IMAGE CLASSIFICATION WITH DENSE SIFT SAMPLING: AN EXPLORATION OF OPTIMAL PARAMETERS

by

AARON J. CHAVEZ

B.S., Kansas State University, 2005
M.S., Kansas State University, 2008

AN ABSTRACT OF A DISSERTATION

submitted in partial fulfillment of the requirements for the degree

DOCTOR OF PHILOSOPHY

Department of Computer Science
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2012

Abstract

In this paper we evaluate a general form of image classification algorithm based on dense SIFT sampling. This algorithm is present in some form in most state-of-the-art classification systems. However, in this algorithm, numerous parameters must be tuned, and current research provides little insight into effective parameter tuning. We explore the relationship between various parameters and classification performance. Many of our results suggest that there are basic modifications which would improve state-of-the-art algorithms. Additionally, we develop two novel concepts, *sampling redundancy* and *semantic capacity*, to explain our data. These concepts provide additional insight into the limitations and potential improvements of state-of-the-art algorithms.

IMAGE CLASSIFICATION WITH DENSE SIFT SAMPLING: AN EXPLORATION OF OPTIMAL PARAMETERS

by

AARON J. CHAVEZ

B.S., Kansas State University, 2005
M.S., Kansas State University, 2008

A DISSERTATION

submitted in partial fulfillment of the requirements for the degree

DOCTOR OF PHILOSOPHY

Department of Computer Science
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2012

Approved by:

Major Professor
David A. Gustafson

Copyright

Abstract

In this paper we evaluate a general form of image classification algorithm based on dense SIFT sampling. This algorithm is present in some form in most state-of-the-art classification systems. However, in this algorithm, numerous parameters must be tuned, and current research provides little insight into effective parameter tuning. We explore the relationship between various parameters and classification performance. Many of our results suggest that there are basic modifications which would improve state-of-the-art algorithms. Additionally, we develop two novel concepts, *sampling redundancy* and *semantic capacity*, to explain our data. These concepts provide additional insight into the limitations and potential improvements of state-of-the-art algorithms.

Table of Contents

List of Figures

List of Tables

Dedication

I dedicate this paper first and foremost to my family: Megan, Mom, Dad, and Lexi. Thank you for providing tireless support and patience. I would not be here if you had not made me who I am. Mom, thank you for talking me into finishing.

I would also like to dedicate this paper to Dr. David Gustafson. Thank you for providing me with the exact amount of guidance to keep me on track, while allowing me to follow my own path.

Thanks also go out to all of my friends for their encouragement, but in particular I must thank Jon Hoag and Joshua Holmgren for taking an active interest in my work and providing stimulating discussion.

1 Introduction

How does one recognize a picture of a dog? A car? A face? When translating this question into an algorithm, the problem is reduced to binary classification. There is a mathematical difference between a picture that contains a dog and one that does not. Finding such differences is the goal of image classification.

While most state-of-the-art image classification algorithms share a set of basic components, there is little research exploring the optimal parameters for these basic components. Most of the work entails finding extensions or modifications to the basic algorithm. But, without a robust set of performance data from a basic algorithm, it is difficult to precisely assess the value of improvements. Furthermore, a lack of complete understanding of the basic algorithm makes it difficult to predict which lines of research are promising.

While even the simplest image classifier entails many design decisions, there are several elements shared by nearly all state-of-the-art approaches. These elements are as follows:

1. Dense low-level feature sampling,
2. Construction of a representative codebook of features,
3. Codebook matching via a bag-of-features approach,
4. Spatial pyramid matching for organization, and
5. Learning and classification via support vector machine.

First, low-level features must be extracted from the image. These features are gathered by looking at small windows in the image, and performing some low-level processing on the pixels to generate values that are more indicative of the structure (shape, texture, color, etc.) While many vision applications attempt to find a small number of interest points in an image (to meet time constraints), image classifiers frequently employ a dense sampling strategy, finding thousands of features at regular pixel intervals across the whole image. In this paradigm, no attempt is made to determine which features in the image are salient. While usually slower, this method guarantees that the salient features will be included, and gives superior results in practice.

Next, a codebook is constructed from some or all of the low-level features that have been extracted. A codebook contains a set of visual "words" that will be used to describe any image. Ideally, a codebook will have a sufficient number of words that are reasonably distinct.

The codebook is necessary because next, each image feature will be matched to one or more codebook features. Two features that have been matched to the same codebook feature(s) will be considered to be the same in the remaining steps. The benefit of this reduction is twofold. One, an intractable number of distinct possible features can be categorized into a manageable number of groups, based on the size of the codebook. Two, the generalization allows us to see the same codebook feature many times across many images, which is necessary for learning.

From these codebook matches, a "bag-of-features" is constructed for the image. The image is fully described by a vector consisting of the number of occurrences of each codebook feature in that image. This representation contains no information about where in the image those features were found, or any geometric relationship between these features.

An extension exists in the codebook phase that is not intrinsic to the approach, but is extremely important in practice. The codebook matching phase ignores the location of features in the image. To retain some of this information, a spatial pyramid classifies features into a few different groups based on where the features were found in the original image. The image vector then consists of several "bags-of-features", one for each location, concatenated with each other (including the "bag-of-features" that describes the image as a whole). The localization provided by the spatial pyramid is very coarse but still substantially improves the classifier.

The image has now been transformed from raw pixel data into a vector of values, each of which correlates more with certain image classes than others. If this process is repeated for all images in a set of training data, the resulting vectors can be passed into a support vector machine. From these vectors and some initialization parameters, the support vector machine learns a classifier that can predict the class of test images. The classifier works by evaluating the similarity of a test image to certain positive and negative examples in the training set.

1.1 Justification of Value

This paper will analyze basic image classifiers. It will explore the relationship between various parameters and classification performance. Specifically, it will analyze codebooks (construction and size), sampling strategies (density, scales used), and feature complexity.

Knowledge of these relationships will provide insight into the performance of a basic classifier as well as the tools to optimize it. Additionally, it will clarify the reasons for the limitations of the current baseline, and identify the areas with greatest research potential.

1.2 Hypothesis

There may be specific improvements to parameter selection strategies of a basic image classifier. In particular, strategies that select the parameters of codebook construction, dense SIFT sampling, and feature complexity may be improved.

Furthermore, the mutual information between two features may be predictable based on the sampling redundancy.

Finally, a basic image classifier may have certain theoretical limitations, defined by its semantic capacity, that are applicable regardless of parameter configuration and amount of training data.

1.3 Definitions and Abbreviations

- *Average precision* is an evaluation metric for an image classifier. The precision is measured at each point on the precision/recall curve corresponding to a retrieval, and then the average precision is the average of those precision values.
    - *Mean average precision* is an evaluation metric for an image classifier in the PASCAL VOC Challenge. It is the mean of all twenty average precision measures corresponding to the twenty object classes in the PASCAL VOC Challenge.
    - *Interpolated average precision* is the average of 11 measurements, consisting of the precision measured at 11 intervals on the range [0.0, 0.1…1.0] on the precision-recall curve.
- A *bag-of-features* is a vector describing an image. Each element in the vector is a count of the number of times some feature appears in the image, regardless of where that feature appears in the image.
- A *codebook* is a list of visual features that are intended to represent the space of all possible features.
- *Context* is the effect that the presence or absence of one feature has on another distinct feature. It allows the total information of two features to be greater than the sum of the information of each separate feature.
- A *corner* is a region in an image where a large gradient occurs over a short distance in two or more directions.
- A *dense sampling strategy* is a sampling strategy consisting of a large number of samples at regular intervals.
    - A *full-dense sampling strategy* is a sampling strategy consisting of a large number of samples at multiple scales, with the same pixel stride for each scale.
    - A *semidense sampling strategy* is a sampling strategy consisting of a large number of samples at multiple scales, with a pixel stride that is proportional to the scale.
- A *descriptor* is the vector of values that comprise a particular feature.
- An *edge* is a region in an image where a large gradient occurs over a short distance.

- A *feature* is a vector describing part of an image. A feature is robust to one or more image deformations and can be compared to other features for similarity, using a basic distance measure.
- A *gradient* is a change in intensity when moving in a particular direction in an image.
- *Hard assignment* is an encoding scheme for codebook matching. For each sampled feature, the nearest codebook feature(s) are found, and the corresponding value in the bag-of-features vector is increased by 1, regardless of the distance measure between the codebook feature(s) and the sampled feature.
- A *histogram* is a frequency distribution, expressed as a vector.
- A *hyperplane* is a boundary learned by a support vector machine. All points on one side of the hyperplane belong in the positive class, and all points on the other side belong in the negative class.
- *Image classification* is the task of determining whether an image does or does not belong to a general category based only the visual information present. Such categories may include "horses" or "cars", but would not include "Secretariat" or "Ford Shelby GT500".
    - An *image classifier* is a computer program whose purpose is image classification
    - A *basic image classifier* is an image classifier that consists of the following steps:
        1. Dense low-level feature sampling,
        2. Construction of a representative codebook of features,
        3. Codebook matching via a bag-of-features approach,
        4. Spatial pyramid matching for organization, and
        5. Learning and classification via support vector machine.
- A *kernel* is a function that maps two vectors to a scalar. It is used in a support vector machine to represent a distance measure between two vectors in an implicit higher-dimensional space. In this higher-dimensional space, two groups of vectors may be easier to separate.
    - An *additive kernel* is a kernel K that can be broken down into a function k on each individual variable in the vectors, such that K is the sum of all of the k values. It is described in mathematical detail in Section 7.1.

- *Object recognition* is the task of determining whether an image does or does not contain a specific object based only on the visual information present. A specific object may be a "Coca-Cola can" but it could not be a "can".
- An *octave* is one doubling of scale.
- *PASCAL VOC Challenge* – PASCAL Visual Object Classes Challenge. The PASCAL VOC Challenge is an annual image classification competition in which participants must classify or detect images among twenty different object categories.
- The *performance* of a proposed method is the positive or negative affect on the average precision of the associated image classifier.
- A *pixel stride* is the number of pixels between the center of adjacent samples in a dense sampling strategy.
- The *point of redundancy* is the point at which decreasing the pixel stride of a classifier starts to provide diminishing returns. It represents the point at which adjacent samples become likely to represent the same visual structure.
- *Sampling redundancy* is the amount of mutual information between two samples based on their overlap in the *(x,y)* space of the image.
- *Saturation* is when a classifier's performance remains approximately constant when the pixel stride is decreased.
- *Scale* is the size at which an image is being inspected for gradients. At low scales, only a few pixels are inspected at the same time, and thus gradients are observed over a short distance. As higher scales are inspected, the image is blurred with a Gaussian filter so that gradients over a short distance disappear. Then, gradients are observed over a longer distance.
- *Semantic capacity* – the ability of a basic classifier to represent context.
- *SIFT* – Scale Invariant Feature Transform.
- *SIFT* features are composed of a 4x4 grid of histograms of gradients. Each member of the grid is a bin that contains a histogram
    - *SIFT8* features are SIFT-like features that consist of an 8x8 grid of histograms of gradients.

- *SPM* – Spatial Pyramid Matching. SPM is an extension to a bag-of-features approach. SPM divides the image into several spatial regions and finds a bag-of-features for each region.
- *SVM* – Support Vector Machine. An SVM
- *Test data* are a set of images provided to an image classifier without labeling. The classifier must use a model it has learned to predict the correct class labels.
- *Training data* are a set of images provided to an image classifier along with the correct class labels. The image classifier can learn from this data.
- *Validation data* are a set of images provided to an image classifier along with the correct labels. The image classifier can use this as practice test data if it needs to calibrate certain parameters (specifically, the *C* parameter in a support vector machine). For each parameter setting, a model is learned from the training data and checked against the validation data. The model that performs best on the validation data will ideally perform best on the test data as well.
- A *visual word* is a feature that strongly correlates (positively or negatively) with an object class, regardless of context.
- A *visual letter* is a feature that does not strongly correlate (positively or negatively) with an object class regardless of context. A visual letter may strongly correlate with an object class in a specific context.

2 Literature Review

2.1 Features

The first step in the process is to convert images from an array of pixels to a sampling of features, each of which contains far more information than a single pixel or an unorganized group of pixels. Many features exist in computer vision, but some have proven particularly useful.

2.1.1 SIFT and Histogram of Gradients

The SIFT feature [Lowe99] introduced the concept of a histogram-of-gradients (HoG), a concept that is used in some form by many low-level features. A histogram-of-gradients takes a small group of nearby pixels, and, rather than describing them by their individual values, describes them in terms of their relationship to each other. This moves the representation away from pixels and toward edges and corners, which have more information and are less likely to be explained by noise.

Figure 2-1

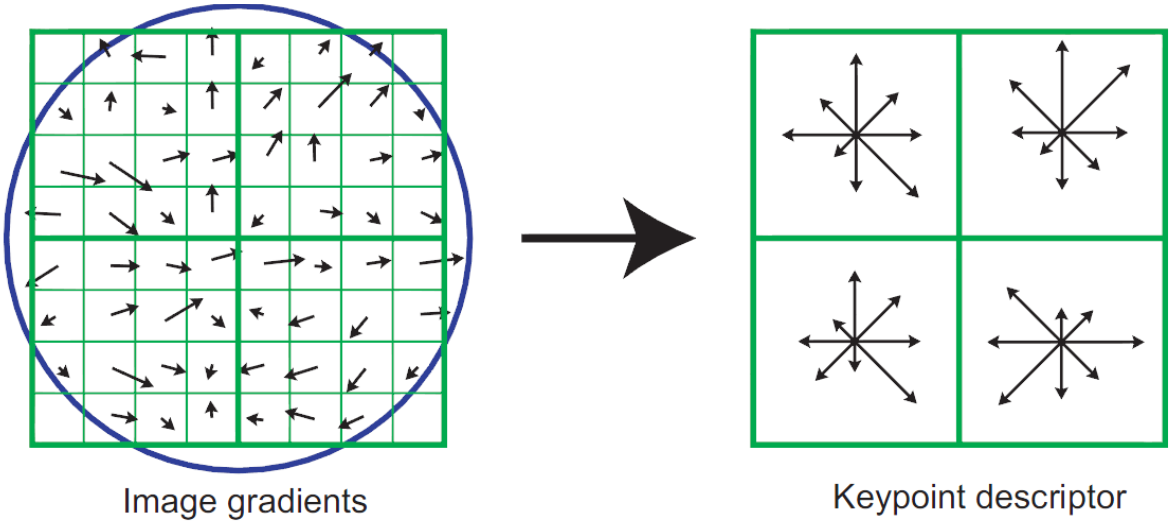


Diagram of a SIFT feature, from [Lowe04]. *Left*: gradients are found at each pixel (small squares) and weighted based on which pixels fall within the circle. *Right*: gradients for each bin (large squares with dark green borders) are accumulated in a histogram.

The SIFT feature is partially or totally invariant with respect to many common image deformations, including position, scale, illumination, rotation, and affine transformation. For

this reason it has been applied in basically every area of computer vision. More relevantly, all state-of-the-art image classifiers use some form of a histogram-of-gradients feature, and many use no other features.

The pyramid histogram of gradients (PHOG) feature uses the same basic components as SIFT, but has a structure with multiple scales. [Bosch07] The feature window is split into different number of bins, and the histograms of gradients are computed similarly to SIFT. Then, the histograms from all scales are concatenated into one feature. This representation has the capacity to show different edge and corner structures at different scales.

The pyramid histogram of words (PHOW) feature has the same pyramid structure as a PHOG feature. However, instead of encoding each scale as one more histograms of gradients, SIFT features are computed at each scale and matched to a small codebook. The resulting bag-of-features encodings are concatenated into one feature. So, the feature is a list of word counts: a microcosm of the representation used for the entire image.

The deformable parts approach begins with a HoG, but provides additional geometric considerations that allow a complex object to be made up of several smaller features ("parts"). Each part will have its own smaller characteristic histogram, and these histograms must be positioned in a particular geometric fashion. Because some error is tolerated, the model is not "rigid", thus allowing the overall shape to be slightly deformed. [Felzenszwalb08]

2.1.2 Other Features

While there are other low-level features besides HoGs used in image classification, these other features still have very similar goals, and some structural similarities. Each feature attempts to look at a group of pixels from a larger image and describe it with a small vector of values that is more informative. This vector should be robust with respect to one or more common image deformations.

