Similar-Part Approximation Using Invariant Feature Descriptors

Sarah Marqusee Sachs

Advisor: Pedro Felzenszwalb

Abstract

This thesis describes the implementation of a new algorithm for identifying similar parts in multiple images. Visual words of Scale Invariant Feature Transform descriptors are used to describe parts of images and Locality Sensitive Hashing is utilized for quick similar-part lookup. This algorithm proves very effective on identical items of different sizes and rotations, and moderately effective on similar yet unidentical items of the same class. Suggestions are made for apply this approximation to building object grammars. Further exploration on feature descriptors and distance functions are needed for better performance accuracy.

Introduction

This paper presents an approach for estimating similar parts in a set of images. Imagine you want to search Google Images for a certain object, but you cannot think of the name of that object. Instead, you might have ten other photographs that all happen to contain that object somewhere in the image – the images have different backgrounds and may have other things in the foreground, but nonetheless share this object in common. The search engine should recognize that the images all share your unnamed object in common. Consequently, the search engine should be focusing its search on that particular object. The object that each image contains can be thought of as the most similar part among all the images.

For example, imagine that you forget the name of President Barack Obama, but you provide the search engine with the ten photographs in Figure 0.1a on the following page. It is clear to the human eye that Barack Obama is in all ten images. However, the images also share other things. Multiple images have an American flag and multiple have repeating characters like Malia Obama or Hillary Clinton. Google Images might return the search query with images of Hillary Clinton without Barack Obama, which would be an incorrect identification of the similar part. We want to develop an algorithm that can realize, without any training from humans or manual image annotation, that Barack

(a) Selection of photographs provided to the algorithm

(b) Yellow boxes indicate the parts of the images that the algorithm should select as being similar parts

Figure 0.1: Sample of photos and their ideal similar-part selection

Obama is the most common thing in all of these images. If our algorithm can determine that the most similar part in these photos is Barack Obama's face, we have enough information to do an informed and accurate image search for things that are similar to Barack Obama's face. An ideal output of the algorithm is illustrated in Figure 0.1b on the previous page, where the highlighted boxes represent which part of each image the algorithm should choose as the similar part.

The ability to recognize a shared object in multiple images has importance for many areas of research. Consider the field of research that focuses on detecting humans in images. Research has already been done in utilizing the parts of an object to improve human detection in images, for example in the work of Girshick, Felzenszwalb, and McAllester [1]. In their study, if the common parts of a person are already known (forming an "object grammar" for the structure of a person), then an algorithm can have more information on how to detect a person in an image and can perform with significantly better accuracy.

However, work on person detection with object grammars currently relies on manually created grammars for objects. Experts in this field have noted that "automatically learning the structure of grammar models remains a significant challenge for future work."[1] If we can estimate similar parts in images automatically without human intervention, then we can estimate multiple similar parts in images. This thesis presents an efficient implementation of this automatic estimation.

To visualize how this estimation can automate grammar modeling, consider the following example. The most similar part among a series of photographs of humans may be a head. A similar-part approximation algorithm may then recognize that the legs in each image are the second most similar part. Third, the algorithm may notice that a long torso is a similar part among the images of humans. Eventually, the algorithm will have noticed a series of similar parts and has enough information to build an object grammar and understand the parts of a human body. Note that this process needs no human annotation or involvement. Hypothetically, the same process could be done on other objects, such as trees, dogs, cars, etc. We now have a process to create object grammars and detect almost any item, provided that the approximation algorithm is run on many photos of that item.

This work in person detection using object grammars is particularly relevant today as we advance the technology for self-driving cars. In order for self-driving cars to safely navigate roads, we need to be confident in their pedestrian detection systems. The real-time algorithms that provide a self-driving car with environmental information should be well-equipped to recognize if something is a dog, a human, a tree, or a car. This information is crucial for safe navigation, as a tree has different movement patterns than a pedestrian or a dog. Part-based models of detection have proved to be paramount in the development of robust models for pedestrian detection systems in advanced driver assistance systems (like self-driving cars or autonomous parallel parking cars) [2, 3]. The work of this thesis provides a first step into a large-scale process of automating the grammars of these models to make comprehensive detection systems.

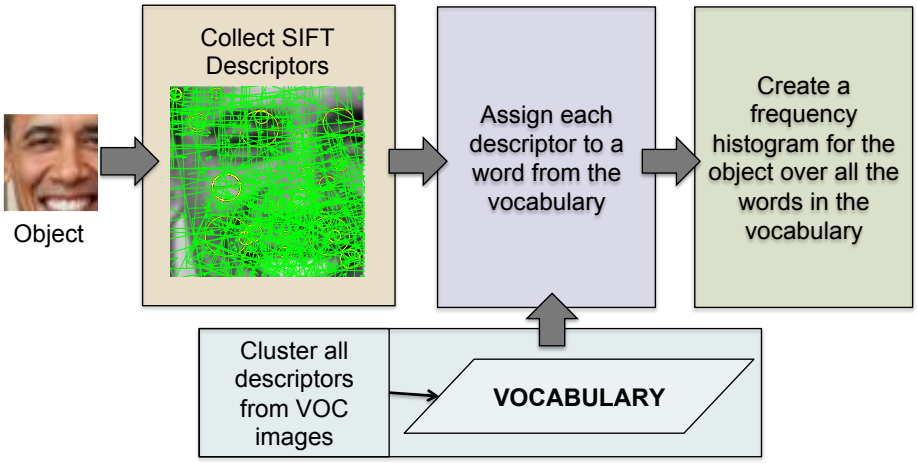

Figure 0.2: Overview of the object description process. An object is described using SIFT descriptor matrices (which are visualized with green grids). Each descriptor is assigned to the closest word in a predetermined vocabulary, which is created over a large set of images. The collection of words used to describe the object is transformed into a word frequency histogram.

From a theoretical perspective, part similarity is measured by minimizing an objective function over pairwise costs. This thesis presents an efficient implementation of an approximation of this algorithm, developed by Felzenszwalb [4], which estimates the optimal solution to the objective without calculating all pairwise costs. Through this approximation, the algorithm is able to approach the problem at large scales.

