useful for fine-grained tasks, because fine-grained problems require representing rich features in order to capture the subtle class differences. Traditional manually designed features, however, can easily eliminate the important subtle features. In this thesis, different aspects of feature engineering are explored, including manual feature design, feature selection, and feature learning algorithms.

Chapter 2

RELATED WORK

2.1 Object recognition

Object category recognition has been a popular research area in computer vision for many years (see [28] for a comprehensive survey). Some approaches are based on local features such as the SIFT [73] descriptor or global features such as color histograms or GIST [84] features; images are typically represented as "bags" of these features without explicit spatial information. Other approaches, such as the "constellation model" [17] and its numerous extensions, model objects as a collection of parts with a predictable spatial arrangement.

2.2 Fine-grained object recognition

An increasing number of papers have focused on fine-grained object recognition in recent years [16, 56, 32, 119, 122, 131]. In [16], multiple kernel learning is used to combine different types of features and serves as a baseline fine-grained recognition algorithm; human help is used to discover useful attributes. In [131], a random forest is proposed for fine-grained object recognition that uses different depths of the tree to capture dense spatial information. In [56], a multi-cue combination is used to build discriminative compound words from primitive cues learned independently from training images. In [130], bagging is used to select discriminative features from the randomly generated templates. In [30], image regions are considered as discriminative attributes and CRF is used to learn the attributes on a training set with a human in the loop. Pose pooling [136] adapted Poselets [12] to fine-grained recognition problems and learned different poses from fully annotated data. Though the deformable parts model [35] is powerful for object detection, it might be insufficient to capture the flexibility and variability in the fine-grained tasks considered here [85].

2.3 Feature engineering

At the end of the day, some recognition pipelines succeed, and some fail. The most important factor that determines the success of a recognition pipeline is the features used, including local features and image-level representations. From the learning perspective, the raw pixels in images are not in a form that is amenable to learning, and we need to construct features and representation from them.

2.3.1 Feature design

Traditionally, researchers put in their efforts to design effective feature and image representations. Particularly in recent years, there are many effective manually designed features in computer vision. At the low level for example, the use of the SIFT descriptor, the HOG features and Textons have been quite successful. At the high level, the color histogram, Bag-of-Words model and spatial pyramid are very expressive feature representations. The following sections give some more detailed explanations of some of these feature designs.

SIFT

SIFT (Scale-Invariant Feature Transform) descriptors were proposed by David Lowe in 2005 [73]. This descriptor is used to capture the appearance feature of an image patch centered at a certain position in an image. The basic idea of SIFT is to use the statistics of gradients of different parts of a circular image patch. Specifically, the image patch is evenly divided into several regions, each of which covers a certain range of angles of the central pixel (like partitioning a pizza). Then, the histogram of the orientations of the gradient in each of the regions is formed using 8 or 16 bins, and the concatination of all the histograms is used as the feature description of this patch.

HOG

HOG (Histogram of Oriented Gradients) [24] is a similar idea to the SIFT feature descriptor, as indicated by the name. The difference is the method for partitioning an image patch.

While SIFT partitions an image into "pizza" slices, HOG partitions an image patch into 3×3 (for example) grids.

Bag of Words

The Bag of Words (BOW) model [33] was first proposed for natural language processing. In computer vision, it is used as a way of pooling local features to form an image-level representation. The idea is very simple. Given a set of local features (like SIFT) extracted from all over an image, as well as a coding scheme to encode the features, Bag of Words will count the number of different codewords based on the encoded local features. These counts (in the form of a histogram) are used as a representation of the whole image.

Spatial pyramid matching

Spatial pyramid matching [62] is an extension of the Bag of Words model. Instead of pooling features from the whole image, spatial pyramid matching partitions the images into sub-regions with different resolution, forming pyramid-like layers. Then, local features are pooled from each of the sub-regions (similar to BOW), and the representations of each of the sub-regions are concatenated as the global represenation of the whole image.

More specifically in spatial pyramid matching, local features are first extracted, such as SIFT features for images, then all feature vectors are quantized into M types, each of which is called a code word in the codebook. It is assumed that features of the same code word can be perceived equivalent to one another. Spatial pyramid matching works in L levels of image resolutions. In level 0, there is only one grid for the whole image, in level 1, the image is partitioned to 4 grids of the same size, and in level l, the image is partitioned to $(2^l)^2$ grids of the same size, and so on. For two images I_1 and I_2, the spatial pyramid matching kernel K is defined as

$$K(I_1, I_2) = \sum_{l=1}^{L} \sum_{i=1}^{G_l} w_{l,i} K_{l,i}(I_1, I_2) K_{l,i}(I_1, I_2) = \sum_{m=1}^{M} \min(H_{l,i}^m(I_1), H_{l,i}^m(I_2)) \tag{2.1}$$

where, $w_{l,i}$ is the weight for the i-th grid in the l level. In 2.1, it is chosen as:

$$l > 0, \quad w_{l,i} = \frac{1}{2^{L-l+1}}; \quad l = 0, \; w_{l,i} = \frac{1}{2^L}. \tag{2.2}$$

L is the total number of levels and G_l is the total number of grids in level l. $H_{l,i}^m(I_l)$ is the number of code word m appearing in i-th grid of l-th level in image I_l. In practice, it is reported that $L = 2$ or $L = 3$ is sufficient.

2.3.2 Feature selection

Today there are more and more algorithms that automate the feature engineering process. One way this is often done is by automatically generating large numbers of candidate features and selecting the best by some measure, for example their information gain with respect to discriminating the classes. But features that look irrelevant in isolation may be relevant in combination. On the other hand, running a learner with a very large number of features to determine which ones are useful in combination may be too time-consuming, or may cause overtting.

Feature selection is the process of selecting a subset of relevant features for use in model construction. The central assumption when using feature selection techniques is that the data contains many redundant or irrelevant features, and the technique must determine a subset which is most relevant and represenatative of the data. A variety of methods can be used for the purpose of feature selection based on mutual information, feature correlation or some other objective criteria. In our work, we focus on a specific family of embedded methods using sparse logistic regression models. Specifically, L_1 regularization [113] in the logistic regression model is applied to induce sparsity, which can avoid overfitting. This method has been rigorously proven to be effective in selecting relevant features when there are exponentially many irrelevant ones [65]. There are other variations of this model that further consider constraints over feature relationships, such as the fused lasso [114] and the group lasso [76].

2.3.3 Feature learning

In recent years, there is a trend of using feature learning algorithm to automatically generate representative feature from images. Feature learning is particularly useful for ne-grained recognition, because rich appearance and shape features are required for describing subtle

differences between categories. Feature learning approaches provide a natural way to capture rich appearance cues by using a large number of codewords (sparse coding) or neurons (deep networks), while traditional computer vision features, designed for basic-level category recognition, may eliminate many useful cues during feature extraction.

Deep Networks

Deep belief nets [48] learn a hierarchy of features by training multiple layers recursively, using the unsupervised restricted Boltzmann machine (pre-train). The pre-train avoids the shallow local minima, and the learned weights are further adjusted to the current task using supervised information. To make deep belief nets applicable to full-size images, Lee et al. [64] proposed convolutional deep belief nets (CDBN) that use a small receptive field and share the weights between the hidden and visible layers among all locations in an image. Invariant predictive sparse decomposition [51, 55] approximated sparse codes from sparse coding approaches using multi-layer feed-forward neural networks and avoided solving computationally expensive optimizations at runtime. Deconvolutional networks [134] reconstruct images using a group of latent feature maps in a convolutional way under a sparsity constraint. These approaches have been shown to yield competitive performance with the SIFT based Bag-of-Visual-Words model on object recognition benchmarks such as Caltech-101 dataset.

Sparse Coding

Recent research has shown that single layer sparse coding on top of SIFT features achieves the state-of-the art performance on many object recognition benckmarks [63, 127, 121, 13, 22, 132]. Yang et al. [127] learned sparse codes over SIFT features instead of raw image patches using sparse coding approaches. Their comparisons have suggested that such an approach outperforms the standard bag-of-visual-words model and convolutional deep belief networks. Wang et al. [121] presented a fast implementation of local coordinate coding that computes sparse codes of SIFT features by performing local linear embedding on several nearest visual words in the codebook learned by K-Means. Boureau et al. [13] compared

many types of feature learning algorithms and found that the SIFT-based sparse coding approaches followed by spatial pyramid max pooling works well, and the macrofeatures can boost recognition performance further. Coates and Ng [22] evaluated many feature learning approaches by decomposing them into training and encoding phases, and suggested that the choice of architecture and encoder is the key to a feature learning system.

Kernel Descriptors

Although the hand-designed low-level features such as SIFT and HOG are very successful, we still lack a deep understanding of the design rules behind them and how they measure the similarity between image or point cloud patches. Recent work on kernel descriptors [6, 5] tries to understand these questions, and shows that these hand-designed features are equivalent to a certain type of match kernel over image patches. Based on this insight, a family of kernel descriptors are proposed, which are able to turn pixel-level attributes such as gradient, color, size and shape into feature vectors. Kernel descriptors [6] avoid the need for pixel attribute discretization by adopting a kernel view of patch similarity. Furthermore, kernel descriptors are extremely flexible, since the distance function between pixel attributes can be any positive definite kernel, such as the popular Gaussian kernel function. In fact, Bo and colleagues showed that histogram features such as SIFT are a special, rather restricted case of match kernels [6]. The previous experiments have shown that kernel descriptors obtain superior recognition accuracy to hand-designed features, sparse coding [127] and deep networks [64] on many object recognition benchmarks [6, 5]. The fact that kernel descriptors are able to capture pixel-level object cues from raw data make them very suitable for fine-grained object recognition. In this work, we adapt kernel descriptors as feature extractors for fine-grained object recognition.

Kernel descriptors [6] provide a unied framework for turning pixel level attributes such as gradients into patch level features, by highlighting the kernel view of orientation histogram features such as SIFT and HoG. Feature extraction using kernel descriptors involves three steps: (1) designing kernels for matching patches using pixel attributes; (2) learning a compact set of basis vectors using kernel principal component analysis (KPCA); (3) constructing

kernel descriptors by projecting the innite-dimensional feature vectors induced by pixel attributes to the learned basis vectors. Following the notations in 2.3, the gradient match kernel $K_{grad}(P,Q)$ measures the similarity between patches P and Q based on the pixel gradient attribute:

$$K_{grad}(P,Q) = \sum_{z\in P}\sum_{z'\in Q} \tilde{m}(z)\tilde{m}(z')k_o(\tilde{\theta}_z, \tilde{\theta}_{z'})k_p(z,z') \tag{2.3}$$

where z denotes the 2D position of a pixel in an image patch normalized to [0,1] and $\tilde{\theta}_z$ is the normalized gradient vector dened as:

$$\tilde{\theta}_z = [\sin(\theta(z)), \cos(\theta(z))] \tag{2.4}$$

The position kernel k_p and orientation kernel k_o are dened using Gaussian kernels as:

$$k_p(z,z') = e^{-\gamma_p||z-z'||^2} k_o(\tilde{\theta}_z, \tilde{\theta}'_z) = e^{-\gamma_o||\tilde{\theta}_z - \tilde{\theta}'_z||^2} \tag{2.5}$$

Based on the patch similarity kernel dened in equation 2.5, kernel descriptors extract compact low-dimensional features by sampling sucient basis vectors uniformly and densely from each pixel attribute's support region, and then learning compact basis vectors using KPCA. Particularly, the gradient kernel descriptor for a patch P will have the form:

$$F^t_{grad}(P) = \sum_{i=1}^{d_o}\sum_{j=1}^{d_p} \alpha^t_{ij} \left\{ \sum_{z\in P} \tilde{m}(z)k_o(\tilde{\theta}(z), x_i)k_p(z,y_j) \right\} \tag{2.6}$$

where $\{x_i\}_{i=1}^{d_o}$ and $\{y_i\}_{j=1}^{d_p}$ are uniformly sampled from the corresponding support regions and d_o and d_p are the sizes of the basis vectors for the orientation and position kernels respectively.

Chapter 3

FEATURE SELECTION FOR FINE-GRAINED SHAPE ANALYSIS

This chapter is focused on classification of human skull shapes. The shapes of human skulls are generally similar to each other, because they all have the same components and deformation appearance on the 3D surface. The differences that distinguish different skulls are usually the degree of deformation of varying parts of the skulls, for example, the width of the skull, or how pointy the top of the skull is. For this reason, shape analysis of human skulls is identified as a fine-grained task, in which the class distinguishing differences are in the details.

3.1 Introduction

Specifically, we have focused on analyzing the shape of human skulls extracted from CT images for patients with craniosynostosis. In our work, we developed a system to identify craniosynostosis based on the 3D shape of the skulls. While classification is performed clinically by doctors, it is important to note that there is a lack of criteria to quantify the severity of the abnormality for research purposes. For example, when estimating the relative effects of different surgical methods on craniofacial shape, quantitative measurement is essential. So the model we developed for classification is also able to retrieve CT images based on quantification of the severity of the abnormality of the 3D skull shape.

Given a data set containing pre-operative and post-operative CT scans of subjects with three classes of craniosynostosis (coronal, metopic and sagittal) plus a set of scans from similar-age control subjects, we conducted a set of experiments in classification, quantification and retrieval using four logistic regression methods [129]. Different sparse logistic regression models were compared in terms of misclassification on whether a skull has craniosynostosis or not. Then a set of CT image retrieval experiments were conducted using the best model for our data - the clustering lasso, a method developed particularly in this

work. Finally the abnormality of the skulls of the same patient before and after surgery were compared using the same quantification criteria. Two sparse logistic regression models were also used for feature selection, the results of which reflect abnormal region localization on the 3D surfaces of the skulls.

The rest of the chapter is organized as follows. Section 3.2 summarizes the related literature on the medical problem of craniosynostosis, Section 3.3 gives an overview of the framework of our approach for shape classification, abnormality quantification and abnormal region localization, Section 3.4 describes the details on how logistic regression models are used in this framework, and Section 3.5 shows the experimental results of our work.

3.2 Related literature on craniosynostosis

Calvarial (skull) abnormalities are frequently associated with severely impaired central nervous system functions due to brain abnormalities, increased intra- cranial pressure and abnormal build-up of cerebrospinal fluid.

There are some previous studies on examining the specific skull shapes of patients. Previously, we proposed the cluster lasso [129], a logistic regression model, for classification of three types of craniosynostoses: coronal, metopic and sagittal. Lin et al. [71] developed symbolic shape descriptors to classify skull deformities caused by metopic and sagittal synostoses. Ruiz-Correa et al. [97] used a set of scaphocephaly severity indices (SSIs) for predicting and quantifying head- and skull-shape deformity in children diagnosed with isolated sagittal synostosis (ISS). In [104], Shapiro et al. introduced several different craniofacial descriptors that have been used in studies of two craniofacial disorders: 22q11.2 deletion syndrome (a genetic disorder) and deformational plagiocephaly/brachycephaly. They provided feature extraction tools for the study of craniofacial anatomy from 3D mesh data obtained from the 3dMD active stereo photogrammetry system. These tools produce quantitative representations (descriptors) of the 3D data that can be used to summarize the 3D shape as pertains to the condition being studied and the question being asked. This work is different from the current study in that it analyzed the shape of the midface and back of the head, while our work focuses on the shape of the skull.

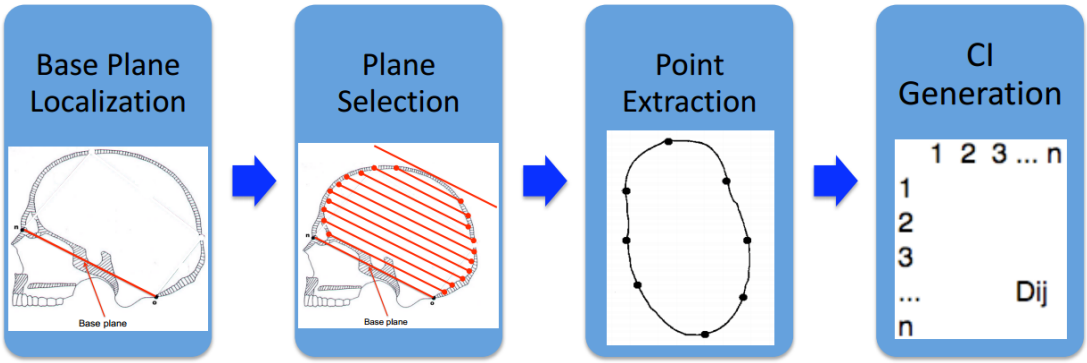


Figure 3.1: Surface points extraction: the first three modules are the three steps for extracting surface points; the last image is the distance matrix generated from the surface points.

3.3 The Pipeline of 3D Shape Analysis

3.3.1 System Design

A system was built for skull shape analysis according to its abnormality. With an input of 3D CT volume data of random pose, our system first extracts the skull and performs pose normalization, so that it is symmetric with respect to the right and left sides. Then, surface points are extracted that are evenly spaced all over the skull. After that, a shape feature called the cranial image [71] is calculated by computing pairwise distances of these points. Last, learning models are applied to analyze the skulls using the shape feature. Figure 3.1 summarizes the process.

3.3.2 Surface Points Extraction

The first step of the module of surface points extraction is to locate a base plane on the skull based on two important landmarks: the nasion and the opisthion. The base plane goes through these two biological landmarks, and is perpendicular to the symmetry plane that separates the right and left sides.

The nasion is the intersection of the frontal and two nasal bones of the human skull [44].