Local binary patterns [Ojala94] are frequently regarded as complementary to HoGs, because they tend to encode texture rather than shape. [Wang09] Local binary patterns, or LBPs, are computed at the pixel level, by analyzing the gradients in eight directions (for the basic descriptor). Instead of combining the gradients of neighboring pixels as in SIFT, every pixel is placed in a distinct category based on whether it has a positive or negative gradient in each direction. Thus, for an eight-directional LBP, there are $2^8=256$ possible combinations of gradient

configurations. The image is described by a single vector of 256 values which represent the frequency of each configuration. This vector is more descriptive of texture than shape because the configurations are all calculated from extremely local data; pixels that are not adjacent do not influence each other. [Ojala94]

Attempts to extend SIFT to the color domain have enjoyed some success. Color can either be added to a SIFT or integrated into it. In [Weijer06], a color feature is proposed. Its descriptor is constructed by concatenating a photometrically and geometrically robust color descriptor with the SIFT descriptor. The feature achieves superior classification to a pure SIFT feature. In [Sande10], several 3-channel color models are analyzed. For each color model there is a corresponding feature. For each feature, the descriptor is constructed by computing the SIFT descriptor individually for each of the 3 color channels of its model, and concatenating the result.

The GIST feature is designed specifically for scene categorization. As such, it attempts to capture global characteristics of the image through low-dimensional features. The low-dimensional features exist in a low-dimensional representation called the *spatial envelope*. Attributes including naturalness, openness, roughness, expansion, and ruggedness contribute to the overall categorization of the scene. [Oliva01]

2.2 Codebook Matching

A good codebook satisfies two opposing criteria. On the one hand, a codebook will ideally consist of a small number of highly distinct features. On the other hand, a codebook must be large enough that a representative feature exists for any feature that might be found in a set of images. If two features are visually distinct, they should not match the same feature or features in the codebook.

Many methods for codebook construction exist. Among state-of-the-art image classifiers, k-means clustering is the most common, according to the literature. [Moosmann07, Jurie05] This is confirmed by the survey of current methods presented in Section 2.7.1.

2.2.1 K-Means Clustering

K-means clustering takes a large number of points in n-dimensional space and divides them into a smaller number of clusters, maximizing the distance between each cluster. The process is an iterative one, where at each step, points are assigned to whatever cluster to which they are closest (in terms of Euclidean distance to the centroid). But, because of those assignments, the clusters may now consist of different points, and thus the centroids change. The process can be repeated until the clusters stabilize or some cutoff threshold is reached.

Figure 2-2

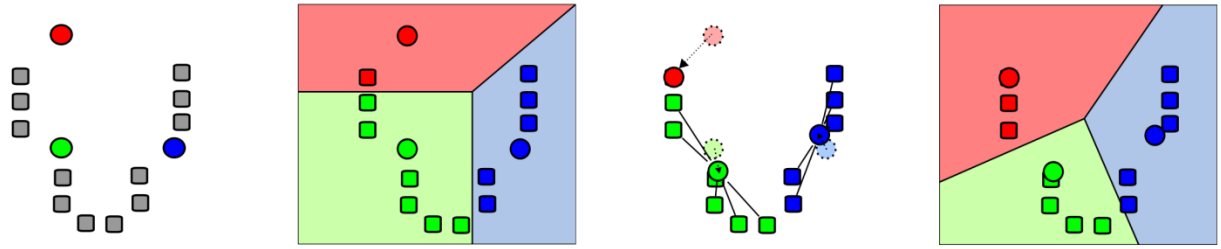


Diagram of k-means clustering shown in four steps, from left to right. 1. k centroids are selected from the data points. 2. Each data point is assigned to the closest centroid. 3. New centroids are computed based on the data points that belong to each cluster. 4. Steps 2 and 3 are repeated to convergence, or some other termination criterion. Images from [Pace07].

Currently, most visual codebooks in state-of-the-art image classifiers are built using some form of k-means clustering. [Moosmann07, Jurie05] It is not clear whether any alternative methods are in use; mention of alternative strategies is absent in the literature.

2.2.2 Alternative Codebook Generation Algorithms

Other algorithms do exist to build an effective codebook from a set of training features. [Moosmann07, Nister06, Jurie05, Wu11]

In [Moosmann07], a set of hierarchical codebooks called clustering trees encode the training features. Each clustering tree assigns an independent classification to each training feature through a hierarchical matching algorithm. Each tree produces a histogram for the image, and all histograms are concatenated into one vector that describes the image.

A hierarchical codebook scheme is also presented in [Nister06]. Codebooks with as many as 10^6 features are used efficiently, as hierarchical matching can be done in time that is logarithmic with respect to the codebook.

In [Jurie05], codebooks are constructed one element at a time. To add a new element, a mean shift estimator finds the maximal density region of a random sampling of training features. However, once a codebook element is added, features that are similar to it in the training set are removed. This produces a codebook that encompasses a wider variety of the feature space rather than over-representing regions that appear more frequently in the set of training features.

Recent work [Wu11] proposes the use of the histogram-intersection kernel in lieu of Euclidean distance to construct codebooks. Improvements over k-means codebooks are observed on some benchmark object recognition data sets.

2.2.3 Universal Codebooks

While codebooks are almost always constructed from scratch for a particular classification task, some research suggests that a universal codebook could be comparably effective on any set of natural images. In [Hou11], codebooks were computed on multiple image datasets, and then every codebook was tested on every dataset. This suggests that a codebook does not need to be constructed specifically for a given classification task.

Other recent work, however, argues that codebooks specifically tailored to each object category are more discriminative. [Yang08]

2.3 Bags of Keypoints

Object categorization seems to require some sort of geometric knowledge, but attempts that largely ignore geometry have been successful. Bag-of-keypoints or bag-of-features approaches look at a large number of features at various points and scales in the image, but do not attempt to rigorously organize them spatially. Instead, the features are mapped to one or more words in a codebook, and the feature counts are summed. [Csurka04] This approach exists in some form in most state-of-the-art classifiers (a survey of successful approaches can be found in Section 2.7.1).

2.4 Spatial Pyramid Matching

Bag-of-features approaches eliminate all of the spatial information accompanying the features. Spatial Pyramid Matching (SPM) is a simple addition to a bag-of-features approach that restores some of that spatial information. [Lazebnik06] When a spatial pyramid is used in conjunction with a bag-of-features approach, there are several vectors constructed instead of one. Each vector is a bag-of-features for one region of the image. Which regions of the image are used is determined a priori by the spatial pyramid. Once all vectors are computed, they are concatenated into one larger vector, one that implicitly contains coarse spatial information.

Figure 2-3

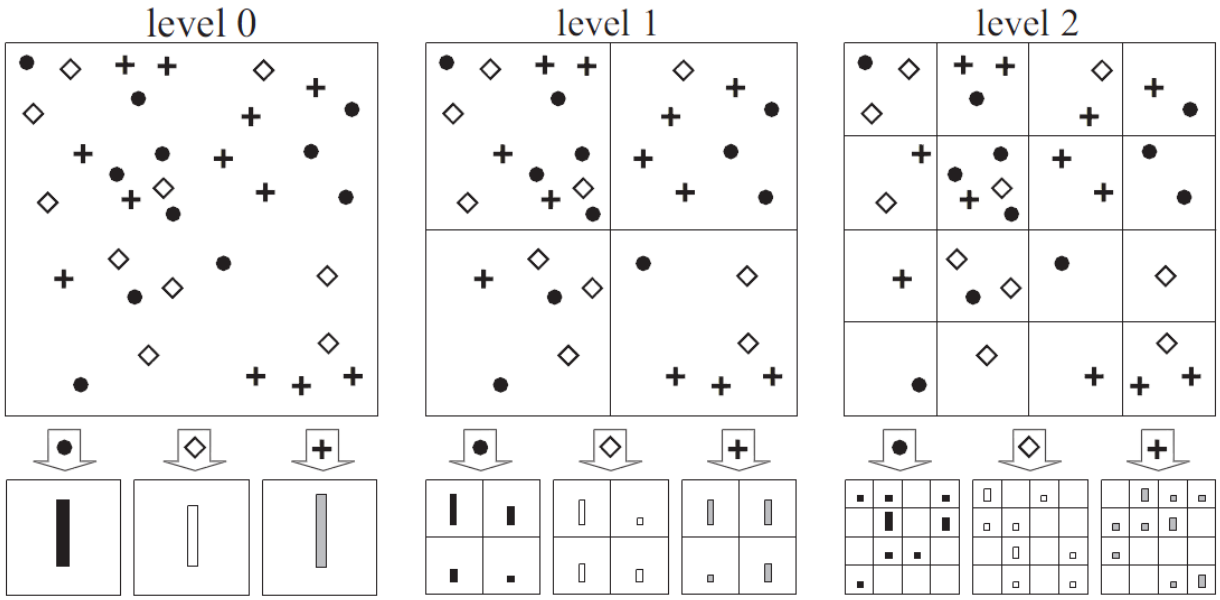


Diagram of SPM, from [Lazebnik06]. *Left*: At the first level of the pyramid, all features are organized into one histogram. *Center, Right*: At subsequent levels, multiple histograms are built corresponding to each region of the image. The concatenation of these histograms describes the whole image.

2.5 Support Vector Machines

Support vector machines are a machine learning tool capable of taking arbitrary vectors of data and separating these vectors into two (or more) classes. [Cortes95] The algorithm uses a kernel function that takes two vectors as input and produces one number as output.

Mathematically, this value represents the distance between the vectors in some higher-dimensional space. Conceptually, this value represents how "similar" these two vectors are.

Restrict the discussion to the two-class example. The support vector machine is given a set of training data consisting of positive and negative examples. The support vector machine uses the kernel function to find the distance between each example, and in doing so learns a separating hyperplane. The hyperplane is a high-dimensional boundary that defines the difference between the positive and the negative class. Then, the class of any new vector can be predicted by computing which side of the hyperplane it falls on.

Figure 2-4

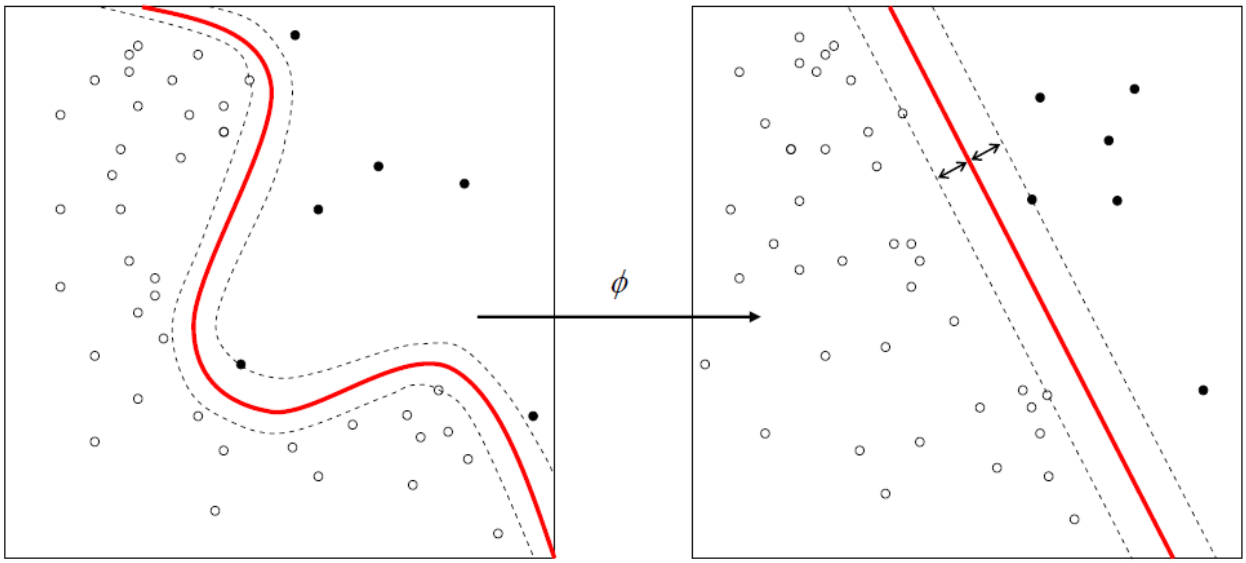


Diagram of SVM. *Left*: In the original problem space, the data points are not linearly separable. *Right*: When transformed into the kernel space, the data points are linearly separable. *Both*: a red line corresponds to the hyperplane maximally separating the two classes. Points that fall on the nearby dotted lines are support vectors, the closest data points to the hyperplane that determine where the optimal hyperplane lies.

The performance of a support vector machine is largely dependent on the kernel chosen. Unfortunately, there are few explicit limitations on what a kernel function can be. The kernel should be appropriate given the underlying structure of the input data. It should be informative, but also efficient to compute between many pairs of large vectors. The χ^2 kernel is a popular

choice for a bag-of-features visual classifier, as are the basic linear kernel and histogram intersection kernel. [Zhang06]

All of these kernels are additive kernels. [Maji12] demonstrates efficient algorithms to exactly compute additive kernels. These algorithms provide the same results but are much faster than the naïve implementation of an additive kernel.

2.6 Affine-Invariant Object Recognition

HoG features are partially invariant with respect to affine deformations, but other attempts exist that specifically address this common image transformation. In [Mikolajczyk04], a feature is described by the values of the second moment matrix. This feature is invariant to affine deformation. Alternately, in [Morel09], affine transformations are simulated on the image, and SIFT features are computed for every transformed image.

2.7 PASCAL Challenge

The PASCAL Visual Object Classes Challenge (PASCAL VOC Challenge) is an annual image classification competition in which participants must classify or detect images among twenty different object categories. The image sets used in the challenge are particularly difficult, showing objects in different poses, scales, backgrounds, lighting conditions, and occlusions. Any successful classifier must be robust with respect to all of these image deformations. Appendix A displays examples images from all twenty object categories, showing both typical examples, as well as difficult examples containing extreme deformations.

There are two primary competitions each year, a classification competition and a detection competition. In the classification competition, an image must simply be classified based on the object(s) found. The object does not need to be localized in the image. In the detection competition, objects must be detected and localized in each particular image. [Everingham12]

While this paper's analysis focuses on the classification problem, advances in both competitions are worth analyzing. Most of the principles that apply to the detection competition hold for the classification competition.

2.7.1 Recent PASCAL VOC Challenge Winners

The winners of the classification portion of the VOC Challenge in 2009 (VOC2009) used basic SIFT features and a linear SVM, but employed advanced coding schemes in the codebook matching phase. [Everingham12]

There were two joint winners of the detection portion of VOC2009. One team's approach was unique at the SVM stage. They employed a series of SVMs with different kernels, including a linear, χ^2, and RBF kernel. [Vedaldi09] The other winning team's approach emphasized the deformable parts model. [Felzenszwalb08]

Both the VOC2010 classification winners and detection winners looked at the co-occurrence of different object classes in the challenge. Thus, the classifier and detector were able to adjust predictions based on contextual relationships between different objects. [Everingham12]

One of the top performers in the VOC2011 classification portion considered the "most telling window" of each image when classifying it. In some sense, they applied detection principles to the classification challenge to improve results. [Everingham12]

The VOC2011 detection winners made further enhancements to the deformable parts approach, using features with reduced dimensionality due to principle components analysis. These features were faster, allowing the algorithm to use more than one model per object and capture different poses accurately. [Everingham12]

Table 2-1

Approaches	HoG features?	Other features?	Codebook construction	Codebook Size	Sampling strategy
UvA-U. Trento 2011	SIFT with Color	no	k-means	4096	Pixel-wise
NUS-Panasonic 2010	SIFT, PHOG	LBP, GIST, color?	?	?	Full-dense, interest point, super-pixel
NLPR 2010	Deformable parts	LBP	?	?	Full-dense (5 scales, 4 pixel stride)
NEC 2009	SIFT	no	n/a	n/a	Full-dense (3 scales, 4 pixel stride)
Oxford-Microsoft India 2009	PHOG, PHOW	visual words, SSIM	?	?	?
UoCTTI 2009	Deformable parts	no	n/a	n/a	Sliding window
UvA-Surrey 2008	SIFT with Color (multiple variants)	no	k-means	4000	Full-dense (multi-scale, 6 pixel stride)
INRIA LEAR 2008	SIFT, overlapping HoG	no	k-means	100	1 scale, 6 pixel stride

Approaches	SPM?	SVM	Special
UvA-U. Trento 2011	yes	histogram intersection	Most telling window
NUS-Panasonic 2010	yes	χ^2, rbf	Context learning on exclusive label sets
NLPR 2010	?	linear	Saliency coding, context learning on class co-occurrence
NEC 2009	yes	linear	Non-linear coding schemes used to encode features
Oxford-Microsoft India 2009	?	linear, χ^2, rbf	Multiple SVMs used iteratively (different kernels)
UoCTTI 2009	no	latent SVM	Deformable parts
UvA-Surrey 2008	yes	χ^2	Kernel discriminant analysis
INRIA LEAR 2008	yes	linear, χ^2	Two-stage (hypothesis, verification)

Table of recent successful approaches in the PASCAL VOC Challenge.