However, in order to even compare the similarity of objects in different images, a shared "vocabulary" needs to be developed that can provide consistent measures of similarity and difference. For instance, just looking at the pixel values of an image will not provide information about the key curves and textures in the image that describe the objects. Moreover, objects that only vary in rotation, illumination, and size should still be viewed as extremely similar. In Section4, this thesis explores the use of SIFT feature descriptors to provide this invariant description of objects.

Once all of the objects in an image can be described using these SIFT descriptors (which are expressed as matrices), we need to develop a system for measuring similarity among descriptors in all images. This provides the need to simplify our descriptors to a fixed and limited set. We can think about this by providing an analogy where descriptors aren't matrices but words in the English language. Assume we have two images, each with one person in the image. In one image we describe the person's hair as "platinum" and in another image we describe the person's hair as "golden". We can assume these two descriptors are describing the same color and we want an increased measure of similarity

between the people in both images. However, if we describe one person's hair as "platinum" and the other person's hair as "brunette", we have less confidence in their similarity. We must make a system that recognizes "platinum" and "golden" as the same descriptor but "platinum" and "brunette" as different. In the case of this analogy, if we only had the words "blonde" and "brown" in our vocabulary, it would be much easier to determine if the people's hair colors are similar or different because there would be no ambiguity among synonyms. It is important to note that we may also lose some descriptive ability, but we gain the ability to make consistent comparisons. This example illustrates the need to create a limited vocabulary of feature descriptors so we can better discriminate object similarity. The creation of this vocabulary is described in Section 5.1 on page 12.

If we count how often each vocabulary word is used to describe a given object, the object can be described with a word-frequency histogram. An overview of this process is defined in Figure 0.2 on the previous page and described in much greater detail in Part II. These histograms are what we will use to determine the similarity of objects. Histograms with a small l_1 distance from each other imply similarity, while histograms with a large l_1 distance imply stark difference.

This thesis explains the implementation of an approximation algorithm to find similar parts. First by breaking images down into subpatches (Section 3), extracting SIFT descriptors from the subpatches (Section 4), describing subpatches by a set of visual words (Section 5), and then approximating the optimal solution using LSH nearest neighbor search (Section 6) and star-graph objectives (Section 2).

Part I
Algorithm overview

1 Objective

Consider a set of I items with some distance metric d that measures the distance between two items. Let $B_1, ..., B_k$ be K subsets of I. We will refer to these subsets as "bags" of items. It is possible, but not necessary, that $B_j \cap B_k \neq \emptyset$, $j \neq k$. The motivation for this thesis is to find a set of elements $x = (x_1, ..., x_k)$ where $x_i \in B_i$ and the elements of x are as similar as possible. We formalize our objective as the sum of pairwise distances among selected items using our distance metric, d. Let $V = \{1, ..., K\}$. $V \times V$ is expressed as V^2. The objective is defined as:

$$c(x) = \sum_{(i,j) \in V^2} d(x_i, x_j) \ . \tag{1.1}$$

Let $x^* = \text{argmin}_x c(x)$ be the optimal solution to this optimization. Brute force computation of argmin over all possible x's is intractable. The cost of calculating $c(x)$ for any given x is $O(k^2)$. Let X be the set of all feasible x's.

If each subset B_i has at most N items, then $|X| = N^K$. Thus, the cost of calculating $c(x)$ for every $x \in X$ is $O(N^K \cdot K^2)$. This slow runtime motivates an approximation of our objective.

2 Approximation

2.1 Overview of Approximation

An optimization of $c(x)$ is difficult when all pairwise distances must be calculated. Instead we consider a tractable approximation proposed by Felzenszwalb [4]. Consider the same space of possible solutions to the initial optimization, but with K "star-graph" objective functions.

For $r \in V$ we define the "star-graph" objective as

$$c^r(x) = \sum_{j \in V \setminus \{r\}} d(x_r, x_j) \ . \tag{2.1}$$

Let $x^r = \text{argmin}_x c^r(x)$ be the optimal solution for this optimization problem. Using dynamic programming, $c^r(x)$ can be solved efficiently by first computing x_r^r and then solving for x_j^r:

$$x_r^r = \underset{x_r \in B_r}{\text{argmin}} \sum_{j \in V \setminus \{r\}} \min_{x_j \in B_j} d(x_r, x_j) \tag{2.2}$$

$$x_j^r = \underset{x_j \in B_j}{\text{argmin}} \ d(x_r, x_j) \ . \tag{2.3}$$

After calculating x^r for $1 \leq r \leq K$ we have $x^1, \ldots, x^k$ that represent K solutions to the "star-graph" objective. Each solution, described by the items in x^i, contains an item x_r^i that has the smallest distance to items in the other bags. Also in x^i are the $K-1$ items, each being the closest to x_r^i in their respective bags. Each collection of items x^i leads to a possible solution to our initial objective $c(x)$. Thus, we can choose among $x^1, \ldots, x^k$ by defining our approximate solution as

$$\hat{x} = \underset{x^r}{\text{argmin}} c(x^r) \quad 1 \leq r \leq K \ .$$

The initial objective pairwise cost function is only calculated K times, as opposed to the brute force solution to the objective 1.1 which calculates $c(x)$ N^K times. Pseudocode for an implementation of this approximation can be seen in Algorithm 1 on the following page. A deeper analysis of the runtime of this approximation can be found in Section 2.3.