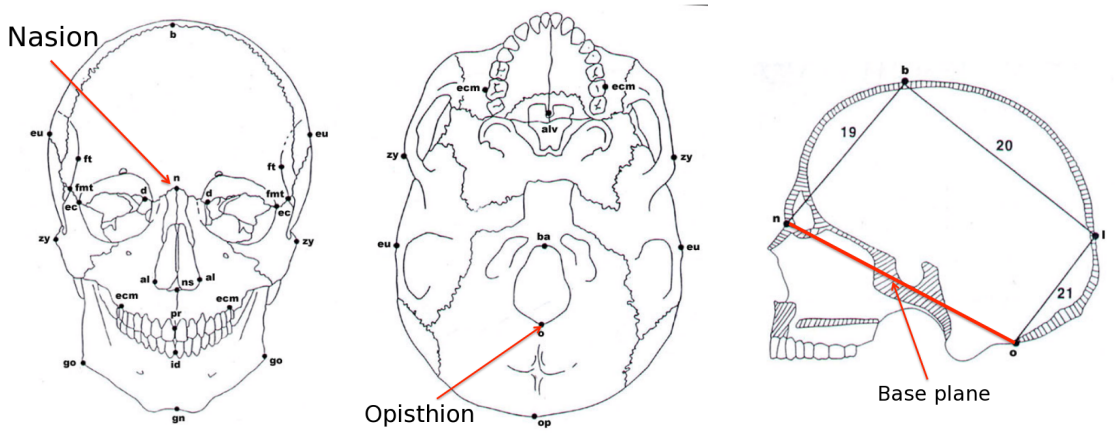


Figure 3.2: Nasion, opisthion and the base plane of the skull: the nasion is the intersection of the frontal and two nasal bones of the human skull [44]; the opisthion is the mid-point on the posterior margin of the foramen magnum on the occipital bone [44]; the base plane is the plane that goes through the nasion and the opisthion and is perpendicular to the middle plane of the head.

Its manifestation on the visible surface of the face is a distinctly depressed area directly between the eyes, just superior to the bridge of the nose. The opisthion is the mid-point of the posterior margin of the foramen magnum on the occipital bone [44]. The two points were chosen because of their locations at the front and back of the head, and because they are stable during the human growth process. The nasion and opisthion are detected as follows (Figure 3.2). First, the plane of symmetry of the left and right sides of the skull on which the landmarks are expected to be is extracted. Then, the tip of the nose is located as the point with the smallest horizontal value. The nasion is located as the point closest to the tip of the nose, which is above it and which has a zero curvature in the vertical direction. The opisthion is located as the point in the left part of the outline, which has the closest distance to the nasion of all points below the nasion.

Our shape measure is based on the distances between points on the surface of the skull, so the second step is to extract a set of planes from which surface points are located.

These planes are parallel to the base plane. The top plane of the skull is a plane that has intersection with the skull and which is parallel to the base plane but has the furthest distance to the base plane. Our system can extract any plane that is parallel to the base plane and located between the base plane and the top plane of the skull, based on the ratio of its distance to these two planes. Multiple planes may be selected with equal distances among them. In the rest of our experiments, 10 planes that are evenly distributed across the whole skull were used to provide a rich 3D shape descriptor.

In the third step of this module, N points are evenly extracted along the outlines of the planes from the previous step. N is chosen by the user, and $N = 100$ in our experiments.

3.3.3 Cranial Image Generation

Our shape feature is a $N \times N$ pairwise distance matrix among the surface points from the previous step. The number at position (i, j) of the matrix represents the distance between point number i and point number j (the last module of Fig. 3.1). The matrix is symmetric. Such a shape feature is a rich representation of the skull shape, and its dimension is usually very high (over 10^4).

3.3.4 3D Shape Analysis

The task of the system is to analyze skull shapes for abnormality recognition and CT images retrieval. To recognize skull abnormality, we explore several logistic regression models to fit the data with the high dimensional shape feature (CI), and perform classification. To rank the CT images, a qualification criterion for shape abnormality is needed. We use the models to generate quantification scores that represent the severity of the abnormality.

Geometrically, each feature corresponds to two points on the skull surface; thus selected features resulting from the models that perform feature selection can represent the skull regions that contribute most to the shape deformation of the skulls.

3.4 Sparse Logistic Regression Models for Analyzing CI Features

3.4.1 Sparse Logistic Regression Models

In this work, we explore logistic regression and three sparse logistic regression models for the purpose of classifying three different types of craniosynostosis. This section describes our methodology.

Logistic Regression

Logistic regression is a workhorse in machine learning that uses a generalized linear model for binomial regression. In logistic regression model, the probability of being classified as one class is a linear function of the features.

$$p(y|\mathbf{x}, \mathbf{w}) = \frac{1}{1 + exp(-y(\mathbf{w}^T\mathbf{x} + w_0))} \tag{3.1}$$

where vector $\mathbf{x}$ contains the feature values of a data sample; y is its class label (for example, $y = 1$ refers to coronal and $y = -1$ refers to non-coronal), $\mathbf{w}$ contains the coefficients for $\mathbf{x}$, and w_0 is the intercept. Furthermore, w_0 and $\mathbf{w}$ are model parameters, and $p(y|\mathbf{x}, \mathbf{w})$ is the probability that a data sample belongs to a certain class.

We can estimate the optimal parameters w_0 and $\mathbf{w}$ that minimize the following loss function:

$$l(w_0, \mathbf{w}) = \sum_{i=1}^{n} \log(1 + \exp(-y_i(\mathbf{w}^T\mathbf{x_i} + w_0))) \tag{3.2}$$

$$\{w_0, \mathbf{w}\} = \min_{w_0, \mathbf{w}} l(w_0, \mathbf{w}) \tag{3.3}$$

where y_i is the actual class label of a data sample $\mathbf{x_i}$.

L_1 Regularized Logistic Regression

Due to the high-dimensionality of the data (i.e. a large number of features and a modest size of samples), learning the unregularized logistic regression [3.3] will result in overfitting. To avoid overfitting, L_1 regularization is usually applied to induce sparsity in the solution $\mathbf{w}$ such that many of the coefficients in $\mathbf{w}$ are set to exactly zero. L_1 regularization [113]

has been rigorously proven to be effective in selecting relevant features when there are exponentially many irrelevant ones [65]. The log-likelihood of L_1 regularized logistic regression is as follows.

$$l(w_0, \mathbf{w}) = \sum_{i=1}^{n} \log(1 + \exp(-y_i(\mathbf{w}^T \mathbf{x_i} + w_0))) + \lambda \sum_{i=1}^{m} |w_i| \tag{3.4}$$

where λ is a regularization parameter for the L_1-norm of the coefficients.

Fused Lasso

One problem with L_1 regularization is that when features are highly correlated, it arbitrarily chooses one of many correlated features. Some variations of L_1 regularization can better exploit the underlying structure of our feature data. Specifically, the fused lasso induces bias from prior knowledge such that correlated feature groups will be assigned similar weights. In this work, the fused lasso [114] places a constraint on the weights of the features that are geographically related - sharing the same or neighboring surface points.

The loss function of the fused lasso with induced bias is,

$$\begin{aligned} l(w_0, \mathbf{w}) = & \sum_{i=1}^{n} \log(1 + \exp(-y_i(\mathbf{w}^T \mathbf{x_i} + w_0))) \\ & + \lambda \sum_{i=1}^{m} |w_i| + \mu \sum_{\{w_i, w_j\} \in M} |w_i - w_j| \end{aligned} \tag{3.5}$$

where μ is a regularization parameter for the new penalty term. M is a set that contains all pairs of features that are neighbors, whose endpoints are the same or next to each other. In equation [3.5], $\lambda \sum_{i=1}^{m} |w_i|$ penalizes large feature weights, and $\mu \sum_{\{w_i, w_j\} \in M} |w_i - w_j|$ penalizes large weight differences between correlated features.

Clustering Lasso

As mentioned before, L_1 regularized logistic regression tends to assign different weights to highly correlated features. When features are highly correlated, it arbitrarily chooses one of them and assigns a non-zero weight only to it. The fused lasso is one way to avoid this problem by placing constraints on the weight differences based on prior knowledge. However,

this requires the model to know ahead of time the right grouping of the features. An alternative to using such prior knowledge, is to penalize the weight differences of correlated features.

We propose a new form of regularization logistic regression, namely the clustering lasso (or cLasso) [129]. The model for the clustering lasso is:

$$p(y|\mathbf{x}, \mathbf{w}, \mathbf{w^c}) = \frac{1}{1 + exp(-y(\mathbf{w}^T\mathbf{x} + \mathbf{w^c}^T\mathbf{c} + w_0))} \tag{3.6}$$

where $\mathbf{x}$ contains the feature values of a data sample; y is its class label (for example, $y = 1$ refers to sagittal and $y = -1$ refers to non-sagittal); $\mathbf{c}$ are the cluster centers of $\mathbf{x}$; $\mathbf{w}$ contains the coefficients for $\mathbf{x}$; $\mathbf{w^c}$ contains the coefficients for $\mathbf{c}$; and w_0 is the intercept. Furthermore, w_0, $\mathbf{w}$ and $\mathbf{w^c}$ are model parameters, while $p(y|\mathbf{x}, \mathbf{w}, \mathbf{w^c})$ is the probability that a data sample belongs to a certain class.

The loss function for the cLasso is

$$l(w_0, \mathbf{w}, \mathbf{w^c}) = \sum_{i=1}^{n} \log(1 + \exp(-y_i(\mathbf{w}^T\mathbf{x_i} + \mathbf{w^c}^T\mathbf{c_i} + w_0))) + \lambda \sum_{i=1}^{m} |w_i| + \nu \sum_{i=1}^{k} |w_i^c| \tag{3.7}$$

where $\mathbf{c_i}$ ($i \in [1, k]$) is the centroid of a group of features $\{x_{i_1}$,x_{i_2},...,$x_{i_k}\}$ (its feature value is their average); w_i^c is the weight for c_i; and ν is the regularization parameter for the weights of the cluster centers.

This loss function is designed to cluster the features based on their correlation, and penalize their shared weights (w_i^c) and individual weights ($w_{i_1}, w_{i_2}, ...,$
w_{i_k}) respectively. When ν is small and λ is large, individual weights are penalized, and features tend to be split into groups based on their correlation and to share the same weights. When λ is large enough, this model is equivalent to the model of L_1 regularized logistic regression (equation [3.4]).

Parameter w_i^c encourages correlated features to share the same weight, and w_i allows unique features to be used. Therefore, the cLasso is equivalent to using only shared weights (w_i^c) when each centroid c_i ($i \in [1, k]$) is computed as a weighted average with the weights determined by $(w_{i_1}, w_{i_2}, \ldots, w_{i_k})$.

3.4.2 Abnormality Quantification

Using the models to fit to the data, the predicted probability of a sample data set $\mathbf{x}$ being a certain class $p(y = 1|\mathbf{x}, \mathbf{w})$ can be viewed as a quantification measure of skull abnormality. However, the use of the sigmoid function $P(t) = \frac{1}{1+e^{-t}}$ in computing the probability does not work well as a quantification criteria. Because a regression model that fits the data well tends to assign a value close to 1 to all positive instances, the quantification results are too similar. Instead, we use the linear function of the features before taking the sigmoid function to obtain the probability. The second option produces a better quantification measure, because the linear function differentiates abnormal skulls better, even when they are classified as the same class.

For the logistic regression, lasso and fused lasso models, the quantification scores are

$$S(\mathbf{x}) = -y(\mathbf{w}^T\mathbf{x} + w_0) \tag{3.8}$$

For the clustering Lasso, the quantification score is

$$S(\mathbf{x}) = -y(\mathbf{w}^T\mathbf{x} + \mathbf{w}^{\mathbf{c}T} + w_0) \tag{3.9}$$

3.5 Experiments

The experiments were designed to show the ability of our system to retrieve CT images based on skull shape abnormality of different types of craniosynostosis. As a measurement for performance, we also provide phenotype prediction accuracy by comparing our prediction with the groundtruth diagnoses of the doctors. Cranial images were generated using our system with 10 planes and 10 points on each plane. Logistic regression, L_1 regularized logistic regression, the fused lasso and the cLasso were compared in terms of phenotype prediction accuracy, while the quantification measure of cLasso was used as the ranking criterion in our system. Implementation of L_1 regularized logistic regression is from the authors of [65], and implementation of the fused lasso is the machine learning package SLEP [72]. In our previous work, we did a thorough study on choosing regularization parameter λ, μ and ν in equations 3.4, 3.5 and 3.7 for the sparse logistic regression models. We continue to use these parameters.

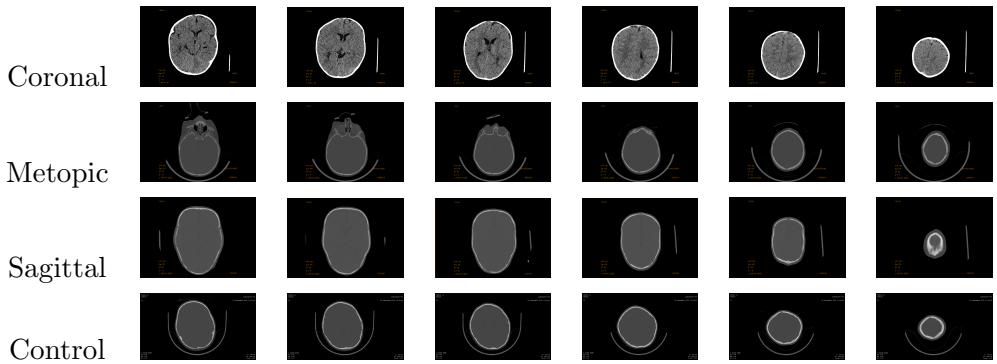


Figure 3.3: Examples of CT image slices

3.5.1 Medical data

Our system was tested on 3D CT images of children's heads from hospitals in four different cities in the US. There are different types of craniosynostosis depending on the affected suture; the sagittal suture is between the parietal bones, metopic between the frontal bones, coronal between the frontal and parietal bones, and lambdoid between the parietal and occipital bones. Our study is focused on three types of synostosis - sagittal, metopic, and coronal. In total we examined approximately 200 CT image volumes, each comprising a stack of image slices (approximately 150 slices per volume). About half the data are controls (normal skulls), and the other half have one of the three types of craniosynostosis. Fig. 3.3 shows some examples of slices from the CT image stacks for all four classes.

3.5.2 Parameter Selection

In the second experiment for multi-plane classification, the cranial images were generated from 10 planes that were evenly spaced across the skull. 10 points were extracted from each plane, and a cranial image with 10,000 features was used for classification using logistic regression and its variations with different regularization terms. Misclassification rates were measured using 3-fold cross validation.

The first problem in conducting these tests was to determine the value of the regularization parameters λ, μ and ν in equations [3.4], [3.5] and [3.7]. The effect of these regular-

ization parameters on classification accuracy was explored by testing the misclassification rate of sagittal versus coronal with fixed training and testing sets. Fig. 3.4 examines the relationship of λ, μ, ν and the misclassification rate in equation [3.4], [3.5] and [3.7]. Based on the observation in the figures, the classification error is the lowest when $0.5 \leq \lambda \leq 1$, $\mu \approx 0.2$ and $\nu < 0.1$.

In the classification task, the regularization parameters were found using 10-fold cross validation on the training set. Specifically, the training set for each test was divided into 10 sub-folds. Nine of them were used for training with certain regularization parameters, and the other fold was used for testing the misclassification rate. The parameter setting with the lowest average rate across all 10 sub-folds was chosen.

3.5.3 Evaluation on Classification

In the first part of our experiment, we evaluate our approach based on its prediction of whether a skull is abnormal or not, meaning the misclassification rate of each class. Classification results using the four different models are shown in Table 3.1. The results of logistic regression substantiates the overfitting problem with the large number of features it uses. The misclassification rate greatly improves when regularization is used in the logistic regression model. Specifically, the clustering lasso exhibits the best results on average. The misclassification rate of the coronal class is higher than the other two classes. This is because the coronal class is the most similar to the controls of all three classes. Its shape deformation is not immediately obvious as the deformation of the sagittal and metopic classes, as can be observed from Figure 3.3. This is consistent with the results from [129].

3.5.4 Abnormality Quantification of Pre- and Post-Operative Skulls

Besides testing our quantification measure on the CT images of patients when they are diagnosed, we also tested it on their CT scans two years after skull surgery was performed. Although the skull abnormality is corrected with surgery, it tends to relapse toward the original deformity with time. Our quantitative severity measure provides an objective measure for the comparison. Fig. 3.5 shows the comparison results for a set of pre-operative and post-operative skulls. The pair of skulls in each column are from the same patient. The

Misclassification Rate	Coronal vs Control	Metopic vs Control	Sagittal vs Control
Logistic regression	37.5%	36.25%	30%
L_1 regression	26.3%	8.75%	**8.75%**
Fused lasso	16.3%	21.3%	**8.75%**
Clustering lasso	**14.1%**	**7.5%**	**8.75%**

Table 3.1: Misclassification rates using multiple planes for three types of craniasynostosis versus controlled skulls. Four different logistic regression models are used for comparison. Classo performs the best on all three types.

abnormality reduction after surgery is according to our scoring method. This shows that the surgeries resulted in improvement in all cases even after two years of growth.

3.5.5 Evaluation of Skull Retrieval

There is no gold standard for evaluating the quantification results, because an objective judgement of severity of craniosynostosis in the form of a medical test does not exist. Medical experts are accustomed to providing diagnoses but have no scoring criteria. In fact, our work was motivated by the need for severity quantification in medical research. However, we were able to have a craniofacial expert rank a subset of the skulls for each type of abnormality for our comparisons.