Table 2-1 organizes recent successful approaches in the PASCAL VOC Challenge. All winners or co-winners of the classification and detection competitions since 2008 are represented. For groups who have won multiple years, only the most recent year with available data are considered.

The table shows that all approaches use some form of a histogram-of-gradients feature to describe images. Five use no other features. A full-dense sampling strategy is used in at least six

approaches. An SPM is used in at least five approaches. Four approaches incorporate a linear kernel and four incorporate a χ^2 kernel.

2.8 Parameter Optimization

Some existing work does outline the effect that some parameters have on a standard dense sampling classification algorithm. [Nowak06, Zhang06, Varma07, Chatfield11] Most recently, [Chatfield11] explored the effect of different encoding schemes on classification performance. This paper also considered different parameters setups, varying the sampling rate, codebook size, and SVM kernel of their implementation.

3 Research Questions

The goals of this paper are two-fold. First, this paper will provide a blueprint to optimize the parameters used in a basic classifier. Second, this paper will reveal the specific limitations of a basic classifier and in doing so provide insight into lines of future research.

Specifically, all work will pertain to a basic classifier, as defined by what is common among all state-of-the-art approaches. Given these requirements, the approach is restricted to the following:

1. Dense SIFT sampling,
2. Construction of a representative codebook of features,
3. Codebook matching via a bag-of-features approach with hard assignment,
4. Standard SPM for spatial information, and
5. Learning and classification via SVM with a hard-margin linear kernel.

Within this framework, many logical research questions align themselves with this goal. Specifically, this paper will consider the following:

- "Is k-means clustering useful when constructing a visual codebook?"
- "What is the relationship between codebook size and classification performance?"
- "What is the relationship between the number and scale of low-level features sampled and classification performance?"
- "What is the relationship between the complexity of a SIFT-like feature and classification performance?”

4 Methodology

Our approach first consists of constructing a robust basic image classification program. The program must be able to modify all of the parameters that are to be optimized. The program will need to be able to modify the following:

1. The codebook construction scheme and codebook size,
2. The sampling behavior for the SIFT features,
3. The complexity of the SIFT feature.

Once constructed, the classifier runs under various parameter configurations in order to explore optimal configurations.

4.1 Building a Basic Classifier

A basic classifier must contain all of the following components:

1. Dense SIFT sampling,
2. Construction of a representative codebook of features,
3. Codebook matching via a bag-of-features approach with hard assignment,
4. Standard SPM for spatial information, and
5. Learning and classification via SVM with a hard-margin linear kernel.

Classifiers in the PASCAL VOC Challenge are not required to run in real time, and in fact are given several weeks to calculate results. While this paper will not attempt to consider real-time constraints, the classifier must be efficient enough that numerous tests with various parameters can be performed. Certain stages in the classification process will require optimization. Toward this end, some of the steps are implemented to run on a GPU. A GPU can compute much faster than a similar-cost array of CPUs, if the program can be written to accommodate the GPU's massive parallelization.

4.1.1 Dense SIFT Sampling

Many open-source programs exist to extract SIFT data from an image. Provided that the software is capable of extracting an arbitrary number of SIFT points from arbitrary locations and scales, it is straightforward to build a classifier that can employ any sampling strategy. Specifically, the classifier in this paper makes use of *SiftGPU* [Wu07].

4.1.2 Codebook Construction

Constructing a codebook generally consists of gathering a set of image features and performing k-means clustering on them to generate a smaller set of codebook features. Because k-means clustering can be time-consuming and is parallelizable, a GPU implementation is useful. The classifier in this paper contains an original k-means clustering implementation that runs on the GPU. The GPU implementation has been verified against a much simpler CPU implementation.

The classifier also provides two additional codebook generation strategies. The first strategy builds a codebook of size *K* by selecting *K* features at random from the training set. The second strategy is referred to as "representative k-means". This strategy performs k-means clustering, resulting in a set of *K* centroids. However, instead of using those centroids for the codebook, the strategy finds the closest feature in the training set to each centroid, and uses that feature instead of the c**e**ntroid in the codebook.

4.1.3 Codebook Matching

Matching image features to one or more codebook features is a straightforward problem that requires only some sort of distance measure. The classifier uses simple Euclidean distance as its distance measure for all configurations.

This step has the potential to be the most time-consuming step in the process. The classifier must compare every feature in the training and test data against every feature of the codebook. Thus, a parallelized implementation on the GPU is imperative. At the time of development there was no readily available GPU implementation of codebook matching. Thus, the classifier contains an original codebook matching algorithm that runs on the GPU.

4.1.4 Spatial Pyramid Matching

SPM is easily incorporated into the codebook matching phase. Unless otherwise noted, all tests use the standard spatial pyramid, which consists of 8 regions (full image, 2x2 quadrants, and 3x1 vertical thirds).

4.1.5 Support Vector Machine

The final step requires an SVM to learn a model that can classify every image vector into different categories. The classifier uses SVM^{light} [Joachims99], with some modifications to read in binary instead of text.

For all tests, unless otherwise specified, a hard-margin (*C=100000*) linear kernel is used. The linear kernel was chosen because there is not a consensus among state-of-the-art classifier as to which kernel is best (and may, of course, be implementation-specific). The hard-margin setting for the regularization parameter *C* was chosen because in testing it was found to be roughly equivalent to an optimized *C*.

If *C* is small, the support vector machine will produce a simpler model at the cost of incorrectly classifying some training examples. If the model is significantly simpler, it may generalize better to the test data. However, producing a simpler model is not useful for the tests presented in this paper. This is because the number of training vectors *m* passed in to the support vector machine is smaller than the length of the vectors *n* ($m \approx 5000$, and for most of the tests in this paper, $n = 32000$). As such, a simple model (comparative to the dimensionality of the kernel space) can be found even while requiring that model to have 100% accuracy on the training data.

When comparing against benchmarks in the literature, it is appropriate to use the χ^2 kernel. However, since this kernel is slower to compute, a pre-computed kernel matrix is desirable to limit the redundancy of repeated runs of the SVM. Because SVM^{light} does not provide an option to use a pre-computed kernel matrix, the classifier uses *LIBSVM* [Chang11] for the tests involving the χ^2 kernel.

4.1.6 Approximate running times

Table 4-1 gives an estimate of approximate running times for a basic configuration (approximately 10000 samples per image, standard SIFT features, codebook of 4000 elements, standard spatial pyramid, hard-margin linear kernel). The GPU used in the implementation is a Geforce GTX 590. The CPU used in the implementation is an AMD Phenom II X4 (3.4 GHz).

Table 4-1

Step	Primary Processor	Approximate Running Time
1. Dense SIFT sampling	GPU	10 minutes
2. Codebook Construction		
2a. K-means Codebook	GPU	45 minutes
2b. Random Codebook	CPU	<1 second, can be done in step 1
2c. Representative K-means	GPU	1 hour
3. Codebook Matching with SPM	GPU	4 hours
4. SVM Learning/Classification	CPU	varies, usually 1-4 hours

4.2 Configuration

A configuration of features has two relevant measurements. The first is the running time of the program, and the second is the mean average precision of the resulting classifier. Of the four phases (feature sampling, codebook generation, codebook matching with SPM, SVM classification), the codebook matching with SPM tends to be the most time-consuming, and is thus the most relevant when evaluating the running time.

In a dense feature sampling, every variable has roughly the same impact on the running time of this step. Also, in this step, the control flow is quite regular, and thus the running time is approximately given by

$$T = K * S * D * C * F$$

where *T* is the running time, *K* is an implementation-specific constant, *S* is the number of scales sampled, *D* is the number of samples per scale (density), *C* is the codebook size, and *F* is the complexity (vector length) of the SIFT feature used.

As this is the limiting step, this is the step most in need of optimizing. The majority of the paper will focus on optimizing this step.

4.3 Testing

Unless noted otherwise, all of the tests use the training data provided for use in VOC2011, the most recent PASCAL VOC Challenge. To limit the number of images to train and test, all test use the training set for training and the validation set for testing. These sets are each comprised of approximately 5000 images across 20 object categories. There is no need for a validation step, because no a posteriori optimizations need to be made (in particular, the regularization parameter *C* of the SVM is fixed, for reasons discussed in Section 4.1.5).

All test runs generate a complete set of results as per the VOC classification competition. Thus, the test set is classified according to all 20 object categories, and an average precision is found. The evaluation metric is the mean average precision across all 20 object categories.

4.3.1 Codebooks

First, this paper will explore various parameters for building codebooks.

4.3.1.1 K-Means Clustering

The first question to evaluate is whether or not k-means clustering is actually useful when constructing a codebook. Despite its popularity, there is no evidence in the literature establishing the value of the approach for constructing visual codebooks. It is thus reasonable to be skeptical of its usefulness.

The de facto argument in favor of k-means clustering is "Why not?" The algorithm can be constructed in a way that is not prohibitively time-consuming. Furthermore, building the codebook is a separate step from any other in the process. The size of the final codebook is the only parameter that actually influences how long it takes to perform k-means.

The conceptual advantage of k-means is that it "spreads out" the features in the codebook. Consider a codebook with a pair of features that are nearly identical. Including both of these features would not be discriminative and thus not useful. It may even be detrimental. K-means

prevents such pairs of features from appearing in the codebook. This is because clusters that are close together have a tendency to change over many iterations of the algorithm.

On the other hand, the k-means algorithm was not specifically designed with visual codebooks in mind. The algorithm can be used with any set of vectors, not just vectors that correspond to visual features. Furthermore, the resulting codebook after k-means clustering contains vectors that do not directly correspond to any of the original visual features. They are instead an "average" of many features. It may be possible that an exact feature vector found in a real image is a better codebook candidate than an "average" vector that never appeared in its exact form in the training data.

The strategy for representative k-means builds the centroids for the codebook, but then finds the closest feature in the training set to each centroid, and uses that feature instead of the centroid in the codebook. Such a strategy may have an advantage from clustering as well as an advantage from consisting of actual features.

Section 5.1 compares the performance of both kinds of codebooks built from the k-means clustering algorithm against a baseline codebook of randomly selected training features.

4.3.1.2 Codebook Size

The second element pertaining to codebooks is the relationship between codebook size and classification performance. A larger codebook has the potential to give more accurate classification results, but a larger codebook takes longer to match. Defining a mathematical relationship between codebook size and classification performance would make it feasible to find the correct codebook size for a given time constraint.

However, the ideal codebook size could vary based on other parameter settings. For example, as more samples are taken from each image, it may be plausible that a larger codebook can be useful to make finer distinctions between features. While it is not feasible to test every possible configuration of parameters, Section 5.4 explores the relationship of codebook size to the sampling rate.

4.3.2 Dense SIFT Sampling

Next, this paper will consider the various sampling approaches for finding SIFT features in an image. The task is simplified slightly because the classifier will not attempt to localize the

salient regions of the image. Instead, all of the samples will be equally dense across the image. However, even with this constraint, there are a number of choices available.

4.3.2.1 Full-dense Sampling Versus Semidense Sampling

The first question is whether the classifier should collect the same number of samples from every scale, or collect fewer samples from larger scales. For example, consider a sampling strategy that collects features at the lowest scale $s=1.0$ (16x16 pixels). The strategy collects these samples at a 4 pixel stride, that is, 4 pixels between samples. For $s=2.0$ (32x32 pixels), should the strategy also collect samples at a 4 pixel stride, or at some larger interval to account for the fact that the samples are themselves larger?

Figure 4-1

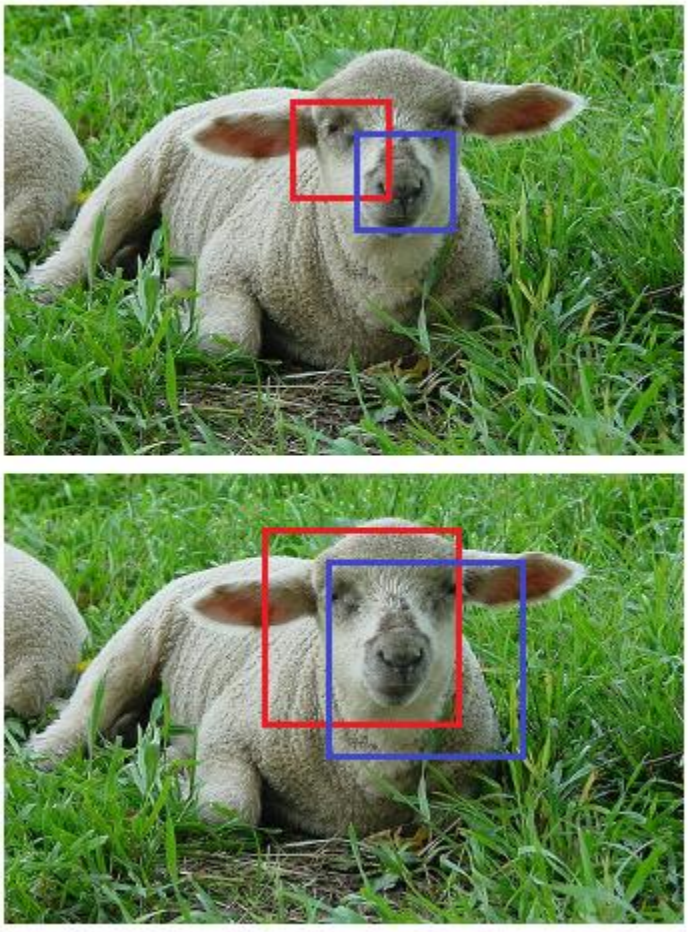


Diagram of overlapping sample windows at two sizes. The pixel offset between the red window and the blue window is the same in both images. Because the windows in the bottom image are larger, they have greater overlap.

Figure 4-1 illustrates the ambiguity. In both examples, the two sample windows are offset by the same number pixels, horizontally and vertically. There is proportionally more shared area between the second pair of samples. Is there more redundancy between the second pair of samples? This is an open question that must be clarified in testing.

Current multi-scale sampling strategies employ a strategy that uses the same pixel stride for every scale. Such a strategy would be sub-optimal if:

1. There is less information available at higher scales. This would imply that higher scales should not be weighted equally.

2. Higher scale features are more redundant at the same pixel stride, due to sharing a greater overlap in terms of the proportional area that each covers. This would imply that fewer samples are necessary at higher scales.

An alternative strategy to the current strategy of taking an equal pixel stride at every scale would be to take a pixel stride proportional to the width of the samples at that scale. So, if the alternative strategy uses a 4 pixel stride for *s=1.0* (16x16), then it would use a pixel stride of 8 for *s=2.0* (32x32). As defined in Section 1.2 this alternative strategy will be called "semidense", in contrast to the current "full-dense" strategy.

This semidense sampling strategy has the potential to be more efficient, allowing the classifier to look at more scales with minimal increase in time cost. But, each successive scale will have less impact on the resulting bag-of-features vector, so this does not guarantee that it will be more effective. Section 5.2 compares the two strategies with multiple scale choices.

4.3.2.2 Number of Scales

In practice, a small number of scales are used in a dense feature sampling. But a robust image classifier needs to be able to recognize objects at a wide variety of scales. The PASCAL VOC Challenge data sets contain examples of objects at many different scales, so it would seem that sampling as many scales as possible is desirable.