2.2 Justification of Approximation

Approximating the initial objective has little use unless the approximation's solution is bounded by the objective's optimal solution. An unbounded approximation cannot be used confidently in practice. However, as described by

Algorithm 1 Similar item approximation

Input: K bags $B_1, \ldots, B_k$, each with up to n items
Output: Graph of similar items, $\hat{x} = x_1, \ldots, x_k$ where $x_i \in B_i$

for K bags, B_i, $i = 1 : K$ **do**
  $minCost = \infty$ and $minStar = null$
  for each of N items in bag, x_j^i, $j = 1 : N$ **do**
    $c^r(x_j^i) = 0$
    $star_j^i \in \mathbb{R}^K$, $star_j^i(j) = x_j^i$
    for K bags, B_m, $m = 1 : K$ **do**
      find $x_{closest}^m \in B_m$ that minimizes $d(x_{closest}^m, x_j^i)$
      add $d(x_{closest}^m, x_j^i)$ to $c^r(x_j^i)$
    end for
    if $c^r(x_j^i) < minCost$ **then**
      $minCost = c^r(x_j^i)$ and $minStar = star_j^i$
    end if
  end for
  set the min-star for B_i such that $x^i = minStar$
end for
$\hat{x} = null$
$c(\hat{x}) = \infty$
for K bags, B_i, $i = 1 : K$ **do**
  $c(x^i) = 0$
  for item x_j^i, $j = 1 : K$ **do**
    for item x_k^i, $k = j : K$ **do**
      $c(x^i) = c(x^i) + d(x_j^i, x_k^i)$
    end for
  end for
  if $c(x^i) < c(\hat{x})$ **then**
    $\hat{x} = x^i$
    $c(\hat{x}) = c(x^i)$
  end if
end for
return $\hat{x}$

Felzenszwalb [4], this approximation objective's optimal solution is no more than twice as large as the original objective's optimal solution.

Theorem 1. *This approximation of the objective leads to a 2-approximation algorithm s.t.*

$$c(\hat{x}) \leq 2c(x^*)$$

where $c(x)$ is described by 1.1, x^* is the optimal solution found by $argmin_x c(x)$, and $\hat{x}$ is the approximation described in Section 2.1.

Proof.

Since the minimum of a set is as most the average,

$$c(\hat{x}) = \min_{x^r} c(x^r) \leq \frac{1}{N} \sum_{r \in V} c(x^r) \ .$$

And by the triangle inequality, for any x^r,

$$c(x^r) = \sum_{j,k \in V^2} d(x^r_{j,} x^r_k) \leq \sum_{j,k \in V^2} (d(x^r_r, x^r_j) + d(x^r_r, x^r_k))$$

$$= \sum_{j \in V} \sum_{k \in V} (d(x^r_r, x^r_j) + d(x^r_r, x^r_k))$$

$$= \sum_{j \in V} \sum_{k \in V} d(x^r_r, x^r_j) + \sum_{j \in V} \sum_{k \in V} d(x^r_r, x^r_k)$$

$$= N \cdot \sum_{j \in V} d(x^r_r, x^r_j) + N \cdot \sum_{k \in V} d(x^r_r, x^r_k)$$

$$= 2N \cdot \sum_{j \in V} d(x^r_r, x^r_j)$$

$$= 2N \cdot c^r(x^r) \ .$$

Thus, $c(x^r) \leq 2N \cdot c^r(x^r)$.
Therefore,

$$\frac{1}{N} \sum_{r \in V} c(x^r) \leq 2 \sum_{r \in V} c^r(x^r)$$

and we can conclude that

$$c(\hat{x}) \leq 2 \sum_{r \in V} c^r(x^r) \ .$$

Given the dynamic programming solution used to compute x^r

$$c^r(x^r) = \min_{x_r \in B_r} \sum_{j \in V} \min_{x_j \in B_j} d(x_r, x_j)$$

we can simplify our inequality into

$$c(\hat{x}) \leq 2 \cdot \sum_{r \in V} \min_{x_r \in B_r} \sum_{j \in V} \min_{x_j \in B_j} d(x_r, x_j)$$

$$\leq 2 \cdot \min_{\bar{x}} \sum_{r \in V} \min_{x_r \in B_r} \sum_{j \in V} d(x_r, \bar{x}_j)$$

$$\leq 2 \cdot \min_{\bar{x}} \sum_{r \in V} \sum_{j \in V} d(\bar{x}_r, \bar{x}_j)$$

$$= 2 \cdot \min_{\bar{x}} \sum_{r,j \in V^2} d(\bar{x}_r, \bar{x}_j)$$

$$= 2 \cdot c(x^*) \ .$$

Thus, $c(\hat{x}) \leq 2 \cdot c(x^*)$. □

2.3 Runtime Analysis

Consider the runtime for this approximation. Assume that upon implementation, items are represented by vectors of length J (see Section 5 on page 12).

For K bags, a star is formed for each of the N items in each bag. In order to form that star, each item must look at $K-1$ other bags, find the closest item in each bag (which is done in a constant amount of time, see Section 6), find the current item's vector's distance to that closest item's vector of length J, and add that distance to the cost of that star. This process will end with a collection of possible approximations, $x_1, \ldots, x_k$. Thus, the process of calculating these K solutions takes $K^2 \cdot N \cdot J$ time.

Now consider the process of choosing the best approximation, $\hat{x}$, that has the lowest $c(x)$ among all possible solutions $x_1, \ldots, x_k$. To calculate $\text{argmin}_x c(x)$, each of the K possible approximations must look at each of its K items. For each item, a distance is calculated from the item's vector of length J to all of the other $K-1$ items in the possible approximation. Thus, finding $\text{argmin}_x c(x)$ takes $K^3 \cdot J$ time.

Thus, the runtime of this approximation algorithm is $O(K^2NJ + K^3J)$. Assuming $N \geq K$, this algorithm runs in $O(K^2NJ)$ time. This is significantly faster[1] than the brute force solution for the objective, which runs in $O(N^K \cdot K^2)$, as discussed in Section 1.

[1] Assuming $N > K$.

Remark. M-root Selection

In analyzing datasets larger than $K = 10$, consider a γ such that $|c(x^i) - c(x^j)| < \gamma$, $\forall i, j \in V$. In practice, this γ is not very large. Thus, it does not drastically increase the cost of our approximation if we don't evaluate all sets x^i $\forall i \in V$, but only find the set x^i for $\forall i \in V \; st \; i \text{mod}(m) = 0$. Worst case, our approximation will cost γmore than $\hat{x}$. Thus, order to increase the efficiency of the algorithm, only every m^{th} bag was considered to create a star-graph when over ten images were analyzed. Thus, $\hat{x}$ was selected as the argmin of $\frac{K}{m}$ stars, not K stars. This did not dramatically change the results of the algorithm, though it did significantly decrease the runtime. However, it is important to note that this optimization delegitimizes the theoretical approximation argument made in Section 2.2.