Based on the quantification measure using the clustering lasso, each skull was assigned a value that represents the degree of severity of the craniosynostosis type to which it belongs (coronal, metopic or sagittal). The results for these skulls in each class are shown in Fig. 3.6. (The numbers are normalized from -1 to 1; 0 to 1 means a subject is abnormal, and -1 to 0 means a subject is a control.) This result provides a useful measurement for physicians and researchers to quantify the severity of individual cases with different types of craniosynostosis. Our results correlate well with expert measures. Most of the orderings are only slightly permuted from the expert's ranking. This is because some of the severity scores are very similar in themselves, so any of them can be ranked before the others in a subjective ordering. There are several exceptions on which our system and the expert

disagree in an obvious way, such as No. 4 in the metopic group and No. 9 in the sagittal group in Fig. 3.6. Discussion with the expert disclosed that different parts of the skull were being weighted differently by the system and the expert. For example for No. 4 in the metopic group, the expert ranked it last because his focus, the pointy shape at the front, is not obvious in the skull. However, our system ranked it higher because its global shape is more similar to a typical metopic skull. This discovery reversely inspired the expert to notice certain shape features that he had previously ignored. The expert also told us that our quantification captured some shape features that could not be observed from the three standard views by doctors. This is evidence that our system would be helpful for clinicians to make more accurate diagnoses.

3.5.6 Visualization of Abnormal Region Localization

Based on the definition of Cranial Image, each feature represents the geometric distance between two surface points on the skull. The feature selection result from model fitting can be used directly as abnormal region selection of a certain craniosynostosis. Specifically, the basic Lasso model selects regions without any constraints, while the fused lasso model selects regions based on priors placed upon the model - nearby regions tend to be selected collaboratively.

The interest region localization results using L_1 regularized logistic regression and the fused lasso are shown in Fig. 3.7 and Fig. 3.8, which show the selected points and their distances on a skull, both before (using L_1 regularized logistic regression) and after (using the fused lasso) the constraint on neighboring features is placed. Fig. 3.7 shows that the selected points are scattered all over the skull. As can be seen from Fig. 3.8, penalizing the group term helps to form local shape regions on the skulls, which shows that the fused lasso successfully enforces geographically related features to be selected together. The visualization of pairwise distances in Fig. 3.8 also shows that these distances are mainly between points on either side of the skull and points on the back of the skull.

3.6 Conclusions

In this chapter, we presented a system that performed skull analysis and severity based retrieval for patients with craniosynostosis. The system was tested with four different logistic regression models: logistic regression, L_1 regularized logistic regression, the fused lasso and the clustering lasso on classifying abnormal skulls from normal ones. The cLasso model was used for quantification of skull shape abnormality. Our experimental results validated the model, both by the error rate on the classification task and by comparison with expert ranks on the retrieval task. This retrieval system provides a convenient tool that can help medical researchers to quantify craniosynostosis for research studies. For example, the methods reported here would facilitate studies of the effects of different surgical methods on cranial shape, associations between severity of cranial deformation and subsequent neurodevelopmental outcomes and the relation of severity to genetic processes.

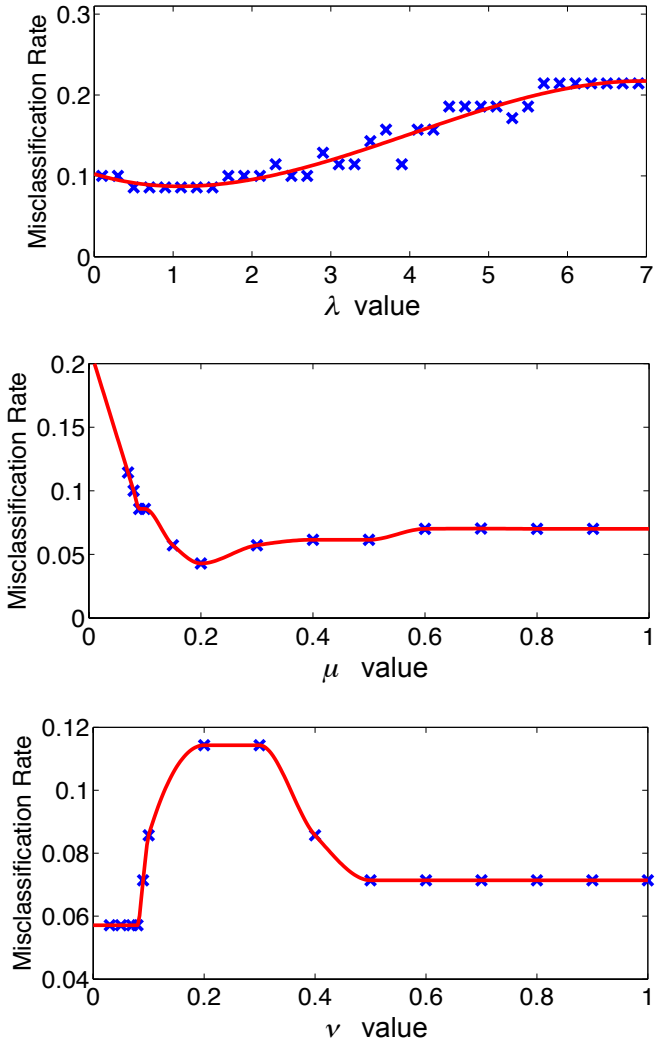


Figure 3.4: Misclassification rate vs regularization parameters: the x axis is the value of regularization parameter λ, μ and ν; the y axis is the misclassification rate achieved using the value.

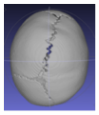
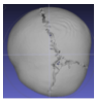
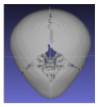
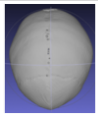
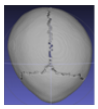
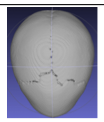
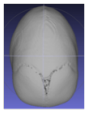
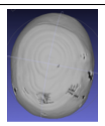
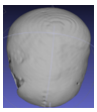
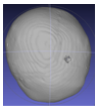
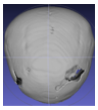
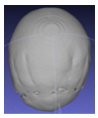
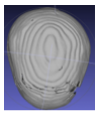
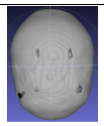
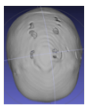
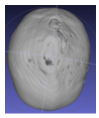
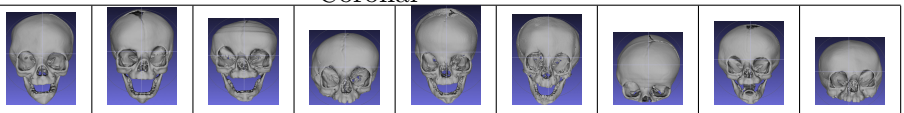
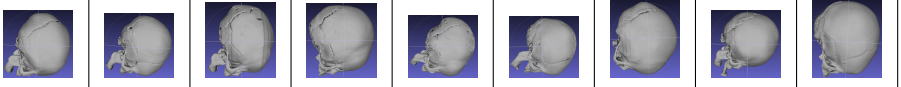
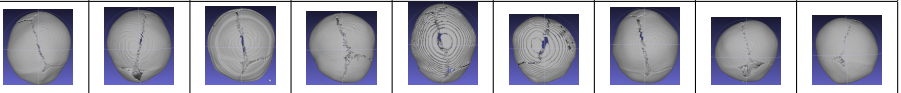
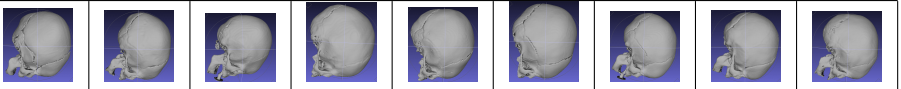
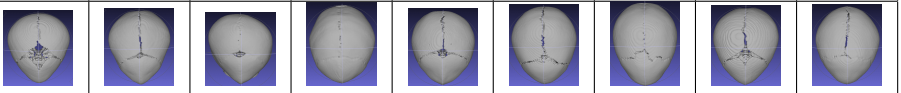
T	Cor	Cor	Cor	Met	Met	Met	Sag	Sag	Sag
Pre									
	0.97	0.92	0.72	1.00	0.54	0.98	0.35	0.58	0.58
Pos									
	0.42	0.38	0.39	0.55	0.31	0.55	0.19	0.38	0.49
Rat	43%	41%	55%	55%	59%	56%	55%	65%	84%

Figure 3.5: Quantification comparison of pre-operative and post-operative skulls: these images are the superior views of nine skulls (for each symptom type). Under the images are their severity scores. The upper row contains pre-operative skulls, and the bottom row are the post-operative skulls of the same subjects as the pre-op ones above it. A subject is normal if the score is ≤ 0. The larger a number is, the more abnormal it is.

Coronal

Frontier									
Superior									
Lateral									
score	0.998	0.992	0.970	0.921	0.876	0.868	0.864	0.853	0.715
ours/expt	1/1	2/4	3/3	4/5	5/2	6/8	7/9	8/6	9/7

Metopic

Frontier									
Superior									
Lateral									
score	1.00	0.869	0.845	0.829	0.823	0.728	0.707	0.666	0.652
ours/expt	1/4	2/2	3/1	4/8	5/5	6/6	7/9	8/3	9/7

Sagittal

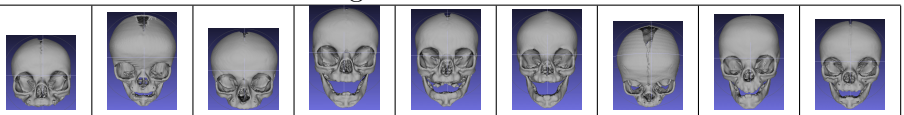
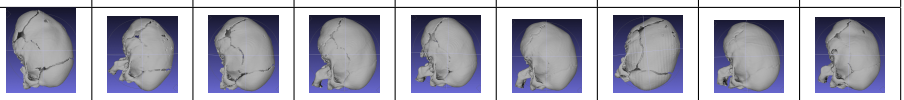
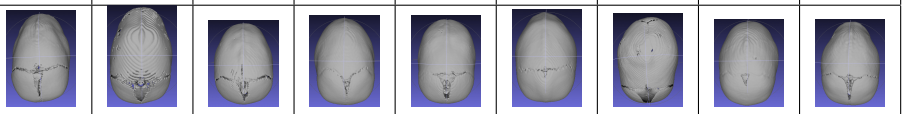
Frontier									
Superior									
Lateral									
score	1.00	0.638	0.583	0.579	0.501	0.462	0.394	0.357	0.351
ours/expt	1/1	2/3	3/4	4/5	5/7	6/6	7/9	8/8	9/2

Figure 3.6: Quantification results: nine skulls are shown for each type of synostosis (coronal, metopic, and sagittal) from three different views. They are ordered by the severity scores produced by our system for the craniosynostosis type to which they belong. Under the images are their severity scores, their ranks by our system (ordered 1-9, with 1 being most severe), and expert ranks for comparison.

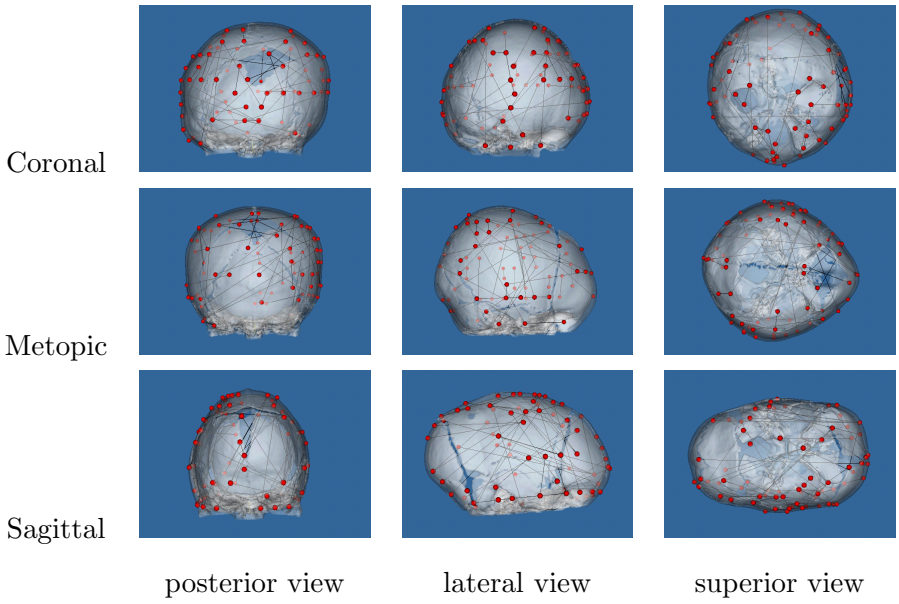


Figure 3.7: Visualization of selected points together with their pairwise distances using L_1 regularized logistic regression: the first row are the results for coronal; the second row is the result for metopic; the third one is the result for sagittal. The left column are the posterior views of the skull; the middle column are the lateral views; and the right column are the superior views.

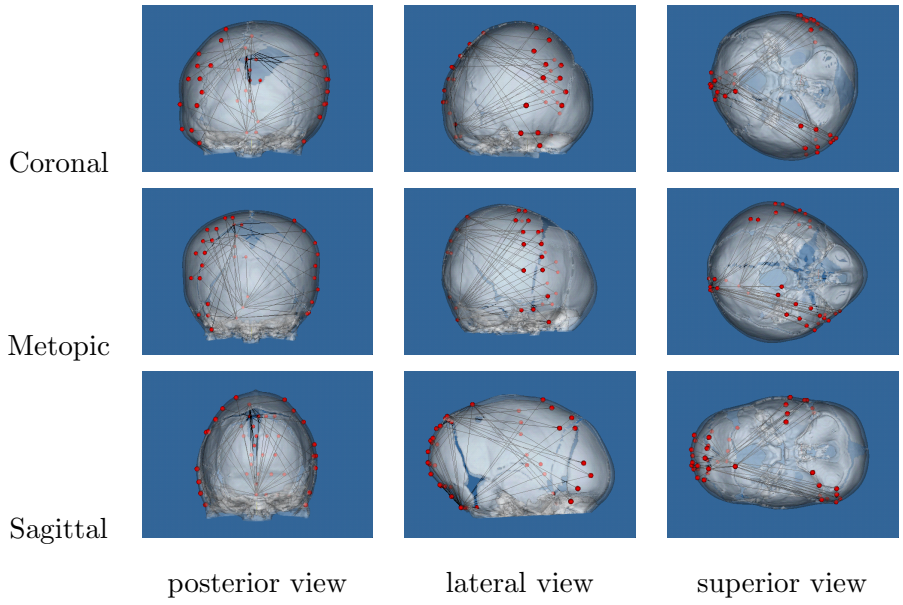


Figure 3.8: Visualization of selected points together with their pairwise distances using the fused lasso: the first row are the results for coronal; the second row is the result for metopic; the third one is the result for sagittal. The left column are the posterior views of the skull; the middle column are the lateral views; and the right column are the superior views.

Chapter 4

FEATURE EXTRACTION IN FOOD RECOGNITION

4.1 Introduction

Food recognition, as a typical fine-grained object recognition problem, is emerging as an important research topic because of the demand for better dietary assessment tools to combat obesity. The goals of such systems are to enable people to better understand the nutritional content of their dietary choices and to provide medical professionals with objective measures of their patients' food intake [110]. People are not very accurate when reporting the food that they consume. As an alternative to manual logging, we investigate methods for automatically recognizing foods based on their appearance. Unfortunately, the standard object recognition approaches based on aggregating statistics of descriptive local features perform poorly on this task [21] because food items are deformable and exhibit significant intra-class variations in appearance; the latter is the case even for relatively standardized items from fast food menus.

This research work was motivated by the observation that a food item can largely be characterized by its ingredients and their relative spatial relationships. For instance, sandwiches are often composed of a layer of meat surrounded on either side by slices of bread, while salads consist of assorted greens whose spatial layout can vary widely. Our hypothesis is that although detectors for any given ingredient are likely to be unreliable, one can still derive sufficient information by aggregating pairwise statistics about ingredient types and their spatial arrangement to reliably identify food items, both at the coarse (e.g., sandwich vs. salad) and fine (e.g., Big Mac vs. Baconator) levels. Fig. 4.1 illustrates how we can exploit the spatial relationship between ingredients to identify this item as a Double Bacon Burger from Burger King.

Although this hypothesis has intuitive appeal, applying it to food recognition is challenging for several reasons. First, even a simple food item can contain numerous visually

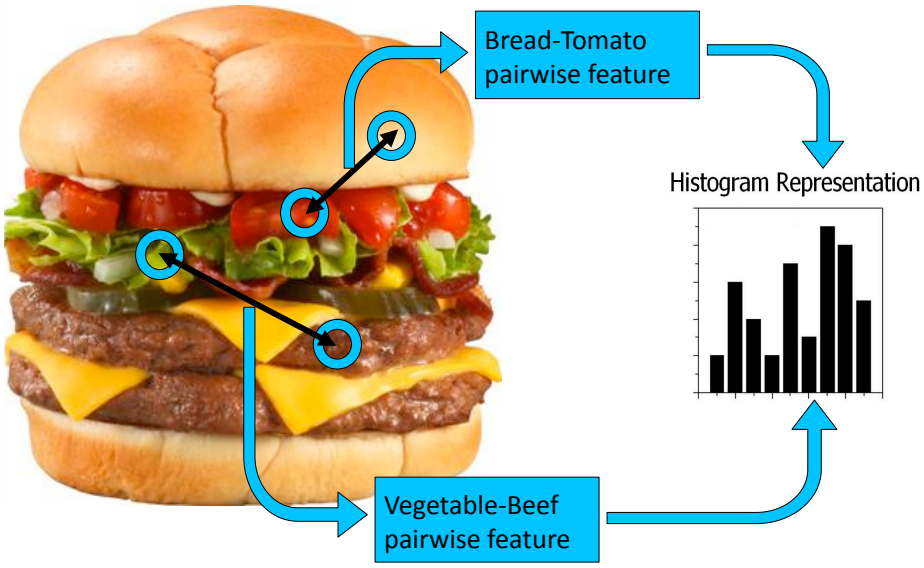


Figure 4.1: Exploiting spatial relationships between ingredients using pairwise feature statistics improves food recognition accuracy.

distinct ingredients, not all of which may be visible in a given image due to occlusion or variations in assembly. In particular, we cannot rely on the presence of distinctive low-level features nor on reliable edges between ingredients. Second, although perfect accuracy is not required at the ingredient level, the proposed method does require the pixel-level segmentation to generate distributions over ingredients from very small local patches. Finally, there can be significant intra-class variation in the observed sizes and spatial relationships between ingredients, requiring us to build representations that can cope with this appearance variability in a principled manner.