There are actually two dimensions involved when determining the scales. The first is the total number of scales sampled, and the second is the spacing factor between. But, it is more convenient for the implementation to vary the number of *octaves* sampled, and the number of *scales per octave*. The number of *octaves* is the number of doublings in scale, and the *scales per octave* determine the step size in scale between doublings. Thus, a *2 octave x 2 scales per octave* sampling would sample at scales $s = \{1, \sqrt{2}, 2, 2\sqrt{2}\}$.

Section 5.2 explores various settings of *octaves* and *scales per octave*, and evaluates them with respect to both semidense and full-dense sampling.

4.3.2.3 Sample Rate per Scale

Next, this paper will consider the problem of how many features to sample at each scale, which the classifier controls by varying the pixel stride (pixels between each sample). If the pixel stride is equal to the pixel width of the feature window, then there will be no overlapping pixels

between samples at the same scale. However, smaller pixel strides are possible and in fact are common among state-of-the-art approaches. Pixel strides of 4 or less are reported in the literature. But, even for features at the lowest scales, this results in significant overlap between samples. So there is some redundancy present, and diminishing returns for higher sample rates are inevitable.

Section 5.4 explores various settings of the sample rate along with the codebook. The relationship between this pair of parameters and the classification quality can be determined.

4.3.3 SIFT Feature Complexity

The original SIFT implementation uses a feature consisting of a 4x4 grid of histograms. However, this implementation was optimized with respect to finding image correspondences between two pictures of the same particular real-world object. An image classifier must find much a more general relationship between different versions of the same object. It must recognize the similarity between different dogs, and between different cars, and between different people. In the PASCAL VOC Challenge, classifiers also have access to numerous training images. Classifiers traditionally employ dense feature samplings while the original SIFT implementation attempts to find only salient sample points.

With these considerations in mind, it is possible that different SIFT grid sizes, such as 8x8, are better suited to the domain of image classification. This paper will evaluate the performance of these SIFT8 (8x8) features while varying other parameters such as codebook size and sample rate. Section 5.5 compares the performance of SIFT8 features against standard 4x4 SIFT features, keeping in mind the difference in processing time for the larger features.

4.3.4 Additional Tests

During testing, a number of additional tests became necessary to clarify the results of the original tests.

4.3.4.1 Individual Scale Information

To clarify the behavior observed in the testing of the full-dense sampling strategy versus the semidense sampling strategy, the behavior at individual scales must be observed. Section 5.3 explores various samplings of exactly one scale. The single scale chosen is varied, as well as the sampling rate.

4.3.4.2 Validity of Results

Due to implementation differences, the results cannot be directly compared against a baseline. However, to provide evidence of the correctness of this paper's implementation, its performance can be evaluated against a similar implementation in the literature. One of the configurations presented in [Chatfield11] is already similar in some ways to the basic image classifier.

To increase the similarity of the comparison, the following modifications are required for this test:

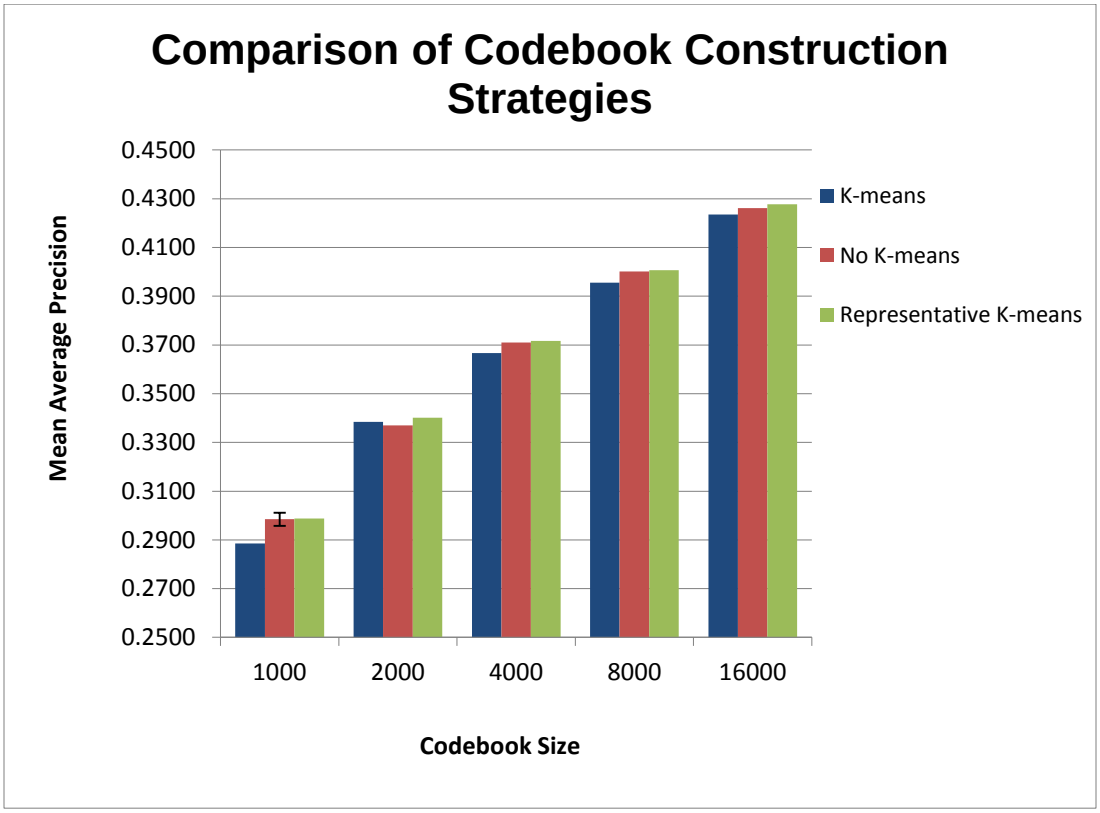
1. Testing is performed on the PASCAL VOC2007 data set, the most common set used in benchmarks in the literature.
2. Sample features are taken from the full-sized original image, rather than from an image that is scaled down initially.
3. The SVM uses a χ^2 kernel instead of a linear kernel, optimizing the regularization parameter with the standard training/validation sets provided in the VOC2007 data set.
4. The scoring for each object class is performed according to interpolated average precision (as specified in the VOC2007 competition protocol), rather than mean average precision.

With these modifications, the implementation is similar, though not identical, to one of the classifiers presented in [Chatfield11], labeled as configuration *(n)* on the table in pages 7-8. The results of both implementations should be similar. The results of both implementations are compared in Section 5.6.

5 Results

5.1 Codebooks

Figure 5-1



Fixed parameter settings: Semidense sampling, 2 octaves x 2 scales per octave, 4 pixel stride at scale=1.

The results of Figure 5-1 confirm that a codebook built from k-means centroids is not superior to a codebook built from random features in the training set. The results also demonstrate that a codebook built from the most representative training feature of k-means clusters is not superior to a codebook built from random features.

The results demonstrate the fact that a codebook consisting of actual features from the training set has an intrinsic advantage over one that does not, if codebooks are otherwise similar. Consider a feature *A* from a training image, to be matched in the encoding phase. If the codebook contains *A* itself, then there is a perfect match present. *A* maps to *A*. For any hard coding scheme, with a codebook of any size, this is the only time that a feature gets encoded without losing any information.

So, for any feature in the codebook that was sampled from a training image, there is at least one example in the training data that is a 100% "correct" match. This particular cat's eye (feature *A*) in the training image is exactly like the cat's eye in the codebook. Also, a cat's eye in a different image (feature *B*) that is very similar to the original may be very likely to match feature *A* in the codebook. But, if the codebook has no feature *A*, and instead consists of k-means centroids, then features *A* and *B* only match if both features map to the same centroid. This is less likely to occur, because now there are two mappings that have to occur correctly, when before there was effectively one.

This advantage alone is sufficient to account for the difference between the performance of k-means codebooks and codebooks without k-means. As testing confirms, choosing a feature that actually exists in the training set (representative k-means) consistently provides a (small) improvement over k-means.

The next factor that has to be considered is the inherent crudeness of encoding millions of image features in terms of a comparatively small codebook. No matter how intelligent a strategy for codebook selection is, it will produce a few thousand representatives that must adequately encode a feature space that represents about 2^{500} possible features (for a 128-dimensional SIFT feature). Not all possible values of a SIFT feature correspond to something that would be common or even existent in real image data, but the intrinsic limitations of the approach are clear. Even some theoretically optimal codebook would be limited in what it can discern between if the codebook is not large enough.

There do exist encoding schemes to hard coding that provide superior results. Such encoding schemes can extend the limits of what a certain-sized codebook can achieve. But, even for many of these schemes, increasing codebooks to very large sizes continues to improve classification, albeit with diminishing returns. [Chatfield11] Current technology has not yet reached a point where one can build a visual codebook that is in any way exhaustive.

Continuing on the idea of larger codebooks: the usefulness of any clustering or selection strategy deteriorates when a larger fraction of the training features may be selected for the codebook. As an example, if there are 1 million training features, from which 999,999 may be selected to build a codebook, then no strategy can be significantly superior to random selection. No strategy can generate a codebook with more than one differing feature to any other. Given this behavior, at some threshold, random sampling becomes equivalent to any other strategy.

Another limiting factor for visual codebooks in a basic image classifier is that they can only represent an image as some linear combination of the features in the codebook. There is no additional meaning ascribed to particular pairings or groups of features. Attempting to do so requires analysis of many variables: if there are *n* codebook features, there are $O(n^2)$ pairs of codebook features. Attempting to assign a variable to each pair or group of features becomes intractable quickly. State-of-the-art approaches do not consider combinations of features in this manner.

But ignoring these relationships, the context of the features, pushes that contextual responsibility down to the feature level. It requires, at minimum, that a single feature must have some appreciable amount of information by itself, free of any sort of context

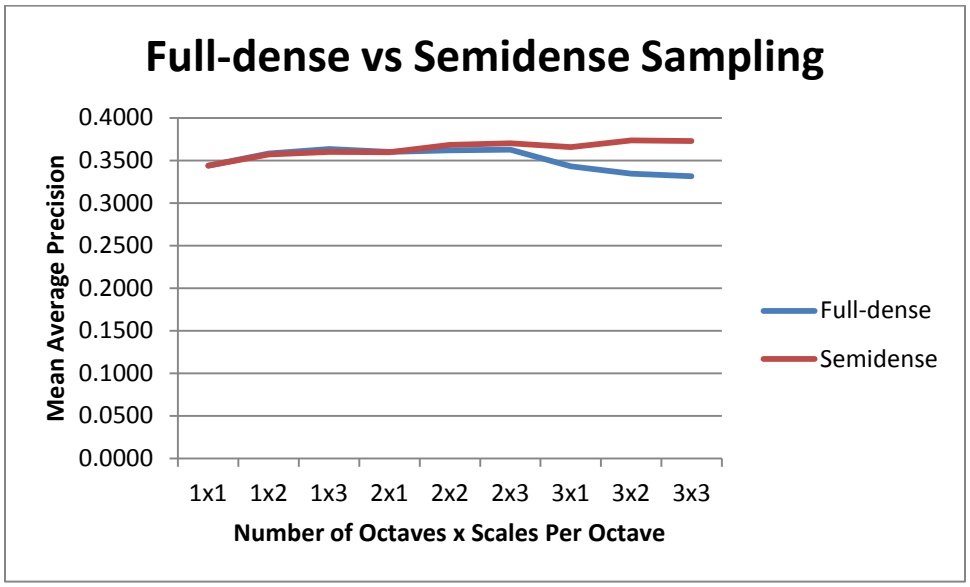
This requirement, that a single visual word have meaning, is consistent with the traditional definition of "word": a semantic unit. In other domains it is clear to see that a "word" possesses information on its own. Consider document classification: if the word "endocarditis" appears in a document, there is appreciable evidence that it is a medical document. The single word has meaning in and of itself. Can the same be said of a SIFT (or SIFT-like) feature?

The fact that bag-of-features approaches work at all for image classification suggests that there is some information present even in a single SIFT feature. But is there enough information to constitute a whole "word"? The original viability of SIFT was demonstrated by finding correspondences between images. A combination of multiple SIFT matches with geometric agreement implied that the same object existed in two images. But that geometric agreement implies a relationship between those features. Any one feature match by itself means little; multiple features grouped in a certain way mean something. In the original algorithm, SIFT features are treated more like visual "letters" than visual "words". This shortcoming will be explored in greater detail in Section 5.5.2, as well as in Chapter 7.

Research into effective codebooks is limited by the discriminative quality of the visual words they encode. As mentioned in Section 2.2.2 there is work in the area of constructing better codebooks with more sophisticated encodings. However, the work has seen little representation in the PASCAL challenge. Many of these more sophisticated codebook structures could be more discriminative if the features they encoded were themselves more discriminative. Evidence of this is beyond the scope of this paper, but a basic adjustment to SIFT features that provides greater complexity and discriminative power is presented in Section 5.5.

5.2 Full-dense Versus Semidense SIFT Sampling

Figure 5-2



Fixed parameter settings: 4 pixel stride at scale=1, random codebook, codebook size=4000.

Table 5-1

O x S	R_S	R_D	R_S / R_D
1 x 1	1.00	1.00	1.00
1 x 2	1.50	2.00	0.75
1 x 3	2.03	3.00	0.68
2 x 1	1.25	2.00	0.63
2 x 2	1.88	4.00	0.47
2 x 3	2.53	6.00	0.42
3 x 1	1.31	3.00	0.44
3 x 2	1.97	6.00	0.33
3 x 3	2.66	9.00	0.30

(O x S): Octaves x Scales per octave; (R_S): total samples taken divided by samples taken at scale=1, semidense strategy; (R_D): same as R_S, but for full-dense strategy

Figure 5-2 demonstrates the superiority of semidense sampling to full-dense sampling. The performance is comparable at all levels before eventually becoming superior, and fewer samples are taken at all levels (except for *1 octave x 1 scale*, where the methods are equivalent).

At one octave, there is little difference in performance between the full-dense and semidense strategies. This is can be attributed primarily to the fact that the methods are not very different when only small scales are used. At *1 octave x 1 scale* the methods are equivalent. At *1 octave x 2 scales*, the methods get an equal sampling for the first scale, and the full-dense sampling takes twice as many features for the second scale. Adding a third scale per octave has minimal positive impact for either method. When more octaves are used, adding a third scale per octave is detrimental for both strategies in some tests.

For two octaves, the performance between the two strategies is similar. But, the semidense sampling strategy takes far fewer samples at the second scale, resulting in far fewer samples overall (less than half if two or three scales per octave are used; see Table 5-1).

When a third octave is introduced, the semidense strategy improves, but it is marginal. This is due to the fact that the first scale of the third octave has only 1/16 as many samples as the first scale of the first octave, and thus little influence on the resulting feature counts. However, the full-dense strategy experiences a sharp decline in performance as scales at the third octave are introduced. The problem is the lack of information present in the scales of this third octave, proportional to the number of features sampled.

The scales of the third octave contain some useful information. Given that the objects of interest can occupy a wide range of scales in our images, there are many useful features to be found even at large scales. But it is also unlikely that any two different scales possess an equal amount of information.

Consider the two most extreme scales. At the small end, a scale with features of one pixel in width could be considered to consist almost entirely of noise. The basic motivating assumption behind research into visual features is that one pixel does not have significant information on its own. If it did, then a simple bitmap of pixel values would be a strong feature. At the other end of the spectrum, an arbitrarily large scale would correspond to an image that is Gaussian-blurred entirely into homogeneity, consisting of one color.