Part II
Implementation

3 Defining the "parts" of an image

Images provide a great landscape for discussing similar parts, particularly because different images of the same item will be similar but not identical. In order to apply our algorithm to images, we convert the colored images to grayscale where each pixel is defined by a value in the range $[0, 255]$. Consequently, the images are now matrices with elements in range $[0, 255]$. Consider applying the algorithm to k images, $img_1, \ldots, img_k$. Consider a grayscale image img_i with size 300×450. The corresponding matrix to img_i is expressed as the $m \times n$ matrix D_i with $m = 300$, $n = 450$, and $\forall x \in D_i$, x is in range $[0, 255]$. Note that not all images need to be the same size.

For this algorithm, each image can be thought of as an individual bag. The items of the bag can be thought of as "patches" of the image In particular, for some w, the bag will contain all possible patches of the image of size $w \times w$. One way to visualize this is to imagine a window of size $w \times w$. The window slides horizontally and vertically, taking "snapshots" of the part of the image that the window contains.

In practice, the algorithm adopts the approach used by Agarwal and Roth [5] in which the window moves by 20% of the window size each iteration. For example, a square window of size 100 pixels would move 20 pixels on each iteration. There are, however, limitations to this approach. Consider the case where the item similar among all images is larger than $w \times w$. To ensure that every object in an image can be fully included in the window, we reduce the dimension of the image. With reduced image size (which results in a grainier image), a window of size $w \times w$ can now capture larger objects. This size reduction continues until the image itself is the size of the window. The collection of these "snapshots" at every size constitute the n items inside the bag. This

method of item generation allows objects of different sizes in different images to still be considered similar.

4 Scale Invariant Feature Transform Descriptors

Pixel-by-pixel descriptions of a subpatch of an image are not sufficient descriptors for identifying commonalities between subpatches. Because pixel-by-pixel descriptions only examine grayscale values, a set of identical images with different light exposures would be seen as completely different. In order to describe subpatches accurately and with informational descriptors, images are described using features attainted through Scale-Invariant Feature Transform (SIFT) filters. The goal of SIFT filters is to provide invariant descriptors of patches. These descriptors are intended to be robust to changes in illumination and viewpoint. Initially developed by Lowe [6], SIFT descriptors provide the perfect tool for subpatch description. This thesis used the SIFT feature descriptor implemented in the open-source library VLFeat [7].

SIFT descriptors utilize gradients to attain illumination, rotation, and size invariance. SIFT key points are initially detected by looking for local maxima or minima among a difference-of-gaussians function. Each feature is described by its key point center (in x,y coordinates), its rotation (described in radians), the radius of its region, and a descriptor. The descriptor of a feature describes the image gradients of a key point with a 3-D spatial histogram. Gradients that are farther from the key point are given less importance by a Gaussian weighting function over all the gradients. SIFT features have the advantage of being invariant to rotation, scale, translation, and some illumination changes. Therefore, SIFT features are ideal for recognizing similar yet rotated objects that vary in size, which is common in the sets of images this algorithm is analyzing. An example of an image and a random selection of some of its SIFT descriptors and keypoints can be found in Figure. In this figure, 800 descriptors are visualized. This image has over 50,000 keypoints and descriptors.

Another similar option for image feature description are Dense-SIFT features, however they lack the ability to make invariant descriptions. Unlike SIFT filters, Dense-SIFT filters do not identify interest points. Rather, they divide the image into a grid of cells and describe the histogram at each cell. These cells may overlap, allowing a comprehensive description of an image. While comprehensive, Dense-SIFT features lack both scale and rotational invariance. Consequently, they were not nearly as effective in recognizing similar features among similar images. Figures 4.2 and 4.3 show the performance of this algorithm while using SIFT and Dense-SIFT features respectively.

Figure 4.1: Original image and a random selection of 800 keypoints (in yellow) and their descriptors (in green)

5 Visual Words

5.1 Building a Vocabulary to Describe Images

In practice, the SIFT features of a given image or subpatch can be summarized in a low dimensional space using visual words. Visual words are an attempt to provide generic visual categorization of images. These visual words create a vocabulary of words that can be used to describe any image. Consequently, visual words allow for easy cross-image comparisons. An image may contain multiple visual words, often called a "bag of words". This thesis attempts to create a vocabulary, or codebook, of visual words using SIFT descriptors. Assuming a vocabulary can be formed, using the "bag of words" approach to describe the SIFT features of a given region allows similarity measurements to be done more quickly and consistently.

Research has already proved the success of using affine invariant point descriptors in a "bag of words" model. Csurka et al. [8] showed the utility of this method in object classification. By describing patches of images with invariant features and clustering those patches into one of 100 visual words, Csurka et al. created a "bag of key points" which counts the number of patches in each image assigned to each visual word. This "bag of key points" was then used as a feature vector to be analyzed by a multi-class classifier. Their approach proved most successful in classifying faces with only a 2% error rate.

Similarly, Agarwal and Roth [5] used a "bag of words" approach on interest points to create binary vectors that describe each image by the visual words

Figure 4.2: Item approximation using SIFT features

Figure 4.3: Item approximation using Dense-SIFT features

contained in the image. They were able to use this approach for classification with similarly high accuracy. However, unlike Csurka et al., Agarwal and Roth chose to embed spatial relationships among the interest points in the feature vector.

In order to form a vocabulary that can apply to most images, a comprehensive dataset was needed. The Pascal VOC[9] dataset provided a wide range of images in multiple categories.[2]

SIFT features were gathered from all of the images in the VOC dataset. These SIFT features were then used to create a generic vocabulary using K-means clustering with J clusters. The selection of J is discussed in Section 5.2. This thesis implemented clustering of visual words using open-source code from [7]. After clustering, we have J centers of each cluster. Each center represents the canonical SIFT descriptor of that cluster. These J descriptors of features form a codebook, or dictionary, of J visual words.