Our approach is illustrated in Fig. 5.2 and can be summarized as follows. First, we assign a soft label (distribution over ingredients) to each pixel in the image using a semantic texton forest [105]. For instance, the distribution corresponding to a pixel in a red patch could have a high likelihood for “tomato” and low for “cheese” while another pixel might have high values for “bread” and “cheese” but not “tomato”. Next, we construct a multi-dimensional histogram feature where each bin corresponds to a pair of ingredient labels

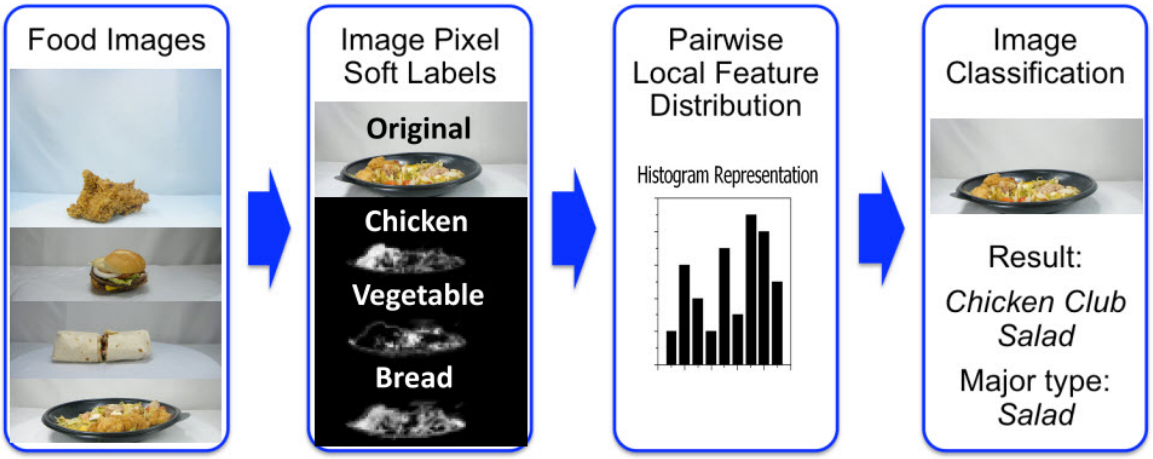


Figure 4.2: Framework of proposed approach: (1) Each pixel in the food image is assigned a vector representing the probability with which the pixel belongs to each of nine food ingredient categories, using STF [105]. (2) The image pixels and their soft labels are used to extract statistics of pairwise local features, to form a multi-dimensional histogram. (3) This histogram is passed into a multi-class SVM to classify the given image.

and discretized geometric relationships between two pixels. For example, the likelihood of observing a pair of "bread" and "tomato" labels that are 20–30 pixels apart at an angle of 40–50 degrees. Since we use soft labels, each observation from a pair of pixels contributes probability mass to a number of bins. Thus, the histogram aggregated from many samples of pixel pairs in the image serves as a representation of the spatial distribution of ingredients for that image. Finally, we treat this histogram as a feature vector in a discriminative classifier to recognize the food item. Our representation is not specific to food and could also be suitable for recognizing objects or scenes that are composed of visually distinctive "ingredients" arranged in predictable spatial configurations.

4.2 Related work

There has been relatively little work on the problem of food recognition. Shroff *et al.* [106] proposed a wearable computing system to recognize food for calorie monitoring. Bolle *et al.* [11]

developed an automatic produce ID system called VeggieVision to help with the produce checkout process. Yang *et al.* [21] introduced the PFID dataset for food recognition along with benchmarks using two baseline algorithms: color histogram and bag of SIFT [73] features. Russo *et al.* [98] monitored the production of fast food using video. Wu and Yang [126] analyzed eating videos to recognize food items and estimate caloric intake.

Of particular interest to food recognition are methods that explicitly address deformable objects. Shape context [4] picks n pixels from the contours of a shape, obtains $n-1$ vectors by connecting a pixel to the others, and uses these vectors as a description of shape at the pixel. Felzenszwalb proposes techniques [37] to represent a deformable shape using triangulated polygons. Leordeanu *et al.* [67] proposed an approach that recognizes category using pairwise interaction of simple features. A recent work [52] learns a mean shape of the object class based on the thin plate spline parametrization.

These approaches work well for many problems in object recognition, but two issues make them hard to apply to foods. First, these approaches all require detecting meaningful feature points, such as edges, contours, key points or landmarks. But precise features like these are typically not available in images of food. Second, shape similarity in food images is hard to exploit with these approaches, since the shape of real foods is often quite amorphous.

To overcome these challenges, our approach does not rely on the detection of features like edge or keypoints. Instead, we use local, statistical features defined over randomly selected pairs of pixels. The statistical distribution of pairwise local features effectively captures important shape characteristics and spatial relationships between food ingredients, thereby facilitating more accurate recognition.

4.3 Pairwise local feature distribution (PFD)

In this section, we give the definition of our image representation — the statistics of pairwise local features, namely the pairwise feature distribution (PFD). Using this representation, each image can be represented by a multi-dimensional histogram.

4.3.1 Soft labeling of pixels

Before obtaining the statistics of local features, we classify all image pixels into several categories based on the appearance of the image patch around the pixels. The local features will later be gathered separately based on different pixels categories. We use nine handpicked categories representing eight common fast food ingredients: beef, chicken, pork, bread, vegetable, tomato/tomato sauce, cheese/butter, egg/other, plus a category for the background.

Rather than labeling every pixel as a member of a single category, we use soft labeling to create a vector of probabilities corresponding to the likelihood that the given pixel belongs to each of the nine categories. Soft labeling defers commitment about the identity of a pixel, allowing for uncertainty about the true label for a pixel, and for the fuzzy distinction between ingredient categories.

Many methods could be used for pixel classification. We chose to employ the Semantic Texton Forest (STF) [105]. STF is a method for image categorization and segmentation that generates soft labels for a pixel based on local, low-level characteristics (such as the colors of nearby pixels). This is achieved using ensembles of decision trees that are each trained on subsets of manually-segmented images. A pixel is classified by descending each tree from root to leaf based on low-level features computed around the given pixel. The leaves contain probabilistic class distributions (over ingredients) that our algorithm uses to compute its soft labels. We chose to use STF, because it incorporates human knowledge in the form of manually labeled training images, and because it has proven effective for soft labeling in other challenging domains like the PASCAL Visual Object Classes (VOC) task.

With STF, each pixel is assigned a vector of K probabilities, where K is the number of pixel categories. $K = 9$ in our case. Fig. 4.3 is a visualization of the soft labeling of an image.

4.3.2 Global Ingredient Representation (GIR)

The simplest way to use the soft pixel labels produced by STF to represent a food image is to create a single one-dimensional histogram with 8 bins representing how frequently each of the

8 food ingredient categories appear in the image. To create an overall histogram representing the distribution of ingredients for the image, we sum up the soft labels for the 8 food ingredients for all the non-background pixels in an image. Then we normalize the histogram by the number of non-background pixels in the image. This global ingredient representation (GIR) is intuitive and easy to compute, but it does not capture the spatial relationships between ingredients that are important for distinguishing one food from another. It will serve as a useful benchmark for comparison with the more sophisticated, pairwise local features described below.

4.3.3 Pairwise Features

We attempt to capture the spatial relationship between pixels of different food ingredients using pairwise local features. Fig. 4.4 is an explanation of the four pairwise features we have explored.

Pairwise distance reflects the distance between two pixels in an image. Similar to the feature descriptors in the Shape Context method [4], the pairwise distance feature is defined to be greater when the two pixels in the pair are close to each other. We employ Sturges' formula [112], $k = \log_2\lceil n+1 \rceil$ and use the log of the distance between the pair of pixels P_1 and P_2.

$$D(P_1, P_2) = \log\lceil |P_1, P_2| + 1 \rceil \tag{4.1}$$

Pairwise orientation, $O(P_1, P_2)$, is defined as the angle of the line connecting the pixel pair. It ranges in $[0^o, 360^o)$, and positive is counter clockwise.

Pairwise midpoint, $M(P_1, P_2)$, is a discrete feature defined as the soft label of the pixel at the midpoint of the line connecting the two pixels. When this feature is added to the multi-dimensional histogram, the midpoint pixel's soft label (distribution over ingredient types) determines the weight of its contribution to the relevant bins.

Like the pairwise midpoint feature, the between-pair feature captures information about the pixel labels lying on the line connecting the two pixels in the pair. But rather than picking only the soft label of the midpoint, the between-pair feature incorporates all pixel soft labels along the line connecting the pair of pixels. So the feature for each pixel pair

has t discrete values, t being the number of pixels that exist along the line between a pair of pixels. We use $B(P_1, P_2) = \{T_i | i = 1, 2, ..., t\}$ to represent the feature set for pixels P_1 and P_2.

In addition to these individual features, we also explore joint features that are composites of the above features, such as a joint feature of distance and orientation, $DO(P_1, P_2)$, and a joint feature of orientation and midpoint, $OM(P_1, P_2)$, as shown in Fig. 4.5.

4.3.4 Histogram representation for pairwise feature distribution

Computing the complete set of pairwise features in an image can be computationally expensive since, for an image with M pixels, we would need to consider $\binom{M}{2}$ pairs. It suffices to consider $\binom{M}{2}$ rather than M^2 pairs because our pairwise relationships are symmetric; we convert directed orientation to a symmetric form by averaging counts in both directions. Nonetheless, $\binom{M}{2}$ can still be prohibitively large for high-resolution images.

Thus, for computational efficiency, we randomly sample N ($N = 1000$) pixels from the non-background portion of the image and estimate pairwise statistics using these $\binom{N}{2}$ pixel pairs. We use a set $\mathcal{P}$ to represent the N pixels we randomly pick from an image: $\mathcal{P} = \{P_1, P_2, ..., P_N\}$. The soft labels of the N pixels are represented as $\mathcal{S} = \{S_{ik} | i = 1, 2, ..., N;\ k = 1, 2, ..., 8\}$. We calculate pairwise local features ($D(P_i, P_j)$, $O(P_i, P_j)$, $M(P_i, P_j)$ and $B(P_i, P_j)$) for all pixels, and accumulate their values into a distribution. The distribution is weighed by the soft labels of the two pixels. Specifically, this weight is the product of the probabilities that a pixel is assigned to the ingredient label corresponding to the given bin.

We use a multi-dimensional histogram to represent the distribution of pairwise local features, as detailed in Fig. 4.6. The first two dimensions of the histogram are the ingredient labels. The other dimensions are the weighted pairwise features calculated for the given pair of pixels.

The first and second dimensions of the above multi-dimension histograms have 8 bins, representing the eight pixel categories (excluding the background). The other dimensions have either 12 bins for pairwise distance or orientation, or 8 bins for midpoint or between-

pair category.

4.3.5 Histogram normalization

We normalize the distribution of certain pairwise features to achieve invariance to transformations such as rotation and scaling. We compute the mode of pairwise distance or orientation, and shift the corresponding dimension of the multi-dimension histogram so that it is centered on the mode. Since we apply a logarithm to the distance feature, changing the scale of an image becomes equivalent to shifting its histogram in the distance dimension, resulting in scale invariance. Similarly, centering the orientation histogram at its mode value results in rotation invariance. Since we only measure relative distances between pixels, our representation is also translation invariant.

For joint features, we normalize the joint histogram by centering it at the mode value of the marginal distributions of each of its continuous dimensions.

4.3.6 Classification with local feature distributions

Using Pairwise Feature Distribution or Global Ingredient Representation, each image is represented as a multi-dimension histogram. We employ a Support Vector Machine (SVM) for classification using a χ^2 kernel [99].

The symmetrized approximation of χ^2 is defined as:

$$d_{\chi^2}(x, y) = \sum_i \frac{(x_i - y_i)^2}{x_i + y_i}, \tag{4.2}$$

where x and y represent the PFD histograms of two images, and x_i and y_i are two bins of the histograms.

The χ^2 kernel is defined as:

$$k_{\chi^2}(x, y) = e^{-d_{\chi^2}(x,y)}. \tag{4.3}$$

With this pre-computed kernel matrix, we apply the SVM to classify images into multiple food categories. In our implementation, we use the libSVM package [18].

4.4 Experimental Methodology

In this section, we briefly review our dataset, baseline approaches, and detail the implementation of the pixel-level segmentation using which we generate our ingredient soft labels. Experimental results are given in Section 4.5.

4.4.1 Dataset

We evaluated our work on the recently-released Pittsburgh Food Image Dataset (PFID) [21] and compared our proposed approach against their two baseline methods. The PFID dataset is a collection of fast food images and videos from 13 chain restaurants acquired under lab and realistic settings. Our experiments focused on the set of 61 categories of specific food items (e.g., McDonald's Big Mac) with masked background. Each food category contains three different instances of the food (bought on different days from different branches of the restaurant chain), and six images from six viewpoints (60 degrees apart) of each food instance.

We followed the experimental protocol proposed by Chen *et al.* [21] and performed 3-fold cross-validation for our experiments, using the 12 images from two instances for training and the 6 images from the third for testing. We repeated this procedure three times, with a different instance serving as the test set and average the results. The protocol ensures that no image of any given food item ever appears in both the training and test sets, and guarantees that food items were acquired from different restaurants on different days.

4.4.2 Baseline approaches

We use the two standard baseline algorithms specified by PFID: color histogram + SVM and bag of SIFT [73] features + SVM, as briefly described below.

Due in part to its simplicity, the color histogram method has been popular in object recognition for more than a decade. We employed a standard RGB 3-dimensional histogram with four quantization levels per color band. Each pixel in the image is mapped to its closest cell in the histogram to generate a $4^3 = 64$ dimensional representation for each image. Then we used a multi-class support vector machine (SVM) implementation [18] for classification.

Several studies (e.g., [66]) have demonstrated the merits of using a bag of SIFT features. The basic idea of this approach is to represent each image as a histogram of occurrence frequencies defined over a discrete vocabulary of features (pre-generated using k-means clustering) and then to use the histogram as a multi-dimensional vector in an SVM.

To compare against these baseline approaches, we conducted experiments using the six pairwise local features and the GIR feature proposed in Section 4.3. We describe the experimental details below.

4.4.3 Pre-processing with STF

The initial step in generating the six pairwise local features is to obtain pixel-wise soft labels for the images. We used Semantic Texton Forests (STF) [105] to generate these as follows. First, we trained STF using 16 manually-segmented food images, two examples of which are shown in Fig. 4.7(b). We found 16 training images to be sufficient to cover the appearance of the food ingredients in the PFID dataset. While more data could potentially generate better STF ingredient labels, using a small training set shows that our algorithm can operate with a small amount of relatively noisy training data.

After training, we employed STF to soft label all $61 \times 6 \times 3$ food images in the dataset, creating a 9-element probability vector for each pixel, representing the likelihood of it belonging to each of the eight food ingredient types or the background.

The output of the STF is far from a precise parsing of an image. Fortunately, our method does not require a perfect segmentation of food ingredients nor precise soft labeling results.

4.5 Results

This section summarizes our results, both at the individual food item level (61 categories) and at the major food type level (7 broad categories).

4.5.1 Classification accuracy on the 61 categories

Fig. 4.8 summarizes the classification accuracy results on the 61 categories for the two baselines (color histogram, bag of SIFT), the global ingredient representation (GIR), and

the six pairwise local features.

The chance recognition rate for each of the 61 categories is below 2% (1/61). The two baselines (color histogram and bag of SIFT features) reached only about 10%. The GIR global ingredient histogram method achieved 19%, and the local pairwise features methods range from 19% to 28%. The latter, achieved with the OM feature, is more than twice the accuracy of the PFID algorithms [21] and nearly 20$\times$ chance.

Fig. 4.9 shows the confusion matrices for four approaches: color histogram, bag of SIFT features, GIR, and pairwise feature OM. The darker the diagonal line, the more effective an approach, because it has a higher probability of classifying food of a given category as itself. Clearly, there is a more salient diagonal line in the matrix of the OM feature than the matrices for the other three approaches, which indicates that a greater number of food items are correctly classified as themselves using joint feature OM. In both of the baseline approaches, we can see straight vertical lines in their confusion matrices, showing that certain food categories are frequently (and incorrectly) chosen as the classification result; GIR and OM do not exhibit this problem.

There is a straightforward explanation for why the combination of orientation and midpoint is the higher-order feature that gives the best accuracy. This pair of features is able to leverage the vertically-layered structure of many fast foods. This vertical layering is particularly characteristic of burgers and sandwiches where meat and vegetables are situated between two (or more) horizontally oriented slices of bread. The orientation feature relates only two endpoints and therefore by itself cannot capture the three-way relationships like bread-meat-bread or bread-vegetable-bread that are characteristic of burgers and sandwiches. The midpoint feature captures three-way relationships, but without the orientation constraint, the statistics of the midpoint feature will be dominated by uniform triples like bread-bread-bread and meat-meat-meat created from horizontal samples, which form the majority. By combining the orientation and midpoint features, the algorithm is able to discover and leverage the vertical triplets that are key to accurately discriminating burgers and sandwiches from other food classes (e.g., pizza), as well as distinguishing between individual burgers and sandwiches (e.g., a Big Mac vs. a Quarter Pounder).

4.5.2 Classification accuracy into 7 major food types

The relatively low accuracy of even the best method described above is due in part to the fact that many foods with similar appearances and similar ingredients are assigned to different categories in the PFID food database, as shown in Fig. 4.10. Such cases are challenging, even for humans, to distinguish.