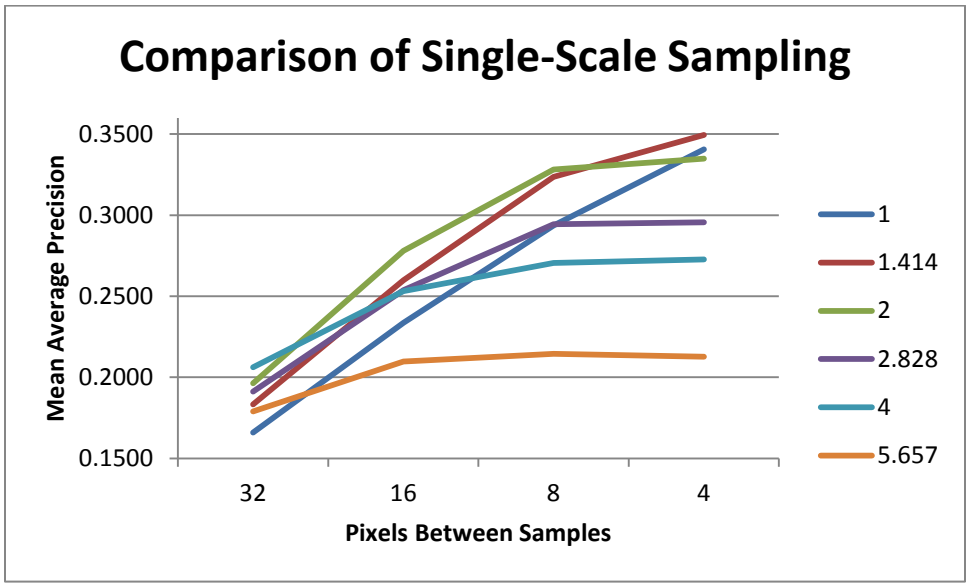
So all useful scales lie in between the two ends of this spectrum, but they may not be of equal usefulness. This presents a problem, because sampling additional scales can actually make an algorithm not only less efficient, but also less accurate. This behavior is responsible for the decline of the full-dense strategy in the third octave in Figure 5-2. The reason for the decreased performance is that every feature is treated as equal when mapping to the codebook. The highest

scales have some useful information, although not as much as the lower scales. But, since the encoding scheme does not account for the redundancy between features, each scale has an equal number of samples. Thus, each scale has an equal proportional influence on the final classification. Taking too many samples of the scales with less information dilutes the effect of the scales with more information.

At this point, however, it is not clear exactly how much information is present at each scale. The proportional quality of different scales can be discovered by testing samples of a single scale at a time and observing the performance. If each test samples one scale to saturation, then each test provides an indicator of how much information is actually present at each scale. The data will indicate a trend, where information may decrease at increasing scales. The next section will explore this.

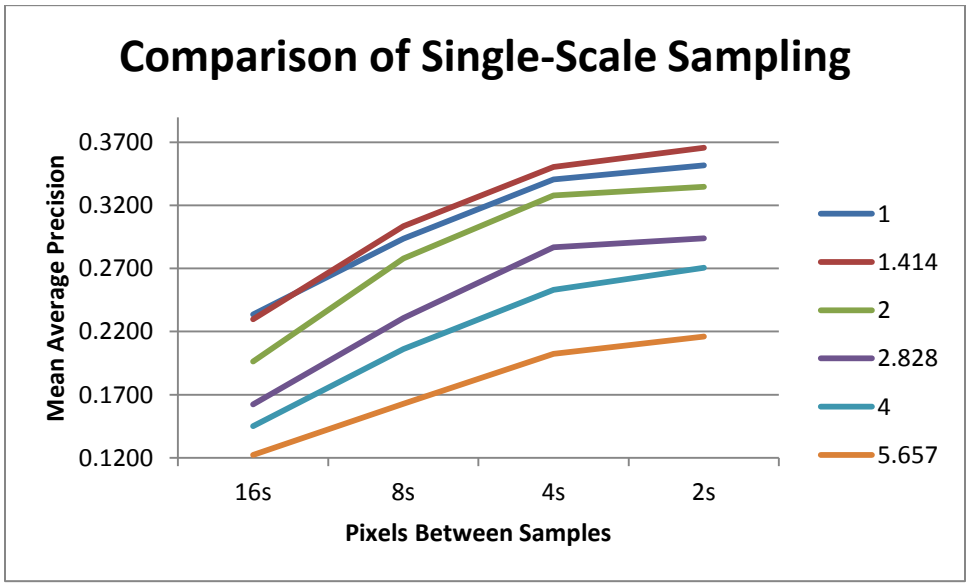
5.3 Individual Scale Information

Figure 5-3



Fixed parameter settings: Semidense sampling, random codebook, codebook size=4000.

Figure 5-4



Fixed parameter settings: Semidense sampling, random codebook, codebook size=4000.

Figure 5-3 and Figure 5-4 explain the behavior of individual scales. Figure 5-3 compares individual scales, varying the sampling rate independently of the scale chosen. Figure 5-4 compares individual scales, varying the sampling rate proportionally to the scale chosen.

The data confirm that there is in fact greater redundancy at increasing scales, given the same sampling rate. At first each scale displays improved precision at approximately a linear rate with respect to the logarithm of the samples taken (each successive test represents four times as many samples). Eventually, each scale except *s=1* reaches a point each successive scale provides diminishing returns. The location and meaning of this point is clarified in Chapter 6.

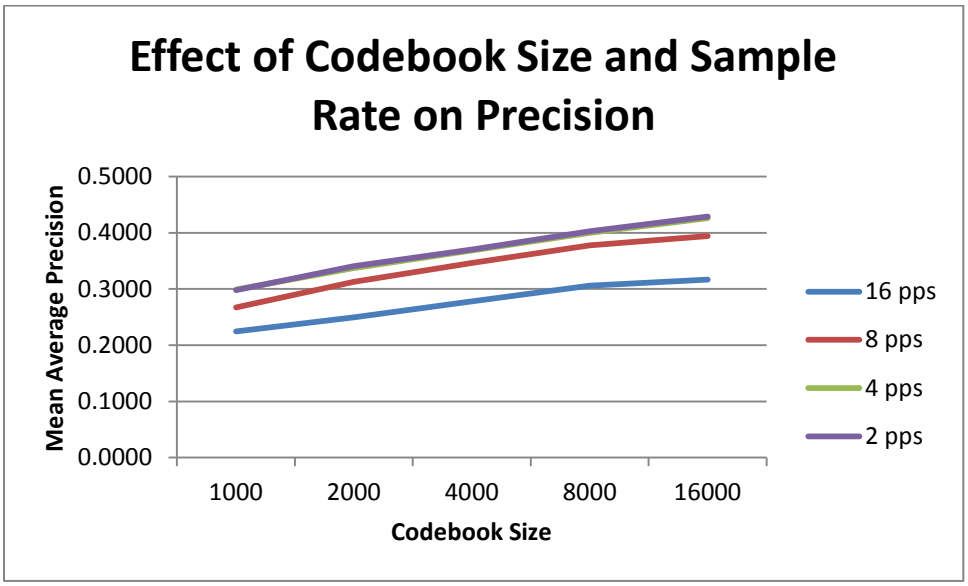
Some of the higher scales give inferior performance, even when sampled to saturation. This means there is less information present at those scales, at least when sampled by SIFT features.

Both of the implicit assumptions of full-dense sampling as stated in Section 4.3.2.1 are refuted by the data here. First, the data show that if the pixel stride is held constant, there is greater redundancy at higher scales. Thus, it is neither necessary nor useful to take as many samples at higher scales. Second, the data show that even when sampled to saturation, the higher scales provide inferior classification performance. Thus, higher scales should not contribute equally to the final feature counts.

Appendix B show examples images from different classes at *s=1* and *s=4*. These images illustrate some of the differences in what types of features are found at higher and lower scales.

5.4 SIFT Sampling Density

Figure 5-5



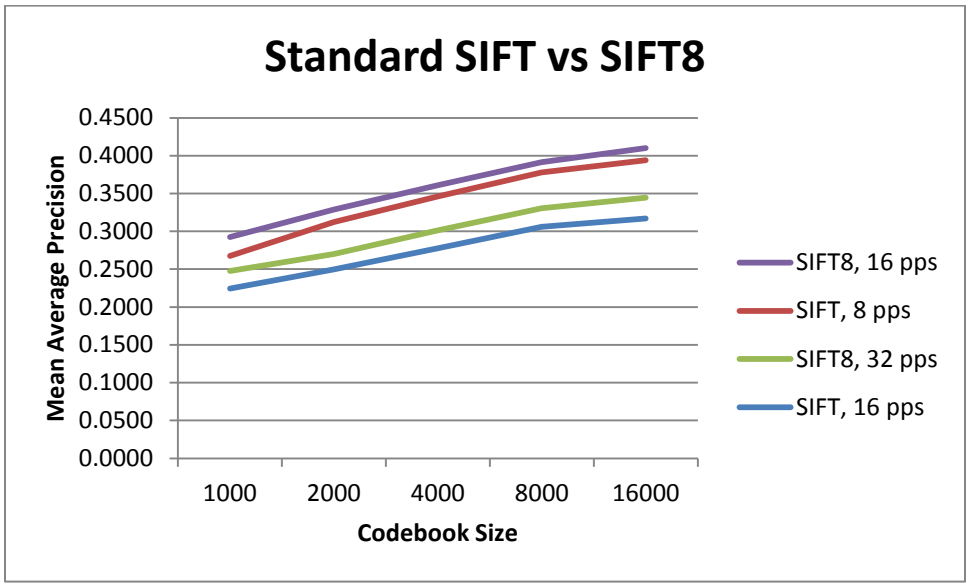
Fixed parameter settings: Semidense sampling, 2 octaves x 2 scales per octave, random codebook. (pps) = pixels per sample = pixel stride.

Figure 5-5 shows that larger codebooks and denser sampling rates both can increase the performance of a classifier, albeit with diminishing returns. The data suggests that there is a hard ceiling near the 2 pixel stride, and this trend is corroborated by the data in [Chatfield11]. The meaning of this ceiling will be clarified in Chapter 6. The sampling rates documented in recent approaches approach or exceed this sampling rate. Even if minor performance gains could be tolerated past this, it is highly unlikely that any sampling strategy would benefit appreciably with a sampling rate that is greater than per-pixel.

Contrastingly, codebooks of much larger size than those sizes used in state-of-the-art classifiers continue to give increasing performance. This is corroborated again by the data in [Chatfield11]. These trends constitute evidence that many state-of-the-art algorithms could derive improvements if they reduced their sampling to a 4 pixel stride, used a semidense approach, and used the corresponding reduction in time complexity to build a much larger codebook.

5.5 SIFT Feature Complexity

Figure 5-6

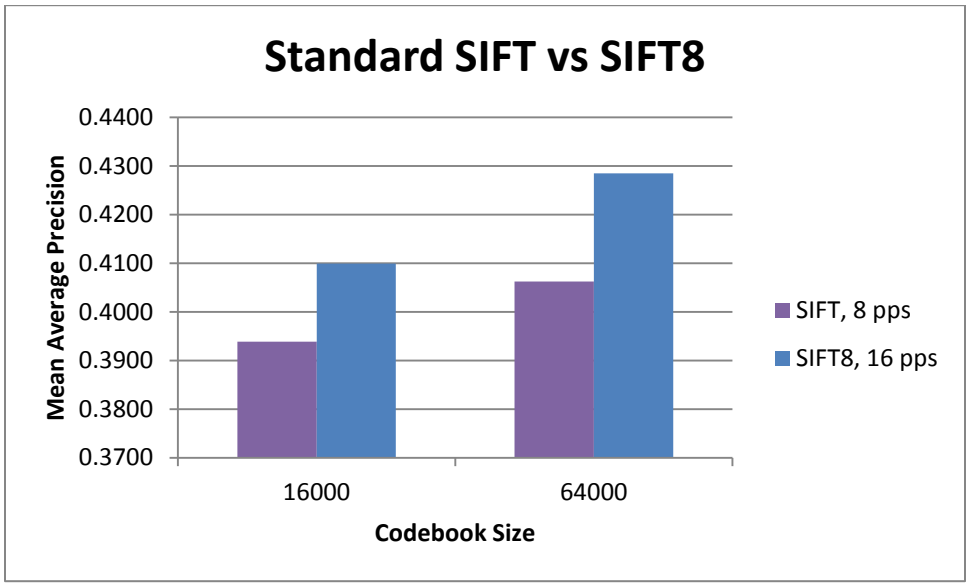


Fixed parameter settings: Semidense sampling, 2 octaves x 2 scales per octave, random codebook. (pps) = pixels per sample = pixel stride.

Figure 5-6 demonstrates that the SIFT8 features are superior to the standard SIFT features for matching. The SIFT8 features are four times the dimension of the standard SIFT features (512 elements instead of 128). As such, they require four times as long to match to the codebook, and to sample. However, if the classifier takes four times fewer samples for the SIFT8 features (32 pixels between samples instead of 16, or 16 instead of 8), the running time is comparable. The results show that even with fewer samples, the SIFT8 features still produce better classification results.

Furthermore, because SIFT8 features are more complex, they are more amenable to increasing performance by increasing the codebook size, even to very large values. Figure 5-7 shows test with a codebook of 64000 elements, much larger than those used in the literature (the largest being codebooks with 25000 elements reported in [Chatfield11]).

Figure 5-7



Fixed parameter settings: Semidense sampling, 2 octaves x 2 scales per octave, random codebook

5.5.1 Original Motivation for SIFT

To understand why larger features are superior in these circumstances, one needs to understand the structure and purpose of the standard SIFT feature. Consider the motivation that produced the original SIFT feature. SIFT features were originally optimized to recognize specific objects, not classify general images. The assumptions of that problem demand an algorithm with very different optimizations than a general classifier. Such assumptions encourage simpler SIFT features. The following sections explore two of these key assumptions.

5.5.1.1 Similarity of Images

First, in specific object recognition, the images to match are very similar. Two pictures of the same phone taken at different angles in different lighting are far more similar than any two pictures of two different phones. Thus, for a robust object recognition algorithm, it is reasonable to require feature matches to be close. Such a requirement encourages features that are easier to match more precisely. SIFT features are relatively easy to match precisely, given their discriminative capability. This is owed in part to their simplicity.

To observe the relationship between simplicity and reliable matching, consider the problem of affine invariance. It is possibly the most challenging invariance to address in a visual system. This is due in large part due to the fact complete affine invariance is not desirable. Full

three-dimensional affine invariance for a three-dimensional object model makes it nearly impossible to distinguish between different objects, when there is only a single two-dimensional available to build an implied three-dimensional model.

Figure 5-8

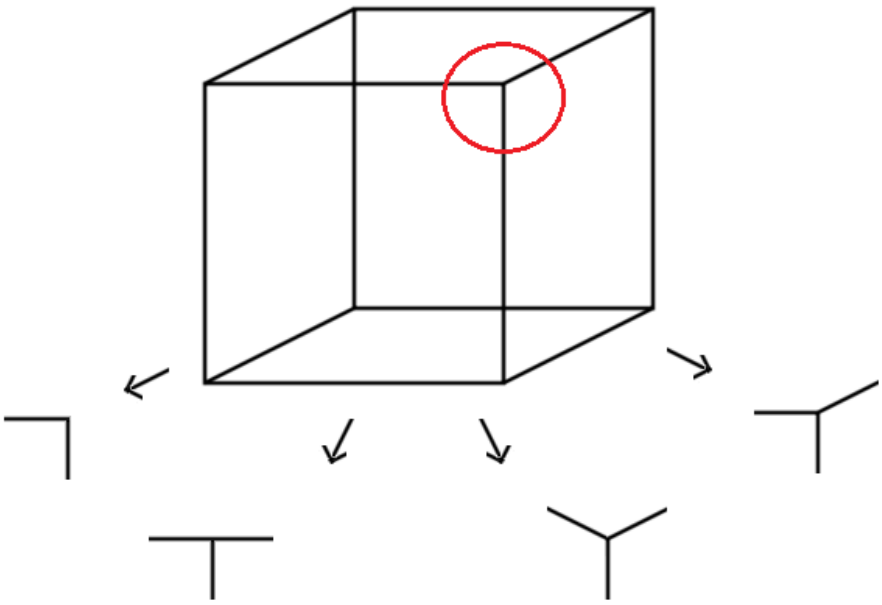


Diagram of affine invariance. The corner circled in red appears different depending on the viewpoint; four examples are shown here.