With this codebook, any given image can be defined by frequency histograms of these J visual words. This practice of describing parts with a histogram of gradients has already been done, notably by Leibe and Schiele [10] and Dalal and Triggs [11]. After calculating the SIFT features of a given image, we have a collection of SIFT features. Those features are then assigned to the closest cluster center using distance metric d. We now have a collection of visual words in an image, with each visual word associated with a feature center. With this information, a histogram of visual word frequency can be made for each subwindow. The histograms are normalized so the histogram for a subpatch represents the frequency of each visual word appearing in that subpatch. This histogram for subwindow i in image j will be the "item" x_i added to the "bag" B_j. When looking for similar items, our approximation algorithm will be calculating similarities between these different histograms.

5.2 Selecting the Number of Visual Words

Selecting the number of visual words can often seem like black magic. Using a codebook that is too small weakens the algorithm's ability to notice differences in images. With such a limited vocabulary, very different items are discussed using the same small collection of words, over exaggerating the similarity between items. On the contrary, using a codebook that is too large weakens the algorithm's ability to recognize similarity in images. If there are many visual words that describe extremely similar features, two items that are almost identical can be described using completely different visual words. Because the algorithm does not decipher the similarity of visual words themselves, we are stuck viewing the two similar items as starkly different. This weakens the algorithm's ability to notice similar items.

After experimenting with different values, the final results of this thesis were created with $J = 1000$ unless otherwise stated. This value was derived from

[2] All images were labeled with one or more of the following tags: 'aeroplane', 'bicycle', 'bird', 'boat', 'bottle', 'bus', 'car', 'cat', 'chair', 'cow', 'diningtable', 'dog', 'horse', 'motorbike', 'person', 'pottedplant', 'sheep', 'sofa', 'train', 'tvmonitor';

Figure 5.1: Sample of five results from similar-part approximation of ten images with 200 and 1000 visual words

a process of trial and error. An example of how results differ with different J values can be found in Figure 5.1. We see that there are two images where the algorithm does not capture anything with $J = 200$ but captures Obama's face with $J = 1000$. Without a quantitative approach to measuring algorithm accuracy (which goes beyond the scope of this thesis), the other results can be qualitatively identified as having equal performance. The topic literature suggests other more advanced approaches to cluster words. Agarwal and Roth [5] implement bottom-up clustering and cluster merging while Csurka et. al [8] select J based on algorithm accuracy when compared to other J values.

6 Locality Sensitive Hashing (LSH)

The similar-item approximation algorithm requires finding an item's nearest neighbor in each bag. This nearest neighbor search can be time-intensive working in high dimensions, where conventional nearest-neighbor search optimizations, such as K-D trees, rarely perform better than brute-force search. In

practice, though, the exact nearest-neighbor is not necessary for the functionality of this algorithm. Nearest neighbor approximations suffice, particularly because in each bag there are many items that are extremely similar.[3]

Locality Sensitive Hashing, first developed by Gionis, Indyk and Motwani [12], achieves nearest-neighbor approximations in sub-linear time. The algorithm creates a series of hash functions such that images that are similar have a high chance of collision. The probability of collision of two items is tied to the distance of those items. Thus, if items are very similar, they have a high chance of being hashed to the same "bucket".[4] Consequently, looking up a new item in the LSH scheme will hash the item to a particular "bucket". LSH will retrieve the closet item to the query in that bucket. This functionally returns the nearest neighbor to a given query point.

In this algorithm, an LSH table is created for each bag. These tables $(T_1, \ldots, T_K)$ are formed after items are added to each bag but before the similar-item approximation begins. During the approximation algorithm, when an item $x_i \in B_i$ is looking for the nearest item in B_j, x_i is queried in T_j. The result of the lookup, x_j, is accepted as the nearest neighbor to x_i in B_j.

This thesis used the LSH implementation created by Andoni and Indyk [13]. This implementation allowed for both l_1 and l_2 distance schemes. Both were experimented with. Ultimately, in practice, l_1 norms were more effective. The comparison of these two parameters can be seen in Figure 4.2 on page 13 and Figure 6.1 above. Figure 4.2 on page 13 are the results of the algorithm with d as the l_1 norm and Figure 6.1 are the results with the l_2 norm. In the top row of each figure are the original set of seven images. On the second row are the items selected for each image. In other words, the first element in column i represents B_i and the second element represents $\hat{x}_i$ where $\hat{x}$ is the optimal solution to the approximation and $\hat{x}_i \in B_i$.

7 Pre-processing of images using entropy

Occasionally in practice, not all subpatches of an image were included in the final bags. After the histogram of each subpatch was calculated, a pre-processing step excluded subpatches with extremely low histogram entropy. This exclusion was to prevent the algorithm from determining that the most similar items in a set of images were the subpatches with no visual words, such as the sky or a white patch.

The entropy of an frequency histogram of visual words, h, with length J can

[3] This is mainly a result of our item generation. For example, sliding windows imply that multiply items share overlapping parts of the image. Similarly, changing the resolution of an image may not alter the visual words of the image, implying that many items contain nearly identical histograms.

[4] A "bucket" can be thought of as the index that a particular hash value references. When an item is added to hash set, a hash function is applied to that item to derive the index to which bucket the item should be added.