In order to match more closely to the way people categorize foods, we organized the 61 PFID food categories into seven major groups — sandwiches, salads/sides, chicken, breads/pastries, donuts, bagels, and tacos. Then, we used the baseline methods and our approach to classify images in the dataset into these categories. Fig. 4.11 shows the classification results.

The rankings between algorithms observed in the previous experiment continued to hold true in this broader categorization task, with OM achieving nearly 80% accuracy. More interestingly, the category-level confusion matrices (Fig. 4.12) provide some insights into the classification errors. We see that sandwiches are often confused with foods whose major ingredients include bread, such as donuts and bagels.

4.6 Conclusion

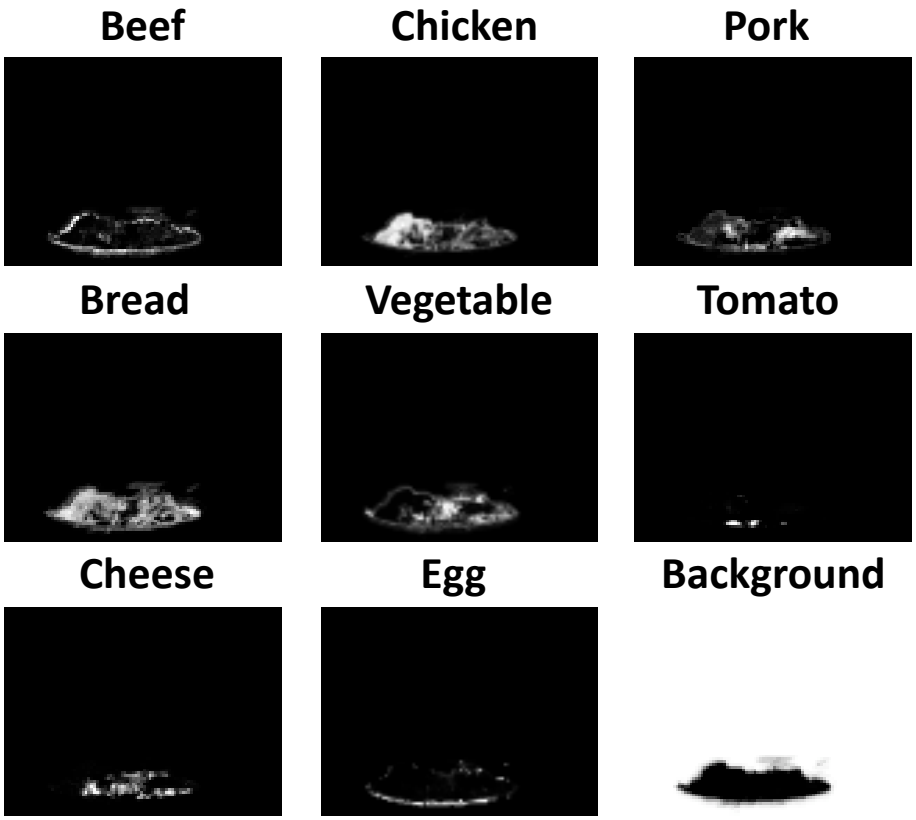
Food recognition is a new but growing area of exploration. Although many of the standard techniques developed for object recognition are ill-suited to this problem, we argue that exploiting the spatial characteristics of food, in combination with statistical methods for pixel-level image labeling will enable us to develop practical systems for food recognition.

Our experiments on a recent publicly-released dataset of fast food images demonstrated that our proposed method significantly outperforms the baseline bag-of-features models based on SIFT or color histograms, particularly when we augment pixel-level features with shape statistics computed on pairwise features.

In future work, we plan to extend our method to: (1) perform food detection and spatial localization in addition to whole-image recognition, (2) handle cluttered images containing several foods and non-food items, (3) develop practical food recognition applications, and (4) explore how the proposed method generalizes to other recognition domains. Preliminary

experiments indicate that pairwise statistics of STF features in conjunction with Gist [82] perform significantly better than Gist+STF+SVM alone on scene recognition tasks.

(a)

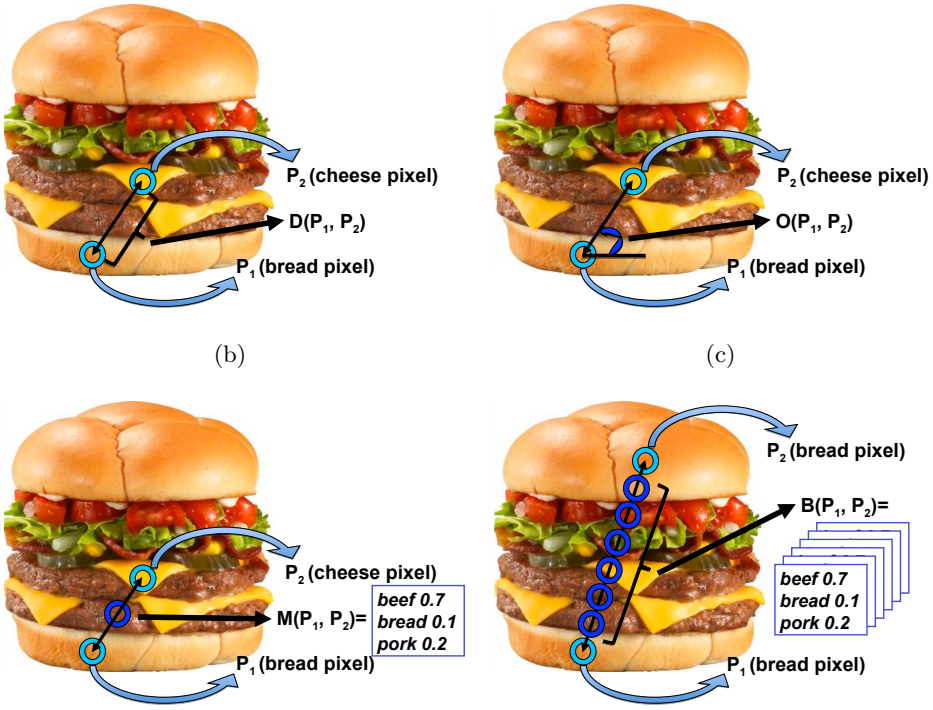


(b)

Figure 4.3: Visualization of soft labeling: (a) original food image. In (b), each of the nine images represents the probability that a given pixel belongs to that category. The brighter the pixel is, the larger its probability.

Pairwise Feature	Feature Description	Expression	Fig.
Distance	The distance between two pixels P_1 and P_2	$D(P_1, P_2)$	(b)
Orientation	The orientation of the line between two pixels P_1 and P_2	$O(P_1, P_2)$	(c)
Midpoint category	The type of the pixel in the middle of two pixels P_1 and P_2	$M(P_1, P_2)$	(d)
Between-pair category	The type of all pixels between two pixels P_1 and P_2	$B(P_1, P_2)$	(e)

(a)



(b) (c)

(d) (e)

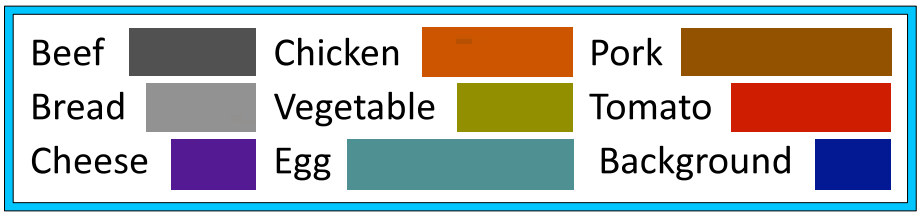
Figure 4.4: Four pairwise local features. In the table in (a), column 1 is the name of the feature; column 2 is a description of the feature; column 3 shows the expression of the feature in the following text; column 4 refers to their illustration figures. (b) (c) (d) (e) are illustrations of these pairwise local features.

Joint Pairwise Feature	Feature Description	Expression
Distance and Orientation	The conjunction of distance and orientation feature between pixels P_1 and P_2	$DO(P_1, P_2)$
Orientation and Midpoint category	The conjunction of orientation and midpoint category feature between pixels P_1 and P_2	$OM(P_1, P_2)$

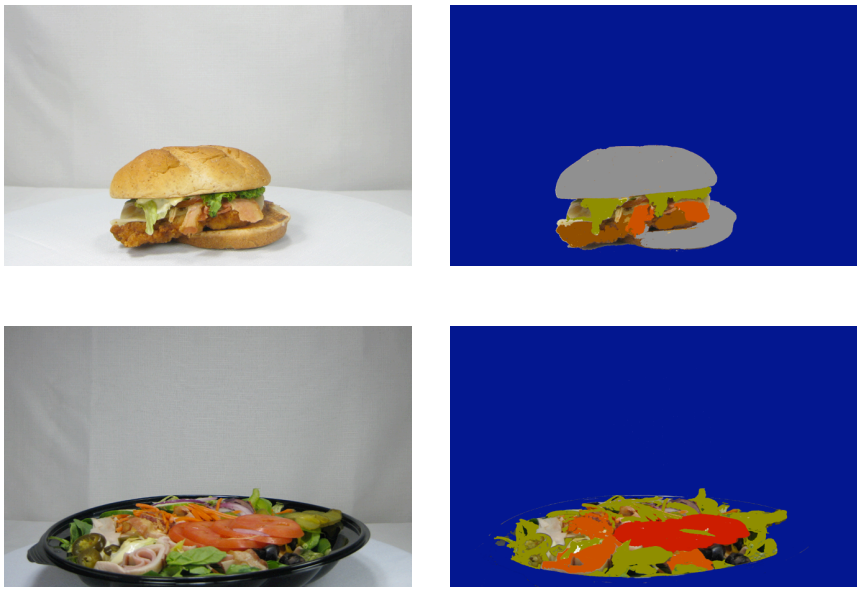
Figure 4.5: Joint pairwise features

Histogram Type	Histogram Description	Size
Distance	label × label × distance	8×8×12
Orientation	label × label × orientation	8×8×12
Midpoint category	label × label × midpoint	8×8×8
Between-pair category	label × label × between-pair	8×8×8
Distance and Orientation	label × label × distance × orientation	8×8×12×12
Orientation and Midpoint category	label × label × orientation × midpoint	8×8×12×8

Figure 4.6: Histogram representation of PFD



(a)



(b)

Figure 4.7: (a) shows nine colors that are used to label training images for STF. Each color represents one of the food ingredient types or the background. (b) shows two samples of manually labeled images for STF: use the nine labels in (a) to color all pixels of the 16 training images for STF. (Best viewed in color.)

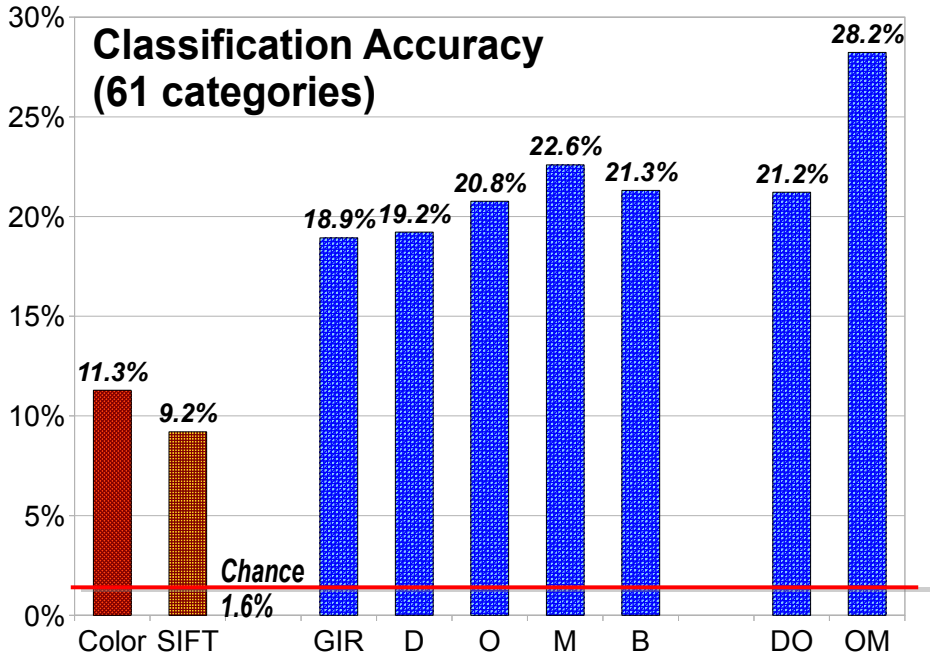


Figure 4.8: Classification accuracy for 61 categories

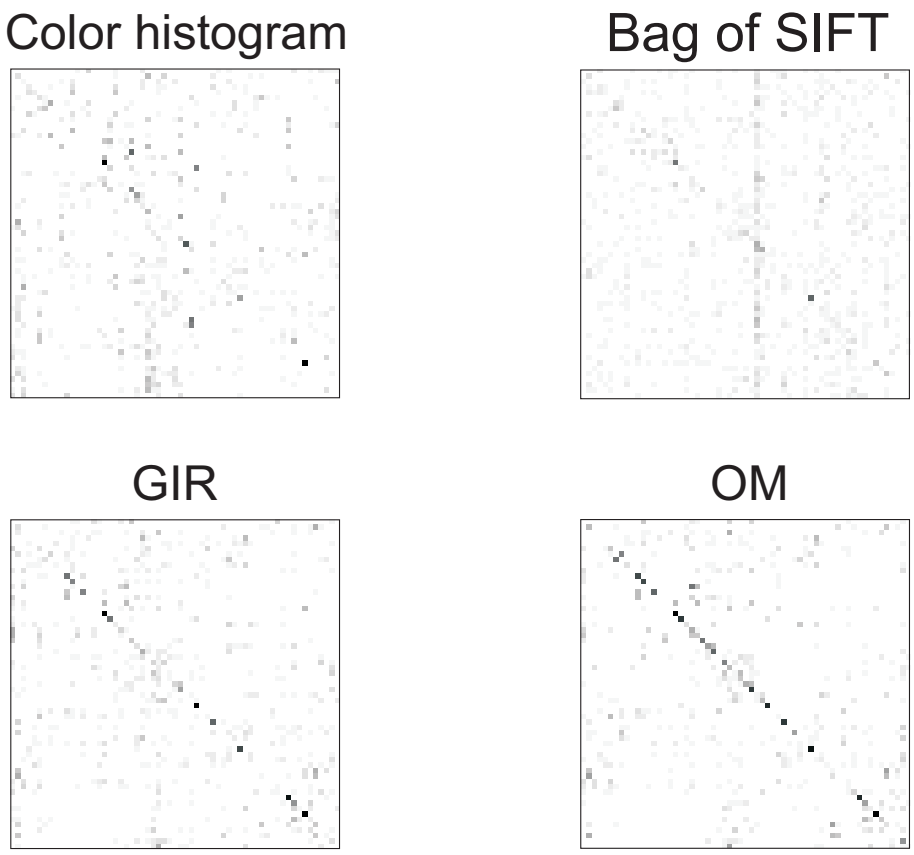


Figure 4.9: Confusion matrix comparison for 61 categories. Rows represent the 61 categories of food, and columns represent the ground truth categories.

Figure 4.10: Similar food items in PFID: these two foods are semantically and visually similar, but are distinct food items and thus appear in different PFID categories. Such subtle distinctions make fast food recognition inherently challenging for vision algorithms. (left) Arby's Roast Turkey and Swiss; (right) Arby's Roast Turkey Ranch Sandwich.

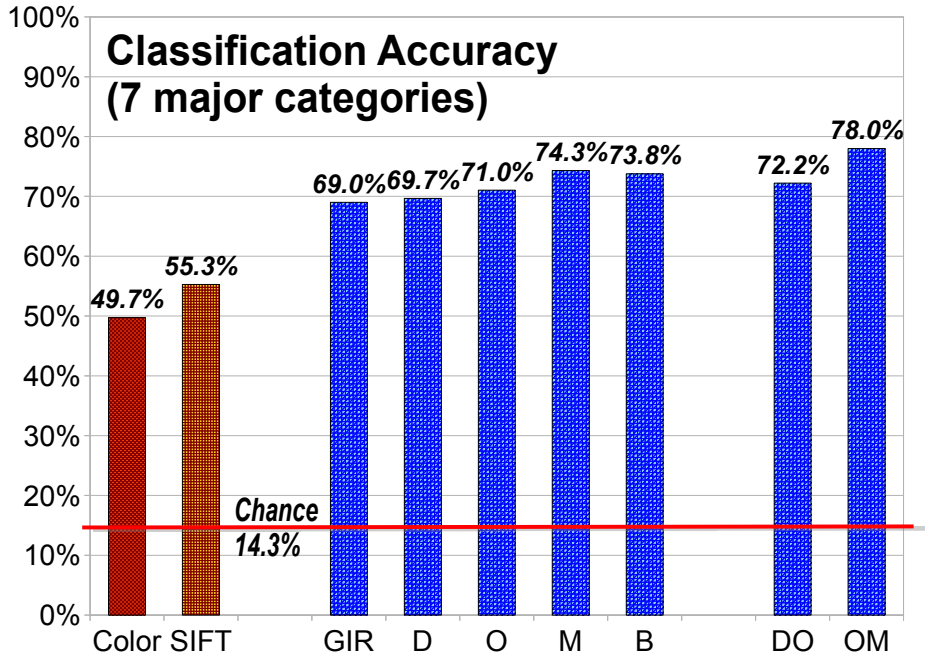


Figure 4.11: Classification accuracy for 7 major types

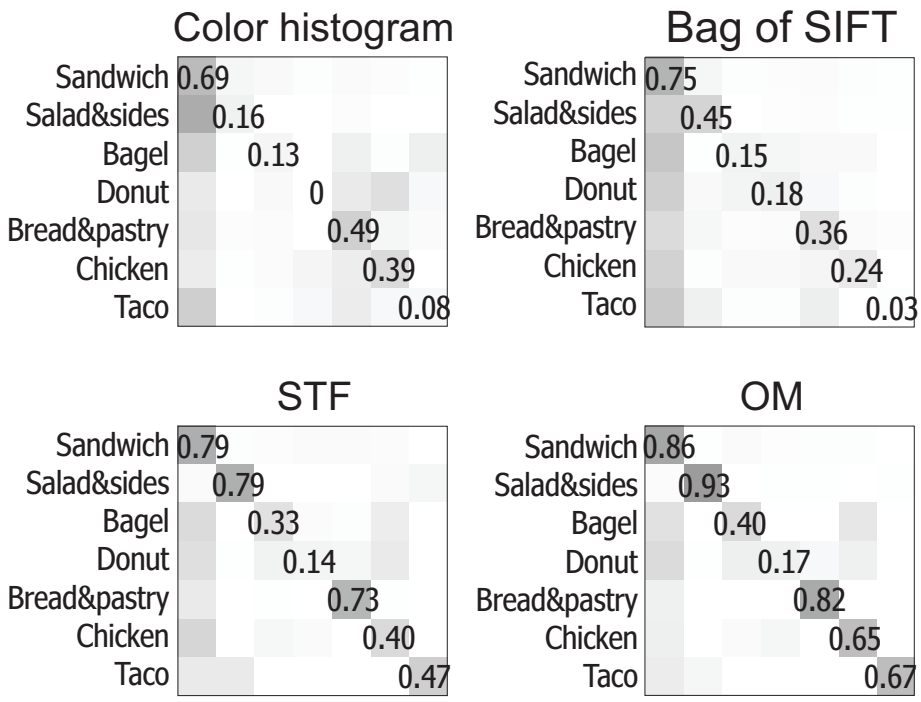


Figure 4.12: Confusion matrix comparison for 7 major categories. Rows represent the major 7 food categories, and columns represent the ground truth major categories.