Observe Figure 5-8. Many different two-dimensional visual structures can be produced from the same three-dimensional object. A three-dimensional model that is fully affine invariant can make no distinction between any of these two-dimensional visual structures (without synthesizing multiple images from multiple viewpoints. Many applications, including the PASCAL VOC Challenge, do not provide such opportunities for multiple viewing angles). A model that can make no distinction between a “T” shape and a “Y” shape is limited in what it can express.

Most successful features do not attempt to model objects in three dimensions anyway, but three-dimensional information is relevant. While planar approximations of three-dimensional objects have demonstrated effectiveness, a certain degree of three-dimensional affine invariance is desirable, because most interesting objects are not planar. Of the 20 object categories of the PASCAL VOC Challenge, none consist of strictly planar objects.

SIFT features are not fully affine-invariant, because there is no normalization performed for that specific purpose. However, hidden within the "binning" behavior of the feature is an incidental robustness to affine transformations. If the viewing angle changes slightly, but the edges and corners of the object in question still fall into the same bins, the values of the resulting SIFT descriptor will change little. The amount of invariance to affine deformation depends largely on the object being deformed, but the original work reported tolerance of rotations of 60 degrees from the viewing angle for planar objects, and 20 degrees for three-dimensional objects. [Lowe99]

If a feature divided the same viewing window into more bins, it would possess less of this incidental affine invariance, because small shifts in viewpoint would be likely to push a corner or edge into a different bin. This causes the resulting SIFT descriptor to be very sensitive to viewpoint, and less robust for matching. This is illustrated in Figure 5-9.

Figure 5-9

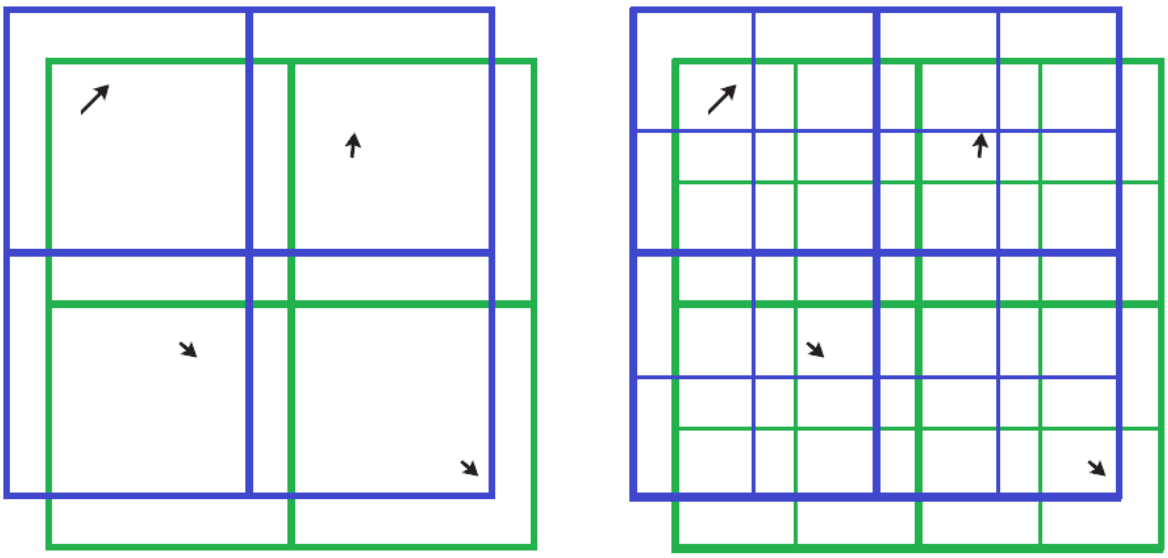


Diagram of decreasing invariance with increasing bins. *Left*: a small shift in location between the blue window and the green window does not result in the important gradients (arrows) falling into different bins.

***Right*: each feature is the same size but has more bins. The same shift causes the top-right gradient to fall into a different bin. The resulting descriptor will change significantly as a result.**

Therefore, the original implementation of SIFT is well-suited for finding correspondences in images that are very similar. When objects are similar, precise matches are desirable, and less complex features (features with fewer bins) match more precisely.

5.5.1.2 Geometric Constraints

Second, in the original SIFT algorithm, geometric constraints add structure to the problem. Because the objects it intends to recognize are mostly rigid, it makes sense to require the SIFT algorithm to search for a very specific geometry. But, this geometry provides additional high-level information, that allows the SIFT features themselves to be less complex. One SIFT feature does not have to be significant on its own. This makes it conceptually closer to a visual "letter" than a visual "word". So again, a simpler, more reliable feature will be preferable, because it becomes more important that it is robust for matching, and less important that a single feature is a strong indicator of an object.

Figure 5-10

Diagram of SIFT matching, from [Lowe99]. *Top*: Images of original objects. *Center*: Image to search for objects. *Bottom*: Correct SIFT feature matches are shown as small squares. Dominant orientation is indicated by a line from the center of the square. Estimated boundaries of the objects are shown by the large rectangles.

Observe Figure 5-10. The features that have been matched do not correspond to large, singularly meaningful portions of the object. At most, they encompass a few letters of print. But there are enough correspondences found here that it is not necessary. And in fact, limiting the complexity of the features allows robust matching to smaller, less consequential portions of the object. Smaller features can provide matches when a larger part of the object is unable to be matched, due to occlusion, shadows, specular highlights, or even physical destruction (if the box was scuffed or marked).

This approach is made possible by the additional geometric constraints that are imposed. Many spurious feature correspondences appear in the entire image. The SIFT features are simple enough that they generate frequent incorrect matches. But these incorrect matches are ignored when they do not agree geometrically with multiple other features, and the resulting recognition algorithm is robust.

5.5.2 Moving towards Generality

Neither of the two assumptions from the previous sections holds for general image classification. If a specific dog is present in the training set, it will not be present in the test set. Thus, the classifier cannot rely on feature correspondences between two pictures of dogs to be perfect or even close to perfect.

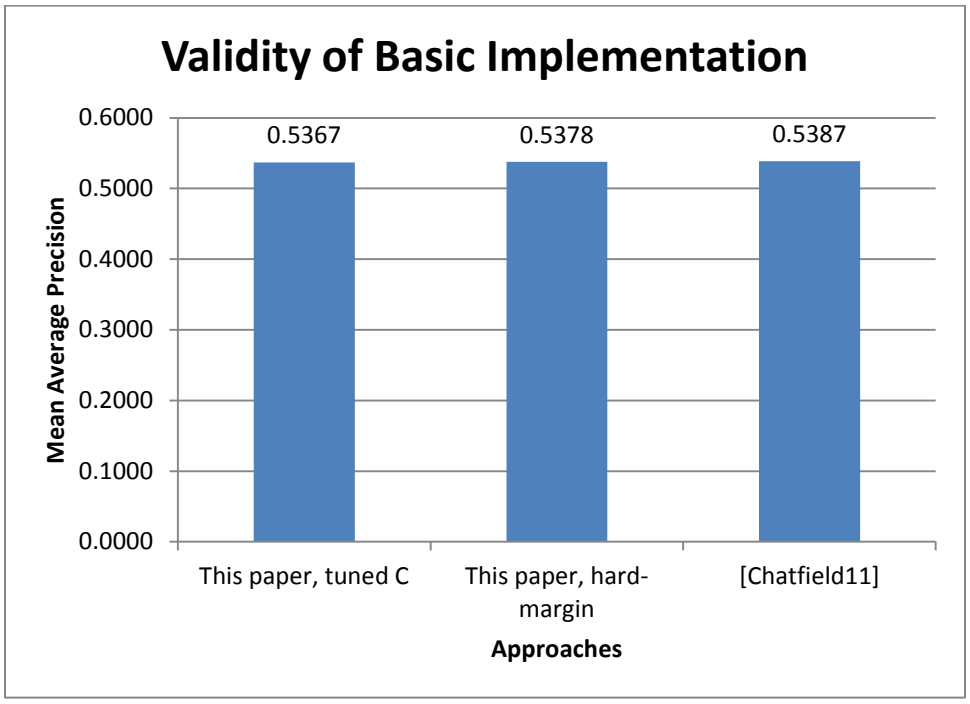
Regarding rigidity: none of the object classes in the PASCAL VOC Challenge that represent natural objects (bird, cat, cow, dog, horse, person, plant, sheep; 8 out of 20 categories) are rigid at all. Furthermore, there is evidence that even rigid objects can benefit from a non-rigid representation, because it allows a more flexible geometric definition that generalizes across different instances of the object. [Felzenszwalb08] That is, two bottles may have similar contours, but different proportions. There may be a single model that can fit both, provided that the model does not impose a rigid representation.

So, the original optimization of SIFT into 4x4 bins was suited for the problem it originally addressed, but this is not evidence that it is appropriate for general image classification. The assumptions of the original problem domain demanded a feature that matched precisely, but an image classifier require a feature that provides greater information.

Furthermore, the information that the feature provides must be provided independent of its relationship to other features. This is explored in greater detail in Chapter 7.

5.6 Validity of Implementation

Figure 5-11



The results of Figure 5-11 provide evidence of the validity of the implementation in this paper. While the implementation is not identical to that of configuration *(n)* in [Chatfield11] (in particular, [Chatfield11] uses PHOW features, which are similar but not identical to SIFT features), the results indicate that the implementation in this paper is similar.

Note that even when using the χ^2 kernel, tuning the error parameter *C* is not useful for the basic classifier. Since the size of the training vectors (200000) is significantly larger than the number of training vectors (~5000), there is little cost associated with finding a hyperplane that separates 100% of the training data.

6 Sampling Redundancy

The goal for a sampling strategy is to maximize the mutual information of the features sampled. The most trivial approach to increase this information is to increase the number of features sampled. However, a secondary consideration is the need to minimize the redundancy of those features. By restricting the analysis to samplings of a single scale, there is enough data to develop a theory of *sampling redundancy*. Sampling redundancy is the relationship between the amount of overlap in *(x,y)* space between adjacent SIFT features in a sampling, and the amount of mutual information, as determined by classification performance.

In the case of only one sampling scale, only the pixel stride is a variable parameter. Choose a pixel stride *X* and collect features at a spacing of *X* pixels, vertically and horizontally. One can observe the redundancy of features at a single scale by increasing the sampling density (lowering *X*) and observing the diminishing returns in image classification.

In varying the single scale chosen for sampling, it is possible to observe how larger scales exhibit larger redundancy. It is clear from the results in Section 5.3 that diminishing returns for higher sample rates occur much faster at higher scales. Specifically, each scale *s* reaches a point of diminishing returns when the pixel stride reaches approximately *4s*.

This is corroborated by the results in Section 5.4. The difference in precision is negligible between a semidense sampling strategy that uses a pixel stride of *4s* and a semidense sampling strategy that uses a pixel stride of *2s*. This indicates that there is negligible total information from all scales when moving from a pixel stride of *4s* to *2s* for each scale *s*. The trend is consistent regardless of codebook size.

This point, after which diminishing returns are observed, will be referred to as the *point of redundancy* for scale *s*. There is a solid theoretical justification for the existence of a point of redundancy at exactly *4s*. A SIFT window at a given scale *s* has bins that are *4s* pixels in width and height. Moving a window fewer than *4s* pixels may not, in some cases, move any of the corners or edges described by the SIFT feature into different bins. Then, the resulting SIFT feature is quite similar. This is by design, and it is one of the reasons SIFT matching is robust in the first place. However, it also causes two samples at the same scale, whose centers differ by fewer than *4s* pixels, to be highly redundant.

Additionally, before the point of redundancy, each scale displays improved precision at an approximately linear rate with respect to the logarithm of the samples taken. There are no diminishing returns until the point of redundancy is reached. Thus the data indicate that there is negligible redundancy when the point of redundancy has not been reached, even though adjacent features will heavily overlap in *(x,y)* space (at the point of redundancy, adjacent samples overlap by 75%).

Figure 6-1

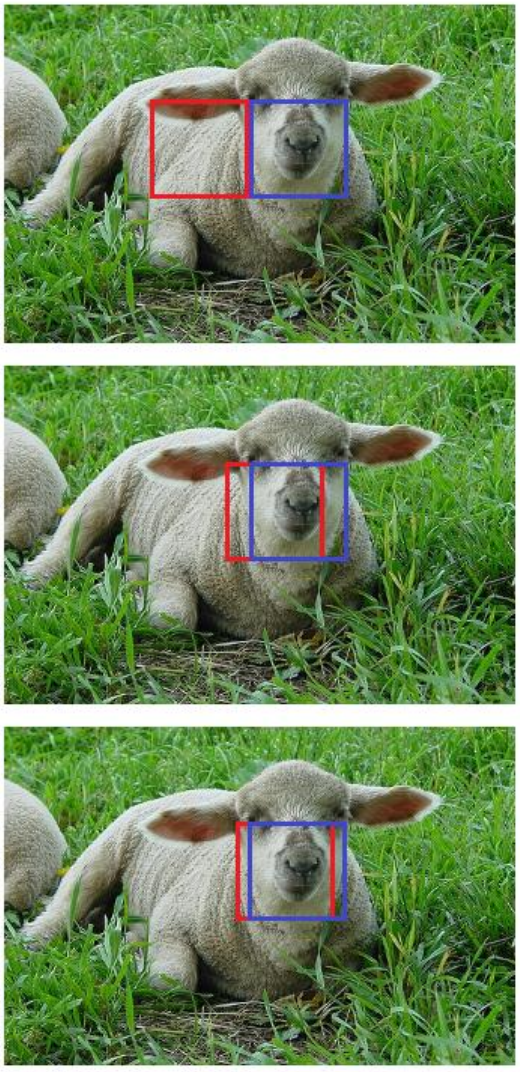


Diagram of point of redundancy. *Top*: sample windows with no overlap have negligible redundancy. *Center*: at the point of redundancy, adjacent sample windows overlap heavily, but still display negligible redundancy. *Bottom*: past the point of redundancy, pairs of adjacent features display high redundancy.

This principle also confirms that the semidense sampling strategy is efficient with respect to the sampling of multiple scales. The semidense strategy samples each scale up to the point of redundancy. While additional precision may be possible by adjusting the weighting of different scales, there is evidence that semidense sampling collects a sampling of SIFT features that is negligibly redundant.

7 Semantic Capacity

This paper will now explore a principle that places a hard limit on the potential of many state-of-the-art classifiers, and determines which modifications to a basic classifier are necessary to transcend this limit. We will call this principle *semantic capacity*.

7.1 Proof of Limitations of Additive Kernels

Consider the fundamental unit of the SVM learning algorithm, which is the kernel function. The SVM can only learn what the kernel function allows. An effective kernel function must produce large differences between most vectors of different classes and strong similarities between most vectors of the same class. It must put as much kernel space as possible between the two classes. The learning algorithm calibrates the hyperplane, but it is ultimately effective only if the kernel function translates the problem space into a kernel space where a good hyperplane exists.

All of the kernels that are popular in image classification (linear, χ^2, and histogram intersection) are *additive*. A kernel *K* is additive if it can be broken down into a function *k* on each individual variable in the vectors, and *K* is the sum of all of the *k* values. Or, formally stated:

$$\forall X = \{x_1 \dots x_n\}, \forall Y = \{y_1 \dots y_n\},$$
$$K(X, Y) = \sum_{i=1}^{n} k(x_i, y_i)$$

With an additive kernel, a vector is essentially described by the sum of its parts. There are specific mathematical limitations on what an additive kernel can distinguish between. The following proof will show these limitations.

Suppose an SVM with an additive kernel learns a model *M* described as follows:

- $\{V_1, \dots V_m\}$: a set of support vectors (SVs), each of length *n*,
- $\{w_1, \dots w_m\}$: a set of weights for each SV,
- $\{c_1, \dots c_m\}, \forall i \; c_i = \pm 1$: a set of class assignments for each SV,
- $K(X, Y) = \sum_{i=1}^{n} k(x_i, y_i)$: an additive kernel function, and
- β : the threshold associated with the hyperplane.