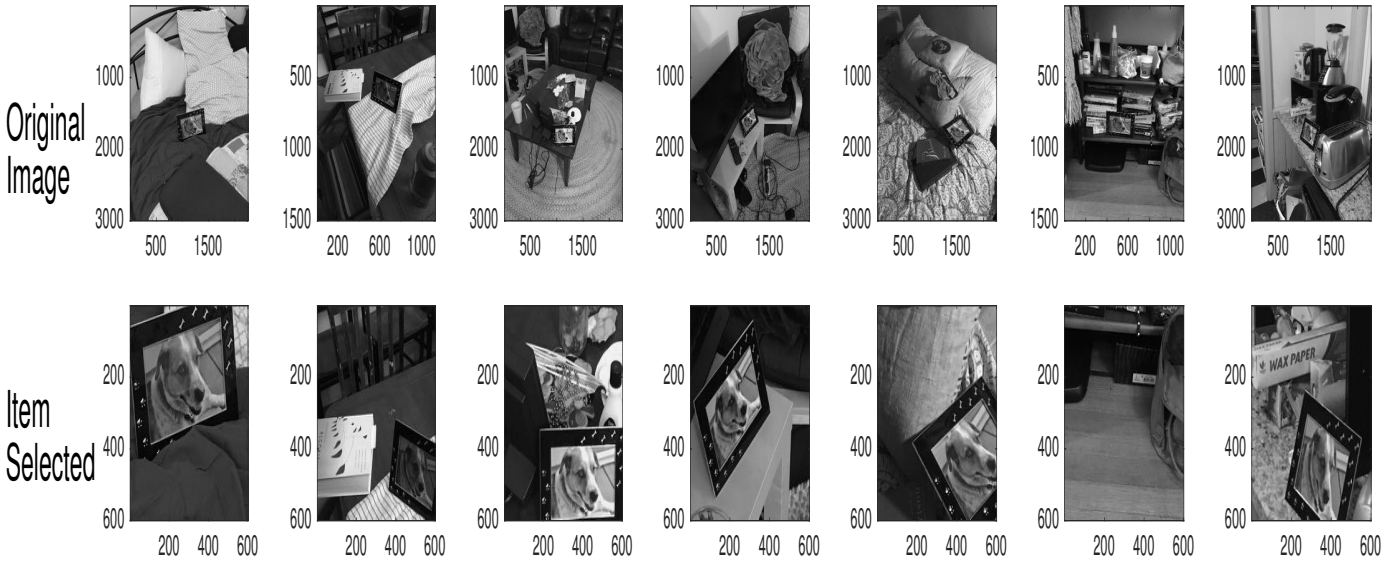

Figure 6.1: Item approximation using l_2 distances

be described by

$$entropy(h) = -\sum_{j=1}^{J} h_j \cdot log_2(h_j) \ .$$

Each histogram's entropy was compared to the mean entropy of all collected subpatches for all images. Histograms with entropy less than $\frac{1}{4}$ of the mean entropy were excluded from their respective bags.

More formally, let M be the total number of items in all bags. Let $B_i^{'}$ be all of the items for image i before pre-processing. For any $x_i \in B_i^{'}$

$$x_i \in B_i \iff entropy(x_i) \geq \frac{1}{4} \cdot \frac{1}{M} \sum_{j=1}^{K} \sum_{x \in B_j^{'}} entropy(x) \ .$$

In rare cases where the similar item in question has low entropy, subpatches with extremely high entropy were excluded. In practice, this was helpful for images with hectic backgrounds. Highly entropic images tended to have many visual words but very low frequency for each visual word. Consequently the normalized histogram for a subpatch with hectic images will have many visual words with nonzero probabilities, but most visual words will have extremely small probabilities. Thus, the l_1 distance between histograms of hectic subpatches will be very low because both will have an extremely small l_1 distance from the origin. Figure 7.1 shows an example of the effect of entropy filtering.

Figure 7.1: Item approximation with and without filtering out highly entropic subpatches

Figure 8.1: Sample of images created using the same face

8 Datasets

The implementation of this algorithm was run on a series of different datasets, each seeking to test the effectiveness of the algorithm in different contexts.

First, to ensure the algorithm was working, it was crucial to determine if the algorithm could find a common element e if $B_1 \cap B_2 \cap \cdots \cap B_k = e$. To confirm this, the same image of a face was inserted in multiple different random images (which were taken from the Pascal VOC dataset[9]). The insert varied in size and location, but nevertheless existed in all images. This provided an excellent test that the algorithm's approximation was able to detect similar (in this case identical) items that had different sizes in different images. The face was taken from the Yale Face Database[14]. Example images from this dataset can be found in Figure 8.1.

In general, the Yale Face Database provided a good dataset where images were almost the same, but varied by individual person. Nevertheless, if each face were superimposed on a different random busy image, the similar items in

each image should be the set of superimposed faces.

To ensure that the SIFT features were rotationally invariant, a small dataset was created that included the same exact item in different scenes. A framed photograph of a dog was placed in many different places. The dog's face and the frame were exactly the same in each image, the only difference being the rotation of the object. This framed photograph was rich enough in texture to contribute a substantial number of visual words to the histogram of a given patch. Moreover, because the framed photograph presented itself as different sizes in each image, it also pressed the algorithm to take advantage of image resizing to detect similar items.

A small dataset of Barack Obama in different scenes was created to test the algorithm's ability to notice extremely similar yet un-identical items. As opposed to the framed photograph, the facial expression and angle of Obama's face differed in each image. This presented a harder problem for the algorithm, because depending on the structure of Obama's face, histograms may be very different but should still be the most similar in patches around his face.

Finally, to notice entirely different items that maintained the same structure, the Pascal VOC dataset was used. Each class (person, dog, bicycle) presented a set of images with similar object grammars. Thus, the algorithm was run on items in a shared category. For example, ten images with the label "car" should return ten similar items that all include a car. The algorithm was also run at a large scale on 100 images of side-facing cars taken from the Pascal dataset. All cars had a minimum size of 60x60 in an image that was around 375x500.[5] This provided a chance to see how the algorithm performed at a larger scale.

Part III
Results

9 Performance of algorithm

In general, the approximation proved very effective. As seen in Figure 9.1, the algorithm worked when the face was superimposed on various images. The algorithm performed similarly well when different (yet almost identical) face portraits were superimposed, which is shown in Figure 9.2. These two examples prove the heart of the problem: there is an obvious similar item that the algorithm should be detecting.

When a key object is inserted in various scenes and rotation, size, and lighting are the only things that change (such as the Dog Frame dataset created for this thesis), the algorithm still is able to choose items with the key object with near perfect accuracy (Figure 4.2). In this case, the algorithm is less able

[5] Car size was determined by the size of the bounding box in the training data associated with the label. Images varied in size and orientation, but were between 202x500 and 375x500 (the most common size).