Chapter 5

TEMPLATE LEARNING IN FINE-GRAINED OBJECT RECOGNITION

5.1 Introduction

In the previous chapter, the focus was a design of image representation for food recognition – a specific fine-grained object recognition task. In this chapter, the focus will be on the general fine-grained object recognition problem, and our work on learning image representation for the general recognition tasks will be described.

Cognitive research study has suggested that basic-level recognition is based on comparing the shape of the objects and their parts, whereas subordinate-level recognition is based on comparing appearance details of certain object parts [32]. This suggests that finding the right correspondence of object parts is of great help in recognizing fine-grained differences. For basic-level recognition tasks, spatial pyramid matching [62] is a popular choice that aligns object parts by partitioning the whole image into multiple-level spatial cells. However, spatial pyramid matching may not be the best choice for fine-grained object recognition, since falsely aligned object parts can lead to inaccurate comparisons, as shown in Figure 5.1.

This work is intended to alleviate the limitations of spatial pyramid matching. Our key observation is that in a fine-grained task, different object categories share commonality in their shape or structure, and the alignment of object parts can be greatly improved by discovering such common shape patterns. For example, bird images from different species may have similar shape patterns in their beaks, tails, feet or bodies. The commonality usually is a part of the global shape, and can be observed in bird images across different species and in different poses. This motivates us to decompose a fine-grained object recognition problem into two sub-problems: 1) aligning image regions that contain the same object part and 2) extracting image features within the aligned image regions. To this end, we propose a

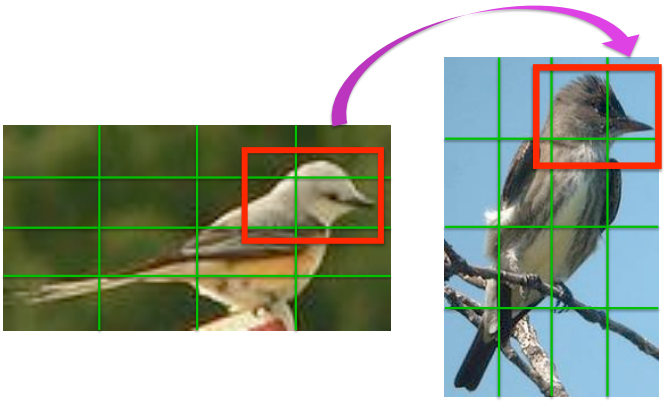


Figure 5.1: Region alignment by spatial pyramid matching and our approach. Spatial pyramid matching partitions the whole image into regions, without considering visual appearance. A 4×4 partition leads to misalignment of parts of the birds while a coarse partition (i.e. 2×2) includes irrelevant features. Our approach aims to align the image regions containing the same object parts (red squares).

template model to align object parts. In our model, a template represents a shape pattern, and the relationship between two shape patterns is captured by the relationship between templates, which reflects the probability of their co-occurrence in the same image. This model is learned using an alternative algorithm, which iterates between detecting aligned image regions, and updating the template model. Kernel descriptor features [6, 9] are then extracted from image regions aligned by the learned templates.

Our model is evaluated on two benchmark datasets: the Caltech-UCSD Bird200 and the Stanford Dogs. Our experimental results suggest that the proposed template model is capable of detecting image regions that correspond to meaningful object parts, and our template-based algorithm outperforms the state-of-the-art fine-grained object recognition algorithms in terms of accuracy.

5.2 Unsupervised Learning of Template Model

This section provides an overview of our fine-grained object recognition approach. We discuss the framework of our template based object recognition, describe our template

model, and propose an alternative algorithm for learning model parameters.

5.2.1 Template-Based Fine-Grained Object Recognition

Over the last decades, computer vision researchers have done a lot of work in designing effective and efficient patch-level features for object recognition [73, 63, 127, 121, 14, 22, 132]. SIFT is one of the most successful features, allowing an image or object to be represented as a bag of SIFT features [73]. However, the ability of such methods is somewhat limited. Patch-level features are descriptive only within spatial context constraints. For example, a cross shape can be a symbol for the Red Cross, Christian religion, or Swiss Army products, depending on the larger spatial context of where it is detected. It is hard to interpret the meaning of patch-level features without considering such spatial contexts. This is even more important for a fine-grained recognition task since common features can be shared by instances from both the same and different object classes. Spatial pyramid models [62, 14, 15] align sub-images/parts that are spatially close by partitioning the whole images into multi-level spatial cells. However the alignments produced by the spatial pyramid are not necessarily correct, since no displacements are allowed in the model (Figure 5.1).

Here, we use a template model to find correctly-aligned regions from different images, so that comparisons between them are more meaningful. A template represents one type of common shape pattern of an object part, while an object part can be represented by several different templates. Certain shape patterns of two object parts (for instance, a head facing the left and a tail pointing to the right) can frequently be observed in the same image. Our template model is designed to capture both properties of templates and their relationships among templates. Model parameters are learned from a collection of unlabeled images in an unsupervised manner. See sections 5.2.2 and 5.2.3 for more details.

Once the templates and their relationship are learned, the fine-grained differences can be aligned based on these quantities. The framework of our template based fine-grained object recognition is illustrated in Figure 5.2. In the learning stage, Algorithm 1 is used to find the templates. In the recognition stage (from left to right in Figure 5.2), aligned image regions are extracted from each image using our template detection algorithm. Color-

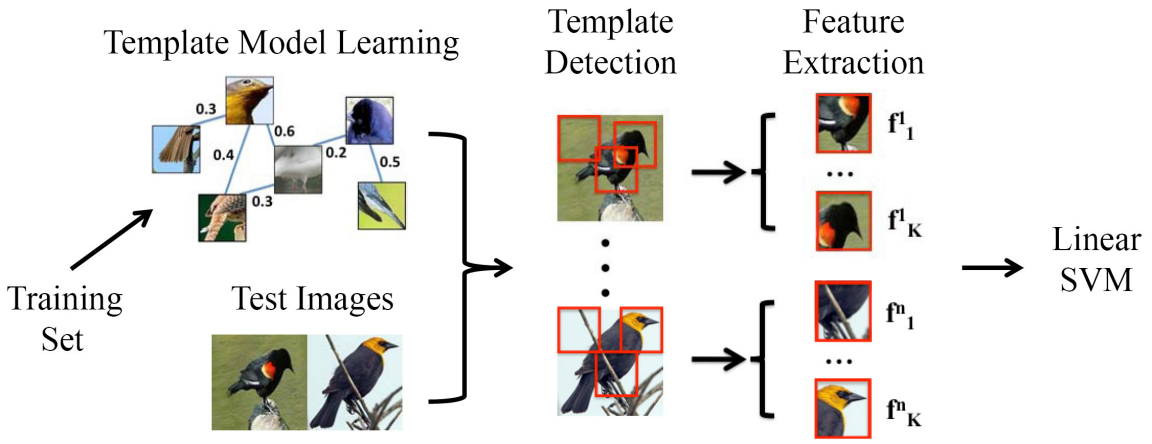


Figure 5.2: The framework for fine-grained recognition: the recognition pipeline goes from left to right. In the training stage, a template model is learned from training images using Algorithm 1. In the recognition stage, the learned templates are applied to each test image, resulting in aligned image regions. Then image-level feature vectors are extracted as the concatenation of features of all aligned regions. Finally, a linear SVM is used for recognition.

based, normalized color-based, gradient-based, and LBP-based kernel descriptors followed by EMK [9] are then applied to generate feature representations for each region. The image-level feature is the concatenation of feature representations of all detected regions from the corresponding image. Finally, a linear SVM [18] is used for recognition.

5.2.2 Template Model

We start by defining a template model that represents the common shape patterns of object parts and their relationships. A template is an entity that contains features that will match image features for region detection. Let $M = \{\mathbb{T}, \mathcal{W}\}$ be a model that contains a group of templates $\mathbb{T} = \{T_1, T_2, ..., T_K\}$ and their co-occurrence relationships $\mathcal{W} = \{w_{11}\, w_{12} ..., w_{KK}\}$, where K is the number of templates, and w_{ij} is between 0 and 1. When $w_{ij} = 0$, the two templates T_i and T_j have no co-occurrence relationship.

When a template model is matched to a given image, not all templates within the model

are necessarily used. This is because different templates can be associated with the same object part, but one part only occurs at most once in an image. Our model captures this intuition by making the templates inactive that do not match images very well. To model appearance properties of templates and their relationships, the score function between templates and a given image I^t should capture three aspects: 1) fitness, which computes the similarity of the selected templates and image regions that are most highly matched to them; 2) co-occurrence, which encourages selecting templates that have a high chance of co-occurring in the same image; and 3) diversity, which gives preference to having the selected templates match separated image regions.

Fitness: We define a matching score $s^f(T_i, x_i^I)$ to measure the similarity between a template T_i and an image region at location x_i^I in image I

$$s^f(T_i, x_i^I) = 1 - \|T_i - R(x_i^I)\|^2 \quad s.t. \;\; |x_i^I - \overline{x}_i^I| \leq \alpha \tag{5.1}$$

where $R(x_i^I)$ represents the features of the sub-image in I centered at the location x_i^I; $\overline{x}_i^I$ is an initial location associated with the template T_i and α is an upper bound of location variation. Both x_i^I and $\overline{x}_i^I$ are measured by their relative location in image I. If $|x_i^I - \overline{x}_i^I| > \alpha$, the location is too far from the initial location, and the score is set to zero.

The features describing $R(x_i^I)$ should be able to capture common properties of object parts. Since the same type of part from different objects usually share similar shapes, we introduce edge kernel descriptors to capture this common statistic. We first run the Berkeley edge detector [75] to compute the edge map of an image, and then treat it as a grayscale image and extract color kernel descriptors [6] over it. Using these descriptors, we compute $s^f(T_i, x_i^I)$; the higher its value, the better is the match.

Summing up the matching score $s^f(T_i, x_i^I)$ for all templates that are used for image I, we obtain a fitness term

$$S^f(\mathbb{T}, \mathcal{X}^I, \mathcal{V}^I) = \sum_{i=1}^{K} v_i^I s^f(T_i, x_i^I) \tag{5.2}$$

where $\mathcal{V}^I = \{v_1^I, ..., v_K^I\}$ represents the selected template subset for image I, $v_i^I = 1$ means that the template T_i is used for image I, and $\mathcal{X}^I = \{x_1^I, ..., x_K^I\}$ represents the locations of all templates on image I. The more templates that are used, the higher the score is.

Co-occurrence: With the observation that certain shape patterns of two or more object parts coexist frequently in the same image, it is desired that templates that have a high chance of co-occurring are selected together. For a given image, the co-occurrence term is used to encourage selecting two templates together, which have a large relation parameter w_{ij}. Meanwhile, a L_1 penalty term is used to ensure sparsity of the template relation.

$$S^c(\mathcal{W}, \mathcal{V}^I) = \sum_{i=1}^{K}\sum_{j=1}^{K} v_i^I v_j^I w_{ij} - \lambda \sum_{i=1}^{K}\sum_{j=1}^{K} |w_{ij}| \quad s.t. \;\; 0 \leq w_{ij} \leq 1 \tag{5.3}$$

Diversity: This term is used to enforce spatial relationship constraints on the locations of selected templates. In particular, their locations should not be too close to each other, because we want the learned templates to be diverse, so that they can cover a large range of image shape patterns. So this term sums up a location penalty on the templates,

$$S^d(\mathcal{X}^I, \mathcal{V}^I) = -\sum_{i=1}^{K}\sum_{j=1}^{K} v_i^I v_j^I d(x_i^I, x_j^I) \tag{5.4}$$

where $d(x_i^I, x_j^I)$ is the location penalty function. We have $d(x_i^I, x_j^I) = \infty$ if $|x_i^I - x_j^I| < \beta$ and $d(x_i^I, x_j^I) = 0$, otherwise. β is a distance parameter.

Summing up all three terms defined above: fitness, co-occurrence and diversity terms for all images in the image set D, we have the overall score function between templates and images

$$S(\mathbb{T}, \mathcal{W}, \mathcal{X}, \mathcal{V}, D) = \sum_{I \in D} (S^f(\mathbb{T}, \mathcal{X}^I, \mathcal{V}^I) + S^c(\mathcal{W}, \mathcal{V}^I) + S^d(\mathcal{X}^I, \mathcal{V}^I)) \tag{5.5}$$

where $\mathcal{V} = \{\mathcal{V}^1, \mathcal{V}^2, ..., \mathcal{V}^{|D|}\}$ are template indicators, $\mathcal{X} = \{\mathcal{X}^1, \mathcal{X}^2, ..., \mathcal{X}^{|D|}\}$ are template locations, and $|D|$ is the number of images in the set D. The templates and their relations are learned by maximizing the score function $S(\mathbb{T}, \mathcal{W}, \mathcal{X}, \mathcal{V}, D)$ on an image collection D.

5.2.3 Template Learning

We use an alternating algorithm to optimize (5.5). The proposed algorithm iterates among three steps:

- updating $\mathcal{X}, \mathcal{V}$ (template detection),

- updating $\mathbb{T}$ (template feature learning), and

- updating $\mathcal{W}$ (template relation learning).

Template detection: Given a template model $\{\mathbb{T}, \mathcal{W}\}$, the goal of template detection is to find the template subset $\mathcal{V}$ and their locations $\mathcal{X}$ for all images to maximize equation (5.5). The second term in S^c in equation (5.3) is a constant given $\mathcal{W}$. So maximizing (5.5) is reduced to maximizing the following term for each image I respectively:

$$\max_{\mathcal{X}^I, \mathcal{V}^I} \sum_{i=1}^{K} v_i^I s^f(T_i, x_i^I) + \sum_{i=1}^{K} \sum_{j=1}^{K} v_i^I v_j^I (w_{ij} - d(x_i^I, x_j^I)) \tag{5.6}$$

The above optimization problem is NP-hard, so a greedy approach is used: the algorithm starts with an empty set, first calculates the scores for all templates, and then selects the template with the largest score. Fixing the locations of all previously selected templates, the next template and its location can be chosen in a similar manner. The procedure is repeated until the object function (5.6) no longer increases.

Template feature learning: The goal of template feature learning is to optimize the templates $\mathbb{T}$ given the relation parameters $\mathcal{W}$ and current template detection results $\mathcal{V}$, $\mathcal{X}$. When maximizing (5.5), S^d and S^c are all constants given $\mathcal{V}$, $\mathcal{X}$ and $\mathcal{W}$. The optimal template T_i can be found by maximizing

$$\max_{T_i} \sum_{I \in D} v_i^I (1 - \|T_i - R(x_i^I)\|^2) \tag{5.7}$$

which can be solved by the closed form equation

$$T_i = \sum_{I \in D} v_i^I R(x_i^I) / \sum_{I \in D} v_i^I \tag{5.8}$$

Eq (5.8) means that the template T_i is updated by the average of the features of all sub-images in D that are detected by the i-th template.

Template relation learning: The goal here is to assign values to the relation parameters $\mathcal{W}$ given all other parameters ($\mathbb{T}$, $\mathcal{V}$ and $\mathcal{X}$) for the purpose of maximizing equation (5.5). Since only $\mathcal{W}$ are optimization parameters, S^f and S^d are both constants. Optimizing (5.5)

Algorithm 1 Template Model Learning

input Image set D, maximum iteration $maxiter$, threshold ϵ

output Template model $M = \{\mathbb{T}, \mathcal{W}\}$.

Initialize $\{T_1, T_2, ..., T_K\}$ with training data; initialize $w_{ij} = 0$; $iter = 0$

for $iter < maxiter$ **do**

update $\mathcal{X}^I$, $\mathcal{V}^I$ for all $I \in D$ based on equation (5.6)

update $\mathbb{T}$ by: $T_i = \sum_{I \in D} v_i^I R(x_i^I) / \sum_{I \in D} v_i^I$ (as in (5.8))

update $\mathcal{W}$ to optimize (5.9)

if $\sum_i |_\Delta T_i| < \epsilon$ **then**

break

end if

$iter \leftarrow iter + 1$

end for

is simplified as maximizing

$$\max_{\mathcal{W}} \sum_{i=1}^{K} \sum_{j=1}^{K} w_{ij} \sum_{I \in D} v_i^I v_j^I - \lambda |D| \sum_{i=1}^{K} \sum_{j=1}^{K} |w_{ij}| \tag{5.9}$$

A L_1 regularization solver [100] is used for optimizing this formula.