For any vector *X*, let

$$f(X) = \left(\sum_{i=1}^{m} w_i c_i K(X, V_i) \right) + \beta$$

Then the hyperplane of *M* is defined by the set of all points satisfying

$$\{X | f(X) = 0\}$$

Then, for any test vector *X*, we can find the classification given by the model as follows:

$$c_X = \begin{Bmatrix} 1, f(X) \geq 0 \\ -1, f(X) < 0 \end{Bmatrix}$$

Theorem: Consider four test vectors *X*, *A*, *B*, and *Y*, such that, for some distinct indices *p*, *q*, *r*, *s*, and for some values u_1, u_2, v_1, v_2,

- $X = \{x_1, \ldots x_n\}$,
- $A = \{a_1, \ldots a_n\}$, where $\forall i\ a_i = x_i$ except $a_p = x_p + u_1$ and $a_q = x_q - u_2$,
- $B = \{b_1, \ldots b_n\}$, where $\forall i\ b_i = x_i$ except $b_r = x_r + v_1$ and $b_s = x_s - v_2$, and
- $Y = \{y_1, \ldots y_n\}$, where $\forall i\ y_i = x_i$ except $y_p = a_p,\ y_q = a_q,\ y_r = b_r,$ and $y_s = b_s$.

Then for any model *M* that uses an additive kernel,

$$(c_X = -1 \wedge c_A = 1 \wedge c_B = 1) \rightarrow c_Y = 1$$

Proof: Using *M*, we find

$$\begin{aligned} f(X) &= \left(\sum_{i=1}^{m} w_i c_i K(X, V_i)\right) + \beta \\ &= \left(\sum_{i=1}^{m} w_i c_i \sum_{j=1}^{n} k(x_j, v_{ij})\right) + \beta \\ &= \left(\sum_{j=1}^{n}\sum_{i=1}^{m} w_i c_i k(x_j, v_{ij})\right) + \beta \end{aligned}$$

Similarly, we find

$$f(A) = \left(\sum_{j=1}^{n}\sum_{i=1}^{m} w_i c_i k(a_j, v_{ij})\right) + \beta$$

Since $a_j = x_j$ except when $j = p$ or $j = q$, *A* and *X* only differ at two points. It follows that

$$f(A) - f(X) = \left(\left(\sum_{j=1}^{n}\sum_{i=1}^{m} w_i c_i k(a_j, v_{ij})\right) + \beta\right) - \left(\left(\sum_{j=1}^{n}\sum_{i=1}^{m} w_i c_i k(x_j, v_{ij})\right) + \beta\right)$$

$$= \left(\sum_{j=1}^{n}\sum_{i=1}^{m} w_i c_i k(a_j, v_{ij})\right) - \left(\sum_{j=1}^{n}\sum_{i=1}^{m} w_i c_i k(x_j, v_{ij})\right)$$

$$= \left(\left(\sum_{i=1}^{m} w_i c_i k(a_p, v_{ip})\right) + \left(\sum_{i=1}^{m} w_i c_i k(a_q, v_{iq})\right)\right)$$

$$- \left(\left(\sum_{i=1}^{m} w_i c_i k(x_p, v_{ip})\right) + \left(\sum_{i=1}^{m} w_i c_i k(x_q, v_{iq})\right)\right)$$

$$= \sum_{i=1}^{m} w_i c_i \left(k(a_p, v_{ip}) + k(a_q, v_{iq}) - k(x_p, v_{ip}) - k(x_q, v_{iq})\right)$$

Symmetrically,

$$f(B) - f(X) = \sum_{i=1}^{m} w_i c_i \left(k(b_r, v_{ir}) + k(b_s, v_{is}) - k(x_r, v_{ir}) - k(x_s, v_{is})\right)$$

And, similarly, since Y differs with X at four points,

$$f(Y) - f(X) = \sum_{i=1}^{m} w_i c_i \Big(k\big(y_p, v_{ip}\big) + \ k\big(y_q, v_{iq}\big) + k(y_r, v_{ir}) + \ k(y_s, v_{is}) - k\big(x_p, v_{ip}\big) - k\big(x_q, v_{iq}\big) - k(x_r, v_{ir}) - k(x_s, v_{is}) \Big)$$

and because $y_p = a_p,\ y_q = a_q,\ y_r = b_r,$ and $y_s = b_s$, we have

$$\begin{aligned} f(Y) - f(X) &= \sum_{i=1}^{m} w_i c_i \Big(k\big(a_p, v_{ip}\big) + \ k\big(a_q, v_{iq}\big) + k(b_r, v_{ir}) + \ k(b_s, v_{is}) - k\big(x_p, v_{ip}\big) \\ &\quad - k\big(x_q, v_{iq}\big) - k(x_r, v_{ir}) - k(x_s, v_{is}) \Big) \\ &= \sum_{i=1}^{m} w_i c_i \Big(k\big(a_p, v_{ip}\big) + \ k\big(a_q, v_{iq}\big) - k\big(x_p, v_{ip}\big) - k\big(x_q, v_{iq}\big) \Big) \\ &\quad + \sum_{i=1}^{m} w_i c_i \big(k(b_r, v_{ir}) + \ k(b_s, v_{is}) - k(x_r, v_{ir}) - k(x_s, v_{is}) \big) \\ &= (f(A) - f(X)) + (f(B) - f(X)) \end{aligned}$$

Because $c_X = -1$ and $c_A = 1$, we know $(f(A) - f(X)) \geq 0$. Therefore

$$\begin{aligned} f(Y) - f(X) = \big(f(A) - f(X)\big) + \big(f(B) - f(X)\big) \Leftrightarrow \\ f(Y) = \big(f(A) - f(X)\big) + f(B) \Longrightarrow \\ f(Y) \geq f(B) \end{aligned}$$

and since $c_B = 1$,

$$c_Y = 1.$$

7.2 Context in Image Classification

The proof of the previous section demonstrates that a support vector model that uses an additive kernel is limited in what it can express. If one feature re-assignment (moving *A* to *B*) changes the vector from a negative to a positive class identification, and a second feature re-assignment (moving *C* to *D*) changes the vector from a negative to a positive class identification, then the combination of those re-assignments must change the vector from a negative to a positive class identification.

If there exist two features that are separately positive indicators, but in combination are a negative indicator, then a support vector machine with an additive kernel cannot build a correct model. *Exclusive-or* behavior cannot be modeled.

Suppose there is a need for a classifier that distinguishes between two classes: "animal" (positive) and "not animal" (negative). The classifier will search for visual features that are indicative of animals.

A feature corresponding to a lion's body would be a positive indicator. A feature corresponding to an eagle's head would be a positive indicator. However, the presence of both these features may be a negative indicator, as seen here:

Figure 7-1

Image classification where context is necessary. An eagle head and a lion body have a different meaning in combination. Each is indicative of an animal (positive) separately, but together, they indicate a gryphon (negative).

A gryphon possesses a lion body with an eagle head. One might debate whether a mythical beast is an animal, but from a functional standpoint, the correct classification would depend on the ultimate goal of the classifier. If the purpose of the "animal" classifier is to assist in identifying natural scenes, then a gryphon clearly is a negative example.

In this example, the meaning provided by two features together is very different than the sum of the separate meanings of each. Or, equivalently, the presence of one feature creates a context which influences the meaning of the second feature. Regardless of how this relationship is understood, it cannot be modeled by a support vector model that uses an additive kernel. A model with an additive kernel lacks the *semantic capacity* to model context between separate features.

Context is necessary because a standard SIFT feature does not contain enough information in and of itself to strongly indicate any object class. In the hypothetical example above, features corresponding to lion bodies and eagle heads were found. But these features would have to be expressed with a more robust feature than SIFT. SIFT features are not complex enough to accurately represent such visual structures. This can be seen in the data of Figure 5-3 which show that standard SIFT features work poorly at large scales. This is because in most images, the features found at large scales are complex structures such as bodies and heads. These structures are too complex for standard SIFT features. At lower scales, structures such as eyes, noses, beaks, or paws are more likely to be found. But these structures are visually basic enough that they are not always distinct. A lion's body is visually distinct from an eagle's body, but a lion's eye looks very similar to an eagle's eye in some images. So both eye structures might be described by a single codebook feature that corresponds to a general eye.

There are seven classes in the PASCAL VOC Challenge that have eyes. So if the classifier is trained to recognize cats, evidence of a general eye cannot be a strong positive indicator in and of itself. There are six other classes with eyes that constitute negative examples, and thus there many are more negative examples with eyes in them than there are positive examples with eyes in them. But, if features corresponding to other elements of a cat face are also present, than an eye should be a strong indicator. However, context is necessary to emulate this decision process.

Context is still useful even in cases where the features are independently strong indicators, because feature matching is an imprecise process. Even when two images contain the

same object, the features are not certain to match correctly. When a codebook is introduced to the process it becomes more difficult, because instead of matching training feature *A* directly to test feature *B*, both *A* and *B* must be matched to (hopefully) the same codebook feature. Some of the matches will be incorrect.

The original SIFT implementation resolved the ambiguity of imprecise matching by requiring geometric consistency. Simply counting the number of matched features is ineffective, but by finding multiple feature matches that share a consistent geometric relationship (that is, they occur at the correct locations and rotations), then the object can be recognized with high precision.

For reasons already clarified in Section 5.5.2, the original SIFT strategy to handle unreliable matches using geometry is not effective for the more general problem of image classification. But handling unreliable matches is even more necessary in image classification than it is in specific object recognition. Since the correspondences must be more general (match a picture of one dog to a picture of a different dog), the feature matches across images are less reliable. Adding a codebook degrades the reliability of matching further. With less reliable matches, there is a greater need to verify those matches through context.

7.3 Extending the Basic Model

The inherent assumptions of an additive kernel ignore context, but these assumptions are made to simplify the learning process. An algorithm that explores the potentially unique information of every combination of input variables is intractable. For a set of n input variables, there are 2^n unique subsets. Considering the potential information in each subset would be infeasible for 100 input variables, let alone the thousands used by state-of-the-art image classifiers.

Practically, in most problem domains, not all combinations of input variables contain additional meaning. Constructing an effective model requires that the model efficiently recognize which combinations are important and which can be ignored. But since these combinations may largely be domain-specific, it is not clear which complex kernel out of infinite possibilities is appropriate. Furthermore, the SVM optimization process is already limited by the time complexity of computing the kernel matrix (for all kernels covered in this paper, computing the kernel matrix requires $O(m^2n)$ time, where m is the number of training examples and n is the

length of the input vector.). Increasing this time complexity by even one degree of n to $O(m^2n^2)$ makes the SVM the slowest step in the image classifier, by a large factor.

Finding the right model to address arbitrary complexity is a problem that applies not just to support vector machines, but essentially any learning model. Consider neural networks. A single-layer network with the standard perceptron can only model functions that are a linear combination of the inputs. It cannot encode "exclusive-or" behavior, or any other function in which a combination of two variables holds information greater than the sum of the parts. A two-layer network can theoretically model any function $\{0,1\}^n \rightarrow \{0,1\}^m$. But such a two-layer network must consider the potentially unique information of every combination of input variables, and thus may require a number of perceptrons that is exponential with respect to the number of input variables.

Because the combinations of training variables that are relevant are largely domain-specific, it is not the responsibility of the learning algorithm to find them. If this responsibility is not placed on the learning algorithm, then it is necessary that the input vectors passed in to the learning algorithm contain variables with significant information independent of the other variables.

To do this, one must process the input vectors. If the input vectors that comprise the training data are properly processed, the task of choosing a kernel is simplified. Specifically, if each variable in the input vector has significant information, independent of context, then a basic kernel function can put those variables together into a cohesive model.

Thus the problems pertaining to context can be addressed before the learning phase. To achieve this, the variables in the input to the learning algorithm must represent either:

1. semantic structures that combine multiple SIFT features in a *non-linear* fashion, or
2. features other than SIFT that are less context-dependent than SIFT features.

7.3.1 Feature Complexity

Section 5.5 shows that even a basic modification to SIFT (increasing the number of bins) produces a superior feature for use in a basic classifier. This is because the modified feature allows more complex pixel patterns to be approximated, with more edges and more corners. Even though it is less robust to common image deformations, the feature possesses additional complexity. A feature that represents an entire object or a larger piece of an object does not

require as much contextual information as a feature that represents a smaller piece of an object. Thus, when this more complex feature is used with an additive kernel, which cannot represent context, the resulting model has fewer theoretical limitations.

The basic modification to SIFT features, SIFT8, in Section 5.5 increases the complexity, but at the cost of the robustness to common image deformations. This makes the feature more difficult to match, mitigating some of the benefit gained from the increased complexity.

If a feature exists that can describe visual structures of the same complexity as a SIFT8 feature, but is more robust to common image deformations, then it will be superior. While the theoretical limitations given by Section 7.1 will be similar, in practice, a feature that matches more accurately and provides the same information will provide higher classification precision in an image classifier.

7.3.2 Deformable Parts Models

Deformable parts models are the clearest example in the literature of a feature that is less context-dependent than SIFT. This is achieved with an additional layer of semantic construction, in which smaller SIFT-like features are combined with a non-rigid geometric grammar to form a larger feature. The larger feature is intended to describe the whole object and thus context is less necessary.

One limitation of deformable parts models is that they must be constructed *a priori* for each specific object class. There is presently no algorithm to learn a deformable parts model for an arbitrary class. Constructing multiple deformable parts models, one for each specific instance of an object, is also potentially useful, as presented by one of the co-winners of the PASCAL VOC Challenge in 2011. [Everingham12] This, however, takes even longer, and is even more dependent on sufficient training data. An algorithm to learn a deformable parts model for an arbitrary image would make the approach more independent of human intervention.

7.3.3 Encoding schemes

A basic classifier encodes the sampled features by hard assignment to the nearest word in a visual codebook. However, other encoding schemes exist. A survey of them is explored in [Chatfield11]. If an additive kernel is used, the vector that results from the encoding scheme still has the same limitation given by Section 7.1. However, if the encoding scheme creates a vector where each variable has more information free of context, the classifier has higher potential.

8 Conclusion

This paper has examined the various parameters that can be adjusted in a basic dense SIFT sampling image classifier. It has examined rigorously many of the decisions that have traditionally been made without supporting evidence.

The data provide evidence of improved strategies for parameter selection in a basic classifier. Some of the results conflict with conventional wisdom in the field.

This paper has also developed two novel concepts, *sampling redundancy* and *semantic capacity*, which explain the data and provide insight into the underlying structure of the image classification problem. Sampling redundancy is a theory that describes mathematically the mutual information provided by two sampled features which overlap in the image space. This is useful for constructing an optimal dense feature sampling. Semantic capacity mathematically clarifies the key limitation of a basic image classifier (the inability to model context). This is useful to direct future research, by explaining which types of extensions to the basic model can give the model fundamentally greater potential.

References

A. Bosch, A. Zisserman, and X. Munoz, "Image Classification using Random Forests and Ferns," in IEEE 11th International Conference on Computer Vision, 2007, pp.1-8, 14-21.

C. Chang and C. Lin, "LIBSVM: A library for support vector machines," ACM Transactions on Intelligent Systems and Technology, 2011, vol. 2, pp. 1-27.

K. Chatfield, V. Lempitsky, A. Vedaldi, and A. Zisserman. “The devil is in the details: an evaluation of recent feature encoding methods,” in Proc. BMVC, 2011, pp. 76.1-76.12.

D. Comaniciu and P. Meer, "Mean shift analysis and applications," in the Proceedings of the Seventh IEEE International Conference on Computer Vision, 1999, vol. 2, pp. 1197-1203.

C. Cortes and V. Vapnik, "Support-vector networks," Machine Learning, 1995, vol. 20, pp. 273-297.

G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray, "Visual categorization with bags of keypoints," in ECCV International Workshop on Statistical Learning in Computer Vision, 2004.

N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005, vol. 1., pp. 886-893.

M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. “The PASCAL Visual Object Classes Homepage”, 2012, http://pascallin.ecs.soton.ac.uk/challenges/VOC/.

P. Felzenszwalb, D. McAllester and D. Ramanan, "A discriminatively trained, multiscale, deformable part model," in IEEE Conference on Computer Vision and Pattern Recognition, 2008, pp. 1-8.

P. Felzenszwalb and D. Huttenlocher, "Efficient Graph-Based Image Segmentation," International Journal of Computer Vision, 2004, vol. 59, pp. 167-181.

K. Fukunaga and L. Hostetler, "The estimation of the gradient of a density function, with applications in pattern recognition," in IEEE Transactions on Information Theory, 1975, vol. 21, pp. 32-40.

J. Hou, Z. Feng, Y. Yang and N. Qi, "Towards a universal and limited visual vocabulary," in Advances in Visual Computing, G. Bebis, R. Boyle, B. Parvin, D. Koracin, S. Wang, K. Kyungnam, B. Benes, K. Moreland, C. Borst, S. DiVerdi, C. Yi-Jen and J. Ming, Eds. Springer Berlin / Heidelberg, 2011, pp. 398-407.

T. Joachims, "Making Large-Scale SVM Learning Practical", in Advances in Kernel Methods - Support Vector Learning, B. Schölkopf, C. Burges, and A. Smola, Eds. MIT-Press, 1999.

Jianguo Zhang, M. Marszalek, S. Lazebnik and C. Schmid, "Local features and kernels for classification of texture and object categories: A comprehensive study," in Conference on Computer Vision and Pattern Recognition Workshop, 2006, pp. 13.

F. Jurie and B. Triggs, "Creating efficient codebooks for visual recognition," in Tenth IEEE International Conference on Computer Vision, 2005, vol. 1., pp. 604-610.

S. Lazebnik, C. Schmid and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2006, pp. 2169-2178.

Liu Yang, Rong Jin, R. Sukthankar and F. Jurie, "Unifying discriminative visual codebook generation with classifier training for object category recognition," in IEEE Conference on Computer Vision and Pattern Recognition, 2008, pp. 1-8.

D. G. Lowe, "Object recognition from local scale-invariant features," in the Proceedings of the Seventh IEEE International Conference on Computer Vision, 1999, vol. 2, pp. 1150-1157.

S. Maji, A. C. Berg, and J. Malik. "Efficient Classification for Additive Kernel SVMs," to appear in IEEE Transactions on Pattern Analysis and Machine Intelligence, 2012.

K. Mikolajczyk and C. Schmid, "Scale & Affine Invariant Interest Point Detectors," International Journal of Computer Vision, 2004, vol. 60, pp. 63-86.

F. Moosmann, B. Triggs and F. Jurie, "Fast discriminative visual codebooks using randomized clustering forests," Advances in Neural Information Processing Systems, 2007, vol. 19.

J. M. Morel and G. Yu, "ASIFT: A new framework for fully affine invariant image comparison," SIAM Journal on Imaging Sciences, 2009, vol. 2, pp. 438.

D. Nister and H. Stewenius, "Scalable recognition with a vocabulary tree," in IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2006, pp. 2161-2168.

E. Nowak, F. Jurie and B. Triggs, "Sampling strategies for bag-of-features image classification," in Computer Vision – ECCV 2006, A. Leonardis, H. Bischof and A. Pinz, Eds. Springer Berlin / Heidelberg, 2006, pp. 490-503.

T. Ojala, M. Pietikainen and D. Harwood, "Performance evaluation of texture measures with classification based on kullback discrimination of distributions," in Proceedings of the 12th IAPR International Conference on Pattern Recognition, 1994, vol. 1 - Conference A: Computer Vision & Image Processing, pp. 582-585.

A. Oliva and A. Torralba, "Modeling the Shape of the Scene: A Holistic Representation of the Spatial Envelope," International Journal of Computer Vision, 2001, vol. 42, pp. 145-175.

W. Pace, "K-Means Example", 2007, http://en.wikipedia.org/wiki/K-means_clustering.

X. Ren and J. Malik, "Learning a classification model for segmentation," in Proceedings of the Ninth IEEE International Conference on Computer Vision, 2003, vol. 1., pp. 10-17.

K. E. A. van de Sande, T. Gevers and C. G. M. Snoek, "Evaluating Color Descriptors for Object and Scene Recognition," in IEEE Transactions on Pattern Analysis and Machine Intelligence, 2010, vol. 32, pp. 1582-1596.

J. van de Weijer and C. Schmid, "Coloring local feature extraction," in Computer Vision – ECCV 2006, A. Leonardis, H. Bischof and A. Pinz, Eds. Springer Berlin / Heidelberg, 2006, pp. 334-348.

M. Varma and D. Ray, "Learning the discriminative power-invariance trade-off," in IEEE 11th International Conference on Computer Vision, 2007, pp. 1-8.

A. Vedaldi, V. Gulshan, M. Varma and A. Zisserman, "Multiple kernels for object detection," in IEEE 12th International Conference on Computer Vision, 2009, pp. 606-613.

A. Vedaldi and A. Zisserman, "Efficient additive kernels via explicit feature maps," in IEEE Conference on Computer Vision and Pattern Recognition, 2010.

X. Wang, T. X. Han and S. Yan, "An HOG-LBP human detector with partial occlusion handling," in IEEE 12th International Conference on Computer Vision, 2009, pp. 32-39.

C. Wu, "SiftGPU: A GPU Implementation of Scale Invariant Feature Transform (SIFT)," 2007.

J. Wu, W.-C. Tan, J. M. Rehg, "Efficient and Effective Visual Codebook Generation Using Additive Kernels," Journal of Machine Learning Research, 2011, vol. 12(Nov), pp. 3097-3118.

Appendix A Sample Images

To explain the PASCAL VOC Challenge, this appendix contains example images from all twenty object classes. All images are taken from the PASCAL VOC2011 training data. A pair of images representing typical positive examples is shown for each class. These images were found by selecting the first two relevant images that occur in the training data, which, given that there is no ordering to the training data, makes them random. A second pair of images representing difficult positive examples is shown for each class. These images were hand-picked to illustrate the wide variety of difficult images to be classified in the PASCAL VOC Challenge.

Aeroplane (Typical)

Planes are consistently classified at the highest average precision of all 20 classes in the PASCAL VOC Challenge. The stringent requirements of their function result in a design that is structurally similar. Thus, many planes look very similar. The same cannot be said of the other object classes.

Aeroplane (Difficult)

Planes can still be challenging to classify. Pictures taken of an airborne plane will be at a large distance, and thus the plane will appear at a small scale. Head-on viewing angles tend to be less common and more distinctive than side angles.

Bicycle (Typical)

Pictures of bicycles in the PASCAL image data sets vary between pictures of stationary bicycles, and pictures of people riding bicycles. The wheels (thin, round) are the most consistent features across different instances of the class.

Bicycles (Difficult)

Bicycles frequently appear in an outdoor context, so a violation of that context can be potentially confusing to a classifier. Bicycles are light and thin by design. This means that while they may constitute a large portion of the image in terms of a bounding box (top image), many of the pixels in that bounding box are background pixels that essentially inject noise into the SIFT representation. Or, if the bicycle is viewed head-on (bottom image), it will represent a very small portion of the image.

Bird (Typical)

Birds can potentially be identified by their beaks or by their wings. Additionally, context (aerial scenes, trees) can be useful.

Birds (Difficult)

Some birds blend in very well with their surroundings. Even when this is not the case, severe occlusions can arise due to their tendency to be found in trees, and thus be occluded by branches.

Boats (Typical)

Boats reliably occur in the context of water. Many boats have tall, thin vertical masts.

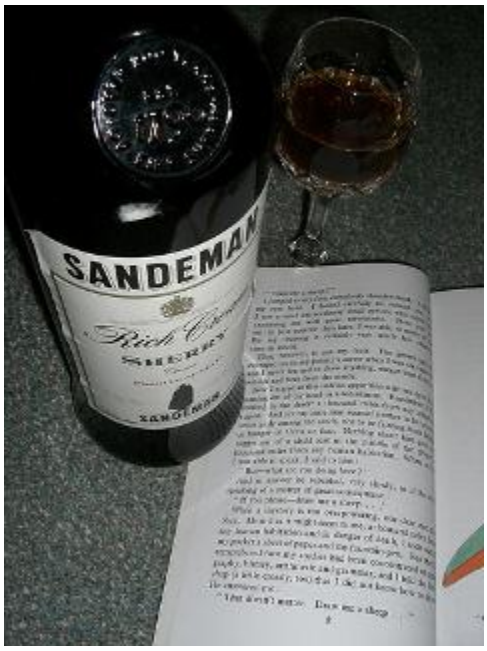
Boats (Difficult)

Sizes of different boats vary widely, and different sizes of boats tend to possess distinct visual features. The bottom image shows a boat that is not presented at an uncommon angle, but is removed from its usual context. It is unlikely to be classified correctly.

Bottle (Typical)

Bottles are defined by their function: to hold liquid. However, since the liquid conforms to the bottle, there is little restriction on the potential shape of the bottle. Furthermore, bottles are commonly either partially transparent, or reflective, each of which results in difficult image deformations. Each year, “bottle” is the class in the PASCAL VOC Challenge that is classified with the lowest average precision by top competitors.

Bottle (Difficult)



While most bottle images are difficult to classify, some are especially difficult because they do not show the full shape of the bottle (top image). Some bottles are less curved and more angular, but there are fewer examples of these, making them harder to match (bottom image).

Bus (Typical)

Buses frequently have a box-like shape. Context (streets) and wheels are indicators of buses but do not necessarily discriminate them from other vehicles, particularly cars.

Bus (Difficult)

All objects with glass can be reflective, generating distorted features with unclear meaning. Some images of buses are taken from close enough that the entire shape cannot be observed, but rather only inferred.

Car (Typical)

Like buses, wheels and context (streets) can be indicative of cars. Cars can potentially be discriminated by their sleeker shapes. Headlights, rearview mirrors, and other features can sometimes be matched across different kinds of cars.

Cars (Difficult)

While SIFT is illumination-invariant, at least some light is needed to detect gradients. Very dark images (top) are challenging regardless of object class. In the bottom image, there is a wheel present, indicating a bus or car. However, it must be inferred from context that a car is the more likely object.

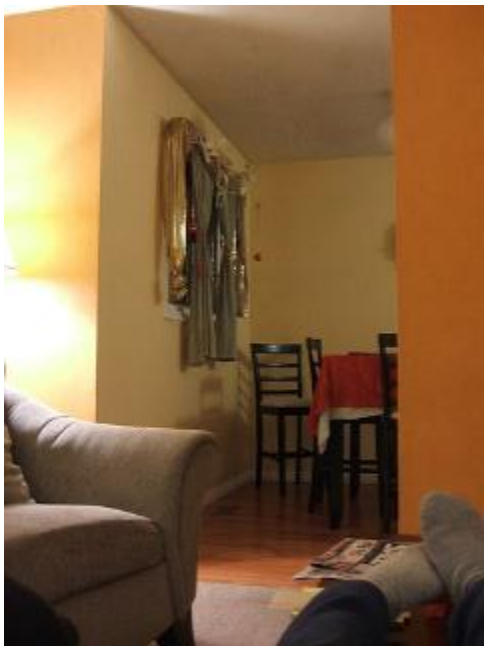
Cat (Typical)

Like all animals, cats have distinctive faces and body shapes. Not all images present both types of features.

Cat (Difficult)

Pictures of cats can be taken when they are in motion (top image), resulting in image deformations due to the camera. Dark images (bottom) are difficult for cats. Smaller features such as facial features may not be present, and even larger shape features may be obfuscated.

Chair (Typical)



The shape of chairs can vary significantly. They do reliably occur in an indoor context.

Chair (Difficult)

Chairs are commonly found near tables or desks. Thus they may be largely occluded from certain angles.

Cow (Typical)

Cow (Difficult)

When an object occurs in the "wrong" context in the test data (top image), it is less likely to be matched. The car features present here would negatively correlate with positive cow classification. If the object occurs in the "wrong" context in the training data, the consequences can be worse, because the model conflates the features of cows and cars. Then, the model will potentially determine car features to be positive indicators of cows.

In the bottom image, it is difficult to recognize exactly what animal is present, even for a human.

Dining Table (Typical)

Dining tables generally occur in an indoor context. They frequently have tall, thin legs, but this is not universal.

Dining Table (Difficult)

Sometimes the presence of a dining table needs to be heavily inferred from context. Nearby chairs or food on top may be more indicative than any features of the table itself.

Dog (Typical)

Dogs occur in a variety of contexts. They have distinct faces and bodies but do share some similarities with cats.

Dog (Difficult)

The dog in the top image blends in heavily with the background. The bottom image presents an unusual viewpoint. Images of animals may potentially display them lying down, but such images are infrequent and thus difficult to match to other examples.

Horse (Typical)

Horses have a distinct gait and identifiable long faces.

Horse (Difficult)

The horse in the top image is heavily occluded. In the bottom image, recognition from the face may be effective, but since there are few images of horses lying down, body features will not match well.

Motorcycle (Typical)

Motorcycle wheels have the potential to be confused with other vehicle wheels, but they are somewhat distinct. Other features such as headlights, handlebars, and seats can potentially be matched as well.

Motorcycle (Difficult)

Motorcycles look very different from a back or head-on view, and, like most objects, are difficult to recognize in the presence of severe occlusion or unique viewpoints.

Person (Typical)

People are the most commonly occurring objects in PASCAL VOC Challenge image data. This is because many images that represent another object class also include people. Faces frequently constitute a large portion of the image and thus are recognizable.

Person (Difficult)

People can occur in a wide variety of poses, which makes using context potentially important. However, people also occur in many settings, in the presence of many different objects. This makes using context potentially difficult.

Potted Plant (Typical)

"Potted plant" is one of the most difficult object classes in the PASCAL VOC. Part of the difficulty is that this object class is limited only to potted plants and not all plants. This makes the visual information of the plant itself much less likely to be useful. The information of the pot is potentially more salient. This distinction (potted plants only) also makes an indoor context more likely.

Potted Plant (Difficult)

As “plant” is a very general class, the visual information associated with plants varies greatly. As seen in the bottom image, even plants in containers that are not pots are considered potted plants. But, this results in a very general “container” class. However, neither class is sufficient to indicate a potted plant on its own, so some sort of context information is necessary.

Sheep (Typical)

Sheep, like all other animals, have distinct bodies and faces, but confusion between animal classes is still common. Sheep can frequently be confused with cows.

Sheep (Difficult)

From challenging viewpoints or at small scales, there are few distinguishing features for a sheep.

Sofa (Typical)

Sofas are a difficult object class because even when they resemble the “standard” shape, as in these images, that shape is not highly distinct. An indoor context is helpful.

Sofa (Difficult)

When in use, sofas are largely occluded, and it is difficult to extract even shape information.

Train (Typical)

Trains have a similar shape to buses in many images. However, they possess some distinguishing features, and rails, while not strictly part of the train, can be indicative.

Train (Difficult)

Like planes, trains are large enough objects that they can potentially be recognized at a wide variety of scales. However, the distinguishing features vary widely based on the scale at which the train is presented.

TV or Monitor (Typical)

Monitors, and to a lesser extent TVs, have a well-defined box shape from the front. They tend to be uniform in color as well.

TV or Monitor (Difficult)

The simple shape of monitors and TVs frequently makes it difficult to pick out of the background (top image). Side views (bottom image) are less uniform in appearance and less common in the training data.

Appendix B Single Scale Samples

To illustrate the difference in small and large scale features, sample images were taken based on the data presented in Section 5.3. Three object classes (bird, horse, and train) are considered, and two images are presented for each class.

The images were selected by comparing two single-scale classifiers from Figure 5-3 (*s=1* at 4 pixel stride, and *s=4* at 4 pixel stride). The two images chosen for each class are the positive test images with the greatest disparity in classification between the two single-scale classifiers. So, the first image in each section is an image that the *s=1* (low-scale) classifier recognized, but the *s=4* (high scale) classifier did poorly with. The second image is an image that the high-scale classifier recognized, but the low-scale classifier did poorly with.

Bird

Low-scale features:

High-scale features:

The facial features of the bird appear useful. When such features take up a large portion of the image (bottom image), the high-scale classifier is more successful.

Horse

Low-scale features:

High-scale features:

Low-scale facial features are best presented from a front view (top image). Larger features that pertain to head and body shape are best presented from a side view (bottom image).

Train

Low-scale features:

High-scale features:

Trains do not appear to have a distinct shape, at least with respect to SIFT features. As such, the high-scale classifier does better only in images with very close viewpoints (bottom image). The high-scale classifier does poorly with trains, with a mean average precision of .410 versus .551 for the low-scale classifier.