The whole learning procedure is summarized in Algorithm 1. The algorithm starts by initiating K templates with various sizes and initial locations that are are evenly spaced in an image. In each iteration, template detection, template feature learning, and template relation learning are alternated. The iteration continues until the total change of template $\{T_i\}_{i=1}^K$ is smaller than a threshold ϵ.

5.3 Experiments

We tested our model on two publicly available datasets: Caltech-UCSD Bird-200 and Stanford Dog. These two datasets are the standard benchmarks to evaluate fine-grained object recognition algorithms. Our experiments suggest that the proposed template model is able to detect the meaningful parts and outperforms the previous work in terms of accuracy.

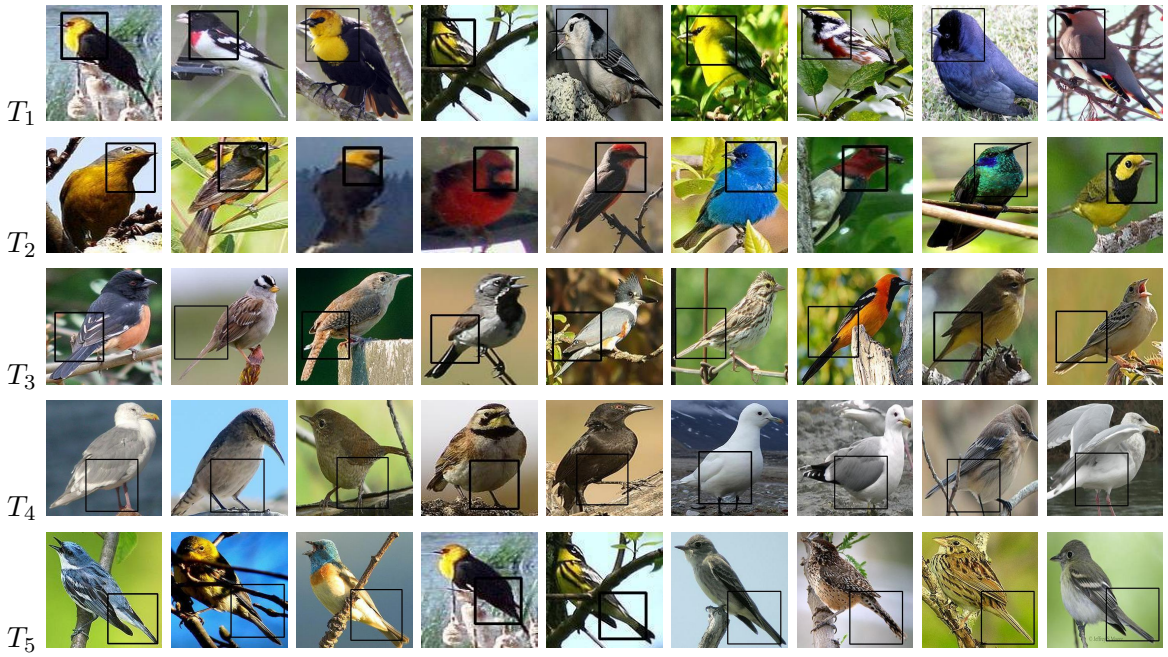


Figure 5.3: Object parts (black squares) detected by learned templates. Each line shows the parts found by one learned template. The sub-image within the black square has the highest matching score for a given image. Meaningful parts are successfully detected such as heads, backs and tails.

5.3.1 Features and Settings

We use kernel descriptors (KDES) to capture low-level image statistics: color, shape and texture [6]. In particular, we use four types of kernel descriptors: color-based, normalized color-based, gradient-based, and local-binary-pattern-based descriptors[1]. Color and normalized color kernel descriptors are extracted over RGB images, and gradient and shape kernel descriptors are extracted over gray scale images transformed from the original RGB images. Following the standard parameter setting, we compute kernel descriptors on 16×16 image patches over dense regular grids with spacing of 8 pixels. For template relation learn-

[1] `http://www.cs.washington.edu/ai/Mobile_Robotics/projects/kdes/`

ing, we use a publicly available L_1 regularization solver [2]. All images are resized to be no larger than 300×300 with the height/width ratio preserved.

To learn the template model, we use 34 templates with different sizes. The template size is measured by its ratio to the original image size, such as 1/2 or 1/3. Our model has 9 templates with size 1/2 and 25 with size 1/3. The initial locations of templates with each template size are evenly spaced grid points in an image. We observe that the learning algorithm converges very fast and usually becomes stable around $15 \sim 20$ iterations.The sparsity level parameter λ is set to be 0.1. Other model parameters are $\alpha = 24$ and $\beta = 32$ pixels. These parameters are optimized by performing cross validation on training set of the Bird dataset. The same parameter setting is then applied to the Dog dataset. On each region detected by templates, we compute template-level features using EMK features [9]. After obtaining these template-level features, we train a linear support vector machine for fine-grained object recognition.

Notice that there is a slight difference between template detection in the learning phase and in the recognition phase. In the learning phase, only a subset of templates are detected for each image. This is because not all templates can be observed in all images, and each image usually contains only a subset of all possible templates. But in the recognition phase, all templates are selected for detection in order to avoid missing features.

5.3.2 Bird Recognition

Caltech-UCSD Bird-200 [122] is a commonly used dataset for evaluating fine-grained object recognition algorithms. The dataset contains 6033 images from 200 bird species in North America. In each image, the bounding box of a bird is given. Following the standard setting [16], 15 images from each species are used for training and the rest for testing.

Template learning: Figure 5.3 visualizes the rectangles/parts detected by the learned templates. The feature in each template consists of a vector of real numbers. As can be seen, the learned templates successfully find the meaningful parts of birds, though the appearances of these parts are very different. For examples, the head parts detected by T_1

[2] `http://www.di.ens.fr/~mschmidt/Software/L1General.html`

Table 5.1: The table in the left show the classification accuracies (%) obtained by templates with different sizes and numbers on a subset of a full dataset. The accuracy is improved with an increasing template number at the beginning, and become saturated when enough templates are used. With the best template number choices, the combination of templates with different sizes are tested. The table in the right shows the accuracies (%) achieved by different combinations on the full dataset. The combination of 9 templates with size 1/2 and 25 templates with size 1/3 performs best (selected using the training set).

Acc	1	4	9	16	25	36
T^1	**46.1**	46.1	46.1	46.1	46.1	46.1
$T^{\frac{1}{2}}$	39.6	46.8	**50.7**	50.7	48.9	47.5
$T^{\frac{1}{3}}$	33.2	42.9	41.8	43.9	**44.3**	44.3
$T^{\frac{1}{4}}$	32.1	37.5	**40.4**	40	**40.4**	40

Combination	Acc
$9T^{\frac{1}{2}}$	27.1
$T^1 + 9T^{\frac{1}{2}}$	27.4
$9T^{\frac{1}{2}} + 25T^{\frac{1}{3}}$	28.2
$T^1 + 9T^{\frac{1}{2}} + 25T^{\frac{1}{3}} + 25T^{\frac{1}{4}}$	28.2

Table 5.2: Effect of sparsity parameter λ: that the best accuracy is achieved when $\lambda = 0.1$.

λ	0	0.001	0.005	0.01	0.05	0.1	0.5	1
Accur	48.57	48.93	49.28	49.29	49.64	50.7	50	48.57

have quite different colors and textures, suggesting the robustness of the proposed template model.

Sparsity parameter λ: We tested different values for the sparsity level parameter λ on a subset of 20 categories (from the training set) for efficiency. If $\lambda = 0$, there is no penalty on the relation parameters $\mathcal{W}$, thus all weights w_{ij} are set to 1 when the template model is learned. If $\lambda \geq 1$, the penalty on the relation parameters is large enough that all w_{ij} are set to 0 after learning. In both these cases, the template models are equivalent to a simplified model without the co-occurrence term in (5.3). If λ is a number between 0 and 1, test results in Table 5.2 show that the best accuracy is achieved when $\lambda = 0.1$.

Template size and number choices: We tested the effect of the number and size of

the templates on the recognition accuracy. All the results are obtained on a subset of 20 categories for efficiency. When the template size is 1, the accuracy is the same with an arbitrary template number, because template detection will return the same results. For templates whose size is smaller than 1, the results obtained with different numbers of templates are shown in Table 5.1 *left*. Based on these results, we selected a template number for each template size for further experiments: one template with size 1, 9 templates with size 1/2, 25 templates with size 1/3, and 25 templates with size 1/4. The results obtained by the combinations of templates with different sizes (each with its optimal template number) on the full dataset are shown in Table 5.1 *right*. The highest accuracy is achieved by the combination of 9 templates with size 1/2 and 25 templates with size 1/3. Our further experiments suggest that adding more templates only slightly improves the recognition accuracy.

Running time: Our algorithm is efficient. With a non-optimized version of the algorithm, in the training stage, each iteration takes $2 \sim 3$ minutes to update. In the test stage, it takes $3 \sim 5$ seconds to process each image, including template detection, feature extraction and classification. This is fast enough for an on-line recognition task.

Comparisons with the state-of-the-art algorithms: We compared our model with four recently published algorithms for fine-grained object recognition: multiple kernel learning [16], random forest [131], LLC [131], and multi-cue [56] in Table 5.3. We also compared our model to KDES [6] with spatial pyramid, a strong baseline in terms of accuracy.

We observe that KDES with spatial pyramid works well on this dataset, and the proposed template model works even better. The template model achieves 28.2% accuracy, about 6 percents higher than the best results reported in the previous work and about 2 percents higher than KDES with spatial pyramid. This accuracy is comparable with the recently proposed pose pooling approach [136] where labeled parts are used to train and test models; this is not required for our template model.

Table 5.3: Comparisons on Caltech-UCSD Bird-200. Our template model is compared to the recently proposed fine-grained recognition algorithms. The performance is measured in terms of accuracy.

MKL [16]	LLC [131]	Rand-forest [131]	Multi-cue [56]	KDES [6]	This work
19.0	18.0	19.2	22.4	26.4	**28.2**

Table 5.4: Comparisons on Stanford Dog Dataset. Our approach is compared to a baseline algorithm in [57] and KDES with spatial pyramid. We give the results of the proposed template model with two types of templates: edge templates and texture templates.

Methods	SIFT [57]	KDES [6]	Edge templates	Texture templates
Accuracy(%)	22.0	36.0	**38.0**	36.9

5.3.3 Dog Recognition

The Stanford Dogs dataset is another benchmark dataset for fine-grained image categorization recently introduced in [57]. The dataset contains $20,580$ images of 120 breeds of dogs from around the world. Bounding boxes of dogs are provided for all images in the dataset. This dataset is a good complement to the Caltech-UCSD Bird200 due to more images in each category: around 200 images per class versus 30 images per class in Bird200. Following the standard setting [57], 100 images from each category are used for training and the rest for testing.

Comparisons with the state-of-art algorithms: We compared our model with a baseline algorithm [57] and KDES with spatial pyramid on this dataset. For the dog datasets, we also tried using the local binary pattern KDES to learn templates instead of the edge KDES due to the relative consistent textures in dog images. Our experiments show that the template learning with the edge KDES works better than that with the local binary pattern KDES, suggesting that the edge information is a stable cue to learn templates. Notice that the accuracy achieved by our template model is 16 percent higher than the best published

results so far.

5.4 Conclusion

In this chapter, we have proposed a template model for fine-grained object recognition. The template model learns a group of templates by jointly considering fitness, co-occurrence and diversity between the templates and images, and the learned templates are used to align image regions that contain the same object parts. Our experiments show that the proposed template model has achieved higher accuracy than the state-of-the-art fine-grained object recognition algorithms on the two standard benchmarks: Caltech-UCSD Bird-200 and Standford Dogs. In the future, we plan to learn the features that are suitable for detecting object parts and incorporate the geometric information into the template relationships.

Chapter 6

CONCLUSION AND FUTURE WORK

Fine-grained image classification, including fine-grained object recognition and fine-grained shape analysis, covers a variety of important and useful problems in computer vision. This research focused on the exploration of different problems in this family. We proposed algorithms and pipelines for fine-grained image classification, including both specific tasks like food recognition and skull shape analysis for craniosynostosis, and general fine-grained object recognition tasks.

The key contribution of the pipelines we proposed are the modules on feature engineering. Our proposed algorithms cover different aspects of feature engineering, including feature design, feature learning and feature selection. The performances of each of the algorithms proposed are demonstrated on one or more benchmark datasets, on which they outperformed the state-of-the-art algorithms.

Although multiple aspects of feature engineering are covered in this thesis, there are still many to be explored in the future work. For fine-grained object recognition, a unified framework that performs both object detection and recognition is lacking. In the human visual system, the processes of detection and recognition cannot be clearly separated. A human usually alternates between the two processes to jointly optimize the outcome of both detection and recognition. A pipeline that performs both object detection and recognition is needed for a computer vision system to work in a real world application. Therefore, it is necessary to develop algorithms that are able to jointly address these two problems. For example in the template model we proposed, the matching of templates and image regions can also be used to locate the position of objects in an image. So this model can be improved for such a unified system.

Furthermore, the current recognition accuracy is not good enough to be used in real world applications. Multiple aspects of the current pipeline can be improved for better

performance. First, from the feature learning perspective, richer features can be applied in this pipeline, such as features learned by Deep Belief Network, which is the new state-of-the-art feature learning method. Second, the objective of our template learning model is straightforward and considers some basic matching criteria in template fitness and template relationships. More comprehensive criteria can be explored for improvement of region alignment. For example when considering template relationships, besides template co-occurrence, the spatial relations between two templates have important information as well, such as the relative location of two templates including distance and orientation. Including this information would change the relation term of the objective of template matching, thus a new learning algorithm for optimizing the model will also be needed. Third, more data needs to be collected and tested. Having more data may affect the performance of the pipeline either in a good way, because of the larger size of the training data, or in a bad way, because new data can potentially introduce new problems.

For fine-grained shape analysis problems, especially in the domain of medical imaging, it is worthwhile to generalize the current framework to the analysis of other parts, for example the face and the feet, as well as to other illness types. It is also worthwhile to use feature learning algorithms for 3D shape features in fine-grained shape analysis.

From a more general perspective, the feature engineering modules covered in this thesis are focused on capturing the spatial information in images. The 3D shape analysis project is aimed at selecting features corresponding to important regions on the 3D shape that distinguish different class types; in the food recognition project, the new image representation proposed is aimed at capturing spatial relationship of multiple food ingredients inside the food images; the template model proposed in the previous chapter is targeted at more accurate alignment of meaningful image regions that contain corresponding object parts. Spatial information in images is one of the most important cues to their meaning, as well as one of the most difficult signals to represent. In order to capture such spatial information, the design and learning of mid-level features, which can bridge the gap between low-level cues and image-level representations, can be key. The templates we proposed are an attempt at such mid-level features. More exploration needs to be done in this direction.

BIBLIOGRAPHY

[1] M. Aharon, M. Elad, and A. Bruckstein. K-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation. *IEEE Transactions on Signal Processing*, 54(11):4311–4322, 2006.

[2] A. Alpher, , and J. P. N. Fotheringham-Smythe. Frobnication revisited. *Journal of Foo*, 13(1):234–778, 2003.

[3] Authors. The frobnicatable foo filter, 2012. BMVC12 submission ID 324. Supplied as additional material `bmvc12.pdf`.

[4] S. Belongie, J. Malik, and J. Puzicha. Shape context: a new descriptor for shape matching and object recognition. In *NIPS*, 2000.

[5] L. Bo, K. Lai, X. Ren, and D. Fox. Object recognition with hierarchical kernel descriptors. *IEEE International Conference on Computer Vision and Pattern Recognition*, 2011.

[6] L. Bo, X. Ren, and D. Fox. Kernel Descriptors for Visual Recognition. *NIPS*, 2010.

[7] L. Bo, X. Ren, and D. Fox. Kernel descriptors for visual recognition. *Advances in Neural Information Processing Systems*, December 2010.

[8] L. Bo, X. Ren, and D. Fox. Depth Kernel Descriptors for Object Recognition. *IROS*, September 2011.

[9] L. Bo and C. Sminchisescu. Efficient match kernel between sets of features for visual recognition. *NIPS*, 2009.

[10] O. Boiman, E. Shechtman, and M. Irani. In Defense of Nearest-Neighbor based Image Classification. *CVPR*, 2008.

[11] R. Bolle, J. Connell, N. Haas, R. Mohan, and G. Taubin. VeggieVision: A produce recognition system. In *WACV*, 1996.

[12] L. Bourdev and J. Malik. Poselets: body partddetectors trained using 3d human pose annotations. *ICCV*, 2009.

[13] Y. Boureau, F. Bach, Y. LeCun, and J. Ponce. Learning Mid-level Features for Recognition. In *CVPR*, 2010.

[14] Y. Boureau, F. Bach, Y. LeCun, and J Ponce. Learning mid-level features for recognition. *CVPR*, 2010.

[15] Y. Boureau and J. Ponce. A theoretical analysis of feature pooling in visual recognition. *ICML*, 2010.

[16] S. Branson, C. Wah, B. Babenko, F. Schroff, P. Welinder, P. Perona, and S. Belongie. Visual recognition with humans in the loop. *ECCV*, Sept. 2010.

[17] M. Burl, M. Weber, and P. Perona. A probabilistic approach to object recognition using local photometry and global geometry. In *ECCV*, 1998.

[18] C. Chang and C. Lin. *LIBSVM: a library for support vector machines*, 2001.

[19] O. Chapelle, P. Haffner, and V. N. Vapnik. Support vector machines for histogram-based image classification. *IEEE Transactions on Neural Networks*, 10(5), 1999.

[20] K. Chatfield, V. Lempitsky, A. Vedaldi, and A. Zisserman. The devil is in the details: an evaluation of recent feature encoding methods. *British Machine Vision Conference*, 2011.

[21] M. Chen, K. Dhingra, W. Wu, L. Yang, R. Sukthankar, and J. Yang. PFID: Pittsburgh fast-food image dataset. *Proceedings of International Conference on Image Processing*, 2009.

[22] A. Coates and A. Ng. The importance of encoding versus training with sparse coding and vector quantization. *ICML*, 2011.

[23] M. Cuturi and J. Vert. Semigroup Kernels on Finite Sets. *NIPS*, 2004.

[24] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. *CVPR*, 2005.

[25] N. Dalal and B. Triggs. Histograms of Oriented Gradients for Human Detection. *CVPR*, 2005.

[26] G. Davis, S. Mallat, and M. Avellaneda. Adaptive Greedy Approximations. *Constructive Approximation*, 13(1):57–98, 1997.

[27] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Fei-fei. ImageNet: A Large-Scale Hierarchical Image Database. *CVPR*, 2009.

[28] Sven Dickinson. The evolution of object categorization and the challenge of image abstraction. In *Object Categorization: Computer and Human Vision Perspectives*. Cambridge University Press, 2009.

[29] W. Dong, Z. Wang, M. Charikar, and K. Li. Efficiently Matching Sets of Features with Random Histograms. *ACMMM*, 2008.

[30] K. Duan, D. Parikh, D. Crandall, and K. Grauman. Discovering localized attributes for fine-grained recognition. *CVPR*, 2012.

[31] R. Fan, K. Chang, C. Hsieh, X. Wang, and C. Lin. LIBLINEAR: A Library for Large Linear Classification. *JMLR*, 9:1871–1874, 2008.

[32] R. Farrell, O. Oza, N. Zhang, V. Morariu, T. Darrell, and L. Davis. Birdlets: subordinate categorization using volumetric primitives and pose-normalized appearance. *ICCV*, 2011.

[33] L. Fei-Fei, R. Fergus, and A. Torralba. Recognizing and learning object categories. *IJCV short course*, 2007.

[34] L. Fei-Fei and P. Perona. A bayesian hierarchical model for learning natural scene categories. *CVPR*, 2005.

[35] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9), 2010.

[36] P. Felzenszwalb, D. McAllester, and D. Ramanan. A Discriminatively Trained, Multiscale, Deformable Part Model. *CVPR*, 2008.

[37] Pedro F. Felzenszwalb. Representation and detection of deformable shapes. *PAMI*, 27(2), 2005.

[38] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. *CVPR*, 2, 2003.

[39] A. Frome, Y. Singer, and J. Malik. Image Retrieval and Classification Using Local Distance Functions. *NIPS*, 2006.

[40] T. Gärtner, P. A. Flach, A. Kowalczyk, and A. Smola. Multi-Instance Kernels. *ICML*, 2002.

[41] P. Gehler and S. Nowozin. On Feature Combination for Multiclass Object Classification. *ICCV*, 2009.

[42] K. Grauman and T. Darrell. The pyramid match kernel: Discriminative classification with sets of image features. *ICCV*, 2005.

[43] K. Grauman and T. Darrell. The Pyramid Match Kernel: Efficient Learning with Sets of Features. *JMLR*, 8:725–760, 2007.

[44] H. Gray and H Carter. *Gray's Anatomy*. Sterling Publishing, 2000.

[45] Chunhui Gu, Joseph J. Lim, Pablo Arbelaez, and Jitendra Malik. Recognition using regions. *CVPR*, 2009.

[46] D. Haussler. Convolution Kernels on Discrete Structures. Technical report, 1999.

[47] A. B. Hillel and D. Weinshall. Subordinate class recognition using relational object models. *NIPS*, 2006.

[48] G. Hinton, S. Osindero, and Y. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006.

[49] Zhang J., Marszalek M., Lazebnik S., and Schmid C. Local Features and Kernels for Classification of Texture and Object Categories: A Comprehensive Study. *IJCV*, 73(2):213–238, 2007.

[50] P. Jain, B. Kulis, and K. Grauman. Fast Image Search for Learned Metrics. *CVPR*, 2008.

[51] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun. What is the Best Multi-Stage Architecture for Object Recognition? *ICCV*, 2009.

[52] T. Jiang, F. Jurie, and C. Schmid. Learning shape prior models for object matching. In *CVPR*, 2009.

[53] F. Jurie and B. Triggs. Creating Efficient Codebooks for Visual Recognition. *ICCV*, 2005.

[54] K. Kavukcuoglu, M. Ranzato, R. Fergus, and Y. LeCun. Learning Invariant Features through Topographic Filter Maps. *CVPR*, 2009.

[55] K. Kavukcuoglu, P. Sermanet, Y. Boureau, K. Gregor, M. Mathieu, and Y. LeCun. Learning Convolutional Feature Hierarchies for Visual Recognition. 2010.

[56] F. Khan, J. van de Weijer, A. Bagdanov, and M. Vanrell. Portmanteau vocabularies for multi-cue image representations. *NIPS*, 2011.

[57] A. Khosla, N. Jayadevaprakash, B. Yao, and L. Fei-Fei. Novel dataset for fine-grained image categorization. *First Workshop on Fine-Grained Visual Categorization, CVPR*, 2011.

[58] K. Kim, M. Franz, and B. Schölkopf. Iterative Kernel Principal Component Analysis for Image Modeling. *IEEE PAMI*, 27(9):1351–1366, 2005.

[59] R. Kondor and T. Jebara. A Kernel Between Sets of Vectors. *ICML*, 2003.

[60] A. Krizhevsky. Learning Multiple Layers of Features from Tiny Images. Technical report, 2009.

[61] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning Realistic Human Actions from Movies. *CVPR*, 2008.

[62] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. *CVPR*, 2006.

[63] H. Lee, A. Battle, R. Raina, and A. Ng. Efficient sparse coding algorithms. *NIPS*, 2007.

[64] H. Lee, R. Grosse, R. Ranganath, and A. Ng. Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations. *ICML*, 2009.

[65] S.-I. Lee, H. Lee, P. Abbeel, and A. Ng. Efficient L1 regularized logistic regression. *Proceedings of the 21st National Conference on Artificial Intelligence*, 2006.

[66] B. Leibe, A. Leonardis, and B. Schiele. Combined object categorization and segmentation with an implicit shape model. *Proc. Workshop on Statistical Learning Computer Vision*, 2004.

[67] M. Leordeanu, M. Hebert, and R. Sukthankar. Beyond local appearance: Category recognition from pairwise interactions of simple features. In *CVPR*, 2007.

[68] F. Li, R. Fergus, and P. Perona. One-Shot Learning of Object Categories. *IEEE PAMI*, 2006.

[69] L. Li and L. Fei-Fei. What, Where and Who? Classifying Event by Scene and Object Recognition. *ICCV*, 2007.

[70] L. Li, H. Su, E. Xing, and L. Fei-Fei. Object Bank: A High-Level Image Representation for Scene Classification and Semantic Feature Sparsification. *NIPS*, 2010.

[71] H. Lin, S. Ruiz-Correa, R. Sze, M. Cunningham, M. Speltz, A. Hing, and L. Shapiro. Efficient symbolic signatures for classifying craniosynostosis skull deformities. *Workshop of ICCV*, 2005.

[72] J. Liu, S. Ji, and J. Ye. *SLEP: Sparse Learning with Efficient Projections.* Arizona State University, 2009.

[73] D. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2), 2004.

[74] S. Lyu. Mercer Kernels for Object Recognition with Local Features. *CVPR*, 2005.

[75] M. Maire, P. Arbelaez, C. Fowlkes, and J. Malik. Using contours to detect and localize junctions in natural images. *CVPR*, 2008.

[76] L. Meier, S. Geer, and P. Bhlmann. The group lasso for logistic regression. *J. R. Statist. Soc. B*, 70:53–71, 2008.

[77] K. Mikolajczyk and C. Schmid. A Performance Evaluation of Local Descriptors. *IEEE PAMI*, 27(10):1615–1630, 2005.

[78] F. Moosmann, B. Triggs, and F. Jurie. Fast Discriminative Visual Codebooks using Randomized Clustering Forests. *NIPS*, 2006.

[79] P. Moreno, P. Ho, and N. Vasconcelos. A Kullback-Leibler Divergence Based Kernel for SVM Classification in Multimedia Applications. *NIPS*, 2003.

[80] J. Mutch and D. Lowe. Multiclass Object Recognition with Sparse, Localized Features. *CVPR*, 2006.

[81] T. Ojala, M. Pietikäinen, and T. Mäenpää. Multiresolution Gray-Scale and Rotation Invariant Texture Classification with Local Binary Patterns. *IEEE PAMI*, 24(7):971–987, 2002.

[82] A. Oliva and A. Torralba. Modeling the shape of the scene: a holistic representation of the spatial envelope. *IJCV*, 42(3), 2001.

[83] A. Oliva and A. Torralba. Modeling the Shape of the Scene: A Holistic Representation of the Spatial Envelope. *IJCV*, 42(3):145–175, 2001.

[84] Aude Oliva and Antonio Torralba. Modeling the shape of the scene: a holistic representation of the spatial envelope. *IJCV*, 42(3), 2001.

[85] O. Parkhi, A. Vedaldi, A. Zisserman, and C Jawahar. Cats and dogs. *CVPR*, 2012.

[86] M. Parsana, S. Bhattacharya, C. Bhattacharyya, and K. Ramakrishnan. Kernels on Attributed Pointsets with Applications. *NIPS*, 2007.

[87] Y. Pati, R. Rezaiifar, and P. Krishnaprasad. Orthogonal Matching Pursuit: Recursive Function Approximation with Applications to Wavelet Decomposition. *The Twenty-Seventh Asilomar Conference on Signals, Systems and Computers*, pages 40–44, 1993.

[88] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Lost in Quantization: Improving Particular Object Retrieval in Large Scale Image Databases. *CVPR*, 2008.

[89] A. Quattoni and A. Torralba. Recognizing Indoor Scenes. *CVPR*, 2009.

[90] Ariadna Quattoni, Michael Collins, and Trevor Darrell. Conditional random fields for object recognition. *NIPS*, 2004.

[91] A. Rahimi and B. Recht. Random features for large-scale kernel machines. *NIPS*, 2007.

[92] A. Rahimi and B. Recht. Weighted Sums of Random Kitchen Sinks: Replacing minimization with randomization in learning. *NIPS*, 2008.

[93] M. Ranzato, Krizhevsky A., and G. Hinton. Factored 3-Way Restricted Boltzmann Machines for Modeling Natural Images. *AISTATS*, 2010.

[94] M. Ranzato and G. Hinton. Modeling Pixel Means and Covariances Using Factorized Third-Order Boltzmann Machines. *CVPR*, 2010.

[95] R. Rubinstein, A. Bruckstein, and M Elad. Dictionaries for Sparse Representation Modeling. *Proceedings of the IEEE*, 98(6):4311–4322, 2010.

[96] R. Rubinstein, M. Zibulevsky, and M. Elad. Efficient Implementation of the K-SVD Algorithm using Batch Orthogonal Matching Pursuit. Technical report, 2008.

[97] S. Ruiz-Correa, R. Sze, J. Starr, H. Lin, M. Speltz, M. Cunningham, and A. Hing. New scaphocephaly severity indices of sagittal craniosynostosis: A comparative study with cranial index quantifications. *Cleft Palate-Craniofacial Journal*, 2006.

[98] R. Russo, N. da Vitoria Lobo, and M. Shah. A computer vision system for monitoring production of fast food. In *Proceedings of Asian Conference on Computer Vision*, 2002.

[99] B. Schiele and J. L. Crowley. Object recognition using multidimensional receptive field histograms. In *ECCV*, 1996.

[100] M. Schmidt, G. Fung, and R. Rosales. Optimization methods for L_1-regularization. *UBC Technical Report*, 2009.

[101] B. Schölkopf, A. Smola, and K. Müller. Nonlinear Component Analysis as a Kernel Eigenvalue Problem. *Neural Computation*, 10:1299–1319, 1998.

[102] B. Schölkopf, A. Smola, and K. Müller. Nonlinear Component Analysis as a Kernel Eigenvalue Problem. *Neural Computation*, 10(5), 1998.

[103] S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal Estimated sub-GrAdient SOlver for SVM. *ICML*, pages 807–814, 2007.

[104] L. Shapiro, K. Wilamowska, I. Atmosukarto, J. Wu, C. Heike, M. Speltz, and M. Cunningham. Shape- based classification of 3d head data. *ICIAP*, 2009.

[105] J. Shotton, M. Johnson, and R. Cipolla. Semantic texton forests for image categorization and segmentation. *CVPR*, 2008.

[106] G. Shroff, A. Smailagic, and D. Siewiorek. Wearable context-aware food recognition for calorie monitoring. In *Proceedings of International Symposium on Wearable Computing*, 2008.

[107] J. Sivic and A. Zisserman. Video Google: A Text Retrieval Approach to Object Matching in Videos. *ICCV*, 2003.

[108] B. Slater, K. Lenton, M. Kwan, D. Gupta, D. Wan, and M. Longaker. Cranial sutures: a brief review. *Plastic and Reconstructive Surgery*, 121(4):170–178, 2008.

[109] S. Smale, L. Rosasco, J. Bouvrie, A. Caponnetto, and T. Poggio. Mathematics of the Neural Response. *Foundations of Computational Mathematics*, 10(1):67–91, 2010.

[110] Reynold Spector. Science and pseudoscience in adult nutrition research and practice. *Skeptical Inquirer*, 33, 2009.

[111] J. Starr, K. Kapp-Simon, Y. Cloonan, B. Collett, M. Cradock, L. Buono, M. Cunningham, and M. Speltz. Pre- and post-surgery neurodevelopment of infants with single-suture craniosynostosis: Comparison with controls. *Journal of Neurosurgery (Pediatrics)*, 107(2):103–110, 2007.

[112] H. A. Sturges. The choice of a class interval. *J. American Statistical Association*, 1926.

[113] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society*, 58(1):267–288, 1996.

[114] R Tibshirani, M Saunders, S Rosset, Y. Heights, J. Zhu, and K. Knight. Sparsity and smoothness via the fused lasso. *J. R. Statist. Soc. B*, 67:91–108, 2005.

[115] A. Torralba, R. Fergus, and W. Freeman. 80 Million Tiny Images: A Large Data Set for Nonparametric Object and Scene Recognition. *IEEE PAMI*, 30(11):1958–1970, 2008.

[116] J. van Gemert, J. Geusebroek, C. Veenman, and A. Smeulders. Kernel Codebooks for Scene Categorization. *ECCV*, 2008.

[117] M. Varma and D. Ray. Learning The Discriminative Power-Invariance Trade-Off. *ICCV*, 2007.

[118] A. Vedaldi and A. Zisserman. Efficient Additive Kernels via Explicit Feature Maps. *CVPR*, 2010.

[119] C. Wah, S. Branson, P. Perona, and S. Belongie. Interactive localization and recognition of fine-grained visual categories. *ICCV*, 2011.

[120] C. Wallraven, B. Caputo, and A. Graf. Recognition with Local Features: the Kernel Recipe. *ICCV*, 2003.

[121] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Guo. Locality-constrained linear coding for image classification. *CVPR*, 2010.

[122] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-ucsd birds 200. *Technical Report CNS-TR-201, Caltech*, 2010.

[123] C. Williams and M. Seeger. Using the Nyström Method to Speed Up Kernel Machines. *NIPS*, 2000.

[124] L. Wolf and A. Shashua. Learning over Sets using Kernel Principal Angles. *JMLR*, 4:913–931, 2003.

[125] J. Wu and J. Rehg. Beyond the Euclidean distance: Creating Effective Visual Codebooks using the Histogram Intersection Kernel. *ICCV*, 2002.

[126] W. Wu and J. Yang. Fast food recognition from videos of eating for calorie estimation. In *Proceedings of International Conference on Multimedia and Expo*, 2009.

[127] J. Yang, K. Yu, Y. Gong, and T. Huang. Linear spatial pyramid matching using sparse coding for image classification. *CVPR*, 2009.

[128] S. Yang, M. Chen, D. Pomerleau, and R. Sukthankar. Food recognition using statistics of pairwise local features. *CVPR*, 2010.

[129] S. Yang, L. Shapiro, M. Cunningham, M. Speltz, and S.-I. Lee. Classification and feature selection for craniosynostosis. *Proceeding BCB '11 Proceedings of the 2nd ACM Conference on Bioinformatics, Computational Biology and Biomedicine*, pages 340–344, 2011.

[130] B. Yao, G. Bradski, and L. Fei-Fei. A codebook-free and annotation-free approach for fine-grained image categorization. *CVPR*, 2012.

[131] B. Yao, A. Khosla, and L. Fei-Fei. Combining randomization and discrimination for fine-grained image categorization. *CVPR*, 2011.

[132] K. Yu, Y. Lin, and J. Lafferty. Learning image representations from the pixel level via hierarchical sparse coding. *CVPR*, 2011.

[133] K. Yu, W. Xu, and Y. Gong. Deep Learning with Kernel Regularization for Visual Recognition. *NIPS*, 2008.

[134] M. Zeiler, D. Krishnan, G. Taylor, and R. Fergus. Deconvolutional Networks. *CVPR*, 2010.

[135] H. Zhang, A. Berg, M. Maire, and J. Malik. SVM-KNN: Discriminative Nearest Neighbor Classification for Visual Category Recognition. *CVPR*, 2006.

[136] N. Zhang, R. Farrell, and T. Darrell. Pose pooling kernels for sub-category recognition. *CVPR*, 2012.

VITA

Shulin Yang is a PhD student in the Computer Science and Engineering Department at the University of Washington.