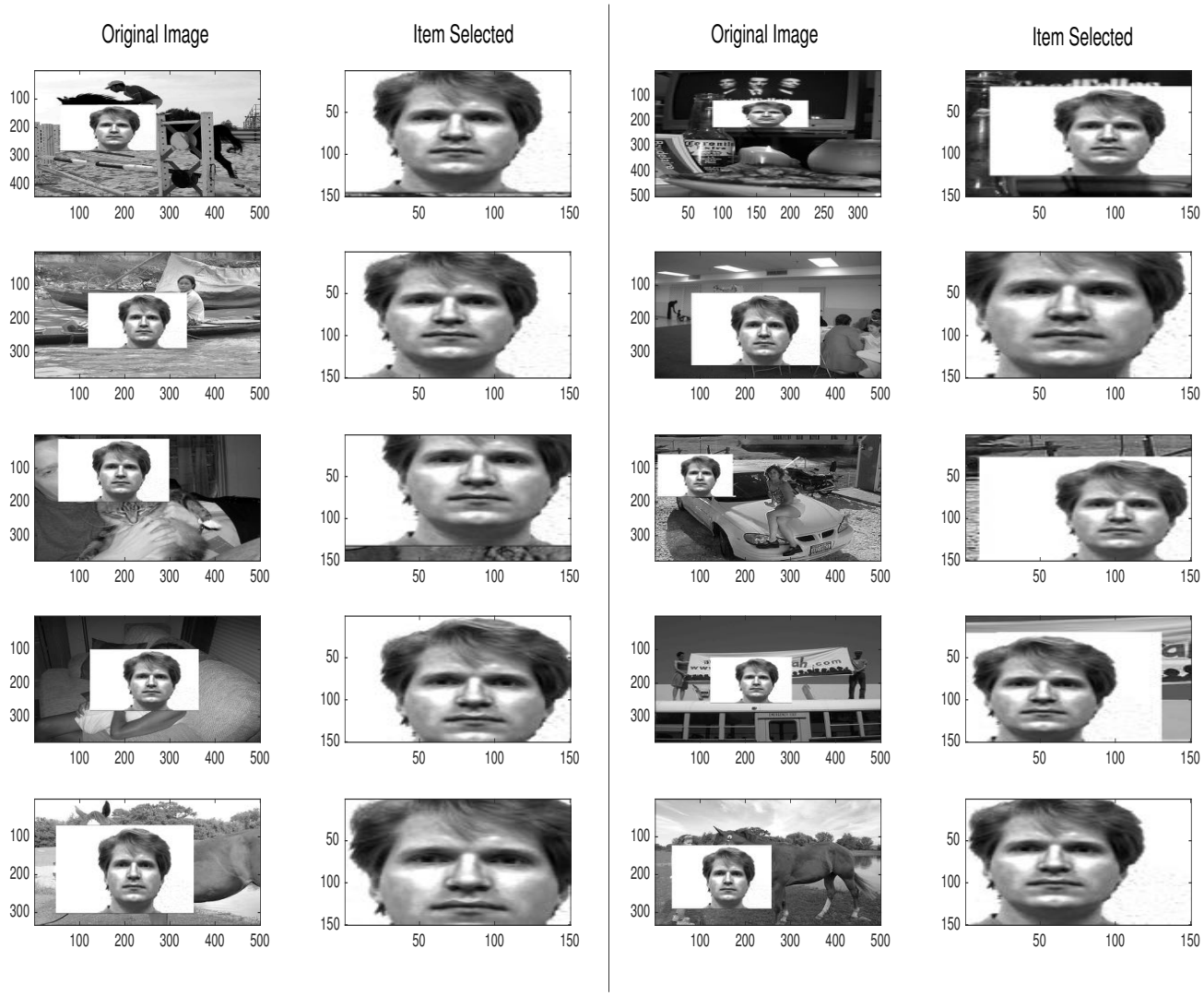

Figure 9.1: Performance of algorithm when the same item exists in every bag

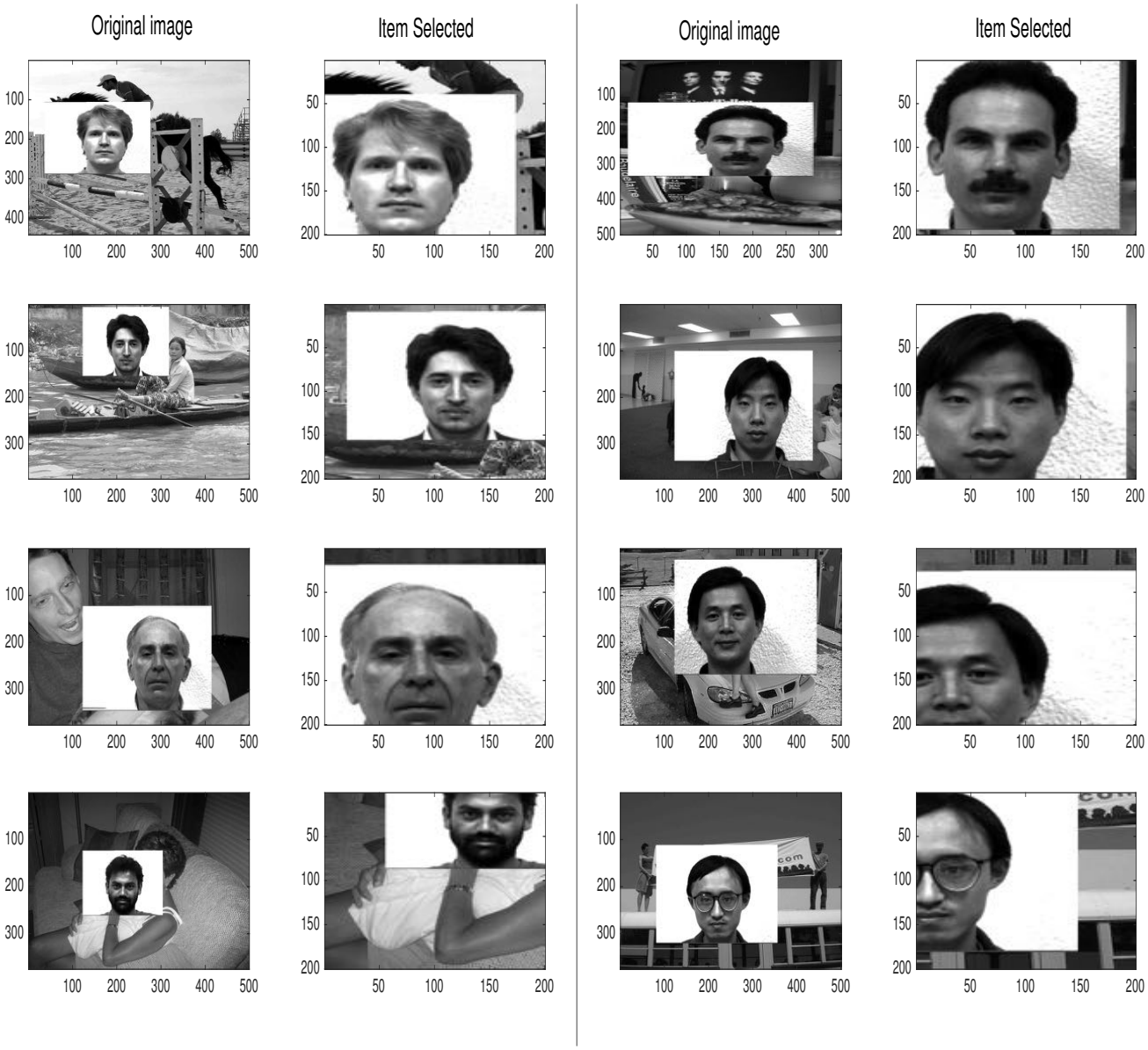

Figure 9.2: Performance of algorithm when near-identical images exist in every bag

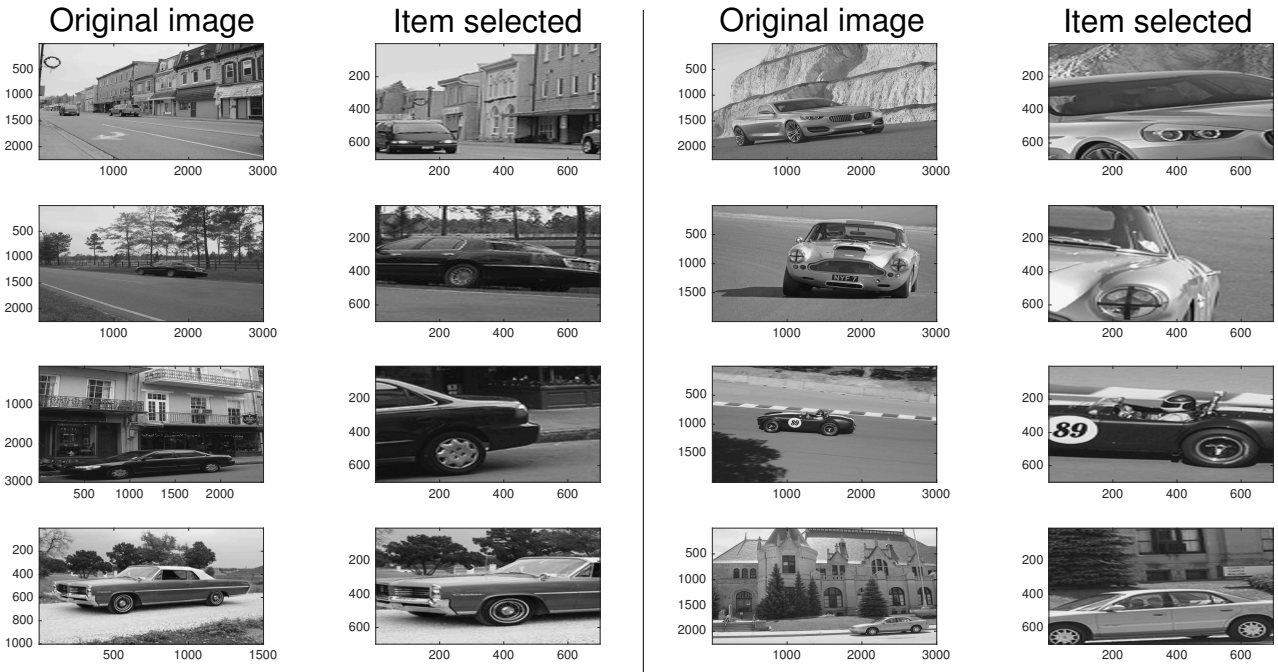

Figure 9.3: Performance of algorithm on 10 Pascal cars of different type, rotation, and size

to form a bounding box around the key object but is able to choose an item containing the key object. This is expected because the item generation schema described in Section 3 does not actually create every possible subpatch in every possible dimension. Consequently, the item that is a perfect bounding box around the key object may not even exist in any of the bags. This item generation implies that we cannot expect the algorithm to do any better than a generally zoomed-in subpatch of the key object.

The algorithm was still able to recognize similar objects from the Pascal dataset. In the Pascal dataset, images might share an object of the same class, but their similarity is much looser. Consequently, this proved a much harder exercise for the algorithm. An image of a frontal facing car and a left facing car may share similar semantic meaning. However, the similarity of the object from the viewpoint of visual words and SIFT features is quite small. This proved a limitation of the algorithm. Nevertheless, as seen in Figure 9.3 it was still able to discern commonality between these rotated and different images.

At a large scale of 100 images of cars selected from the Pascal dataset, the algorithm was able to hone in on key parts of the car in 79 of the images.[6]

[6]It is important to note that 8 of the images consisted entirely of a car. It was impossible

The other 11 items selected did not include parts of a car. Given how diverse the images were, this can be viewed as good performance, though not nearly as accurate as other experiments. A sample of 8 images and their selected items are shown in Figure 9.4.

Part IV
Discussion

The algorithm proved effective when the key objects of similarity among images were almost identical. SIFT was developed to find key points of an object and be able to recognize the same object from different viewpoints. It is consequently unsurprising that the algorithm is most effective on images that contain the same object in different viewpoints. While the car and (in some regards) Obama datasets had completely different objects of the same class, the algorithm was still able to recognize similarity. Further exploration should be done on the effects of other invariant feature descriptors.

While l_1 norm proved the most effective distance measure among l_p distance functions, it may not be the most appropriate. For example, chi-squared distances are most often used when comparing histograms. However, due to limitations given by the support code utilized in this thesis[7], other distance measures were not explored. However, they certainly would be able to improve the algorithm. Because this thesis was limited to l_1 and l_2 distances, entropy filtering (Section 7) was used to eliminate some of the faults of l_1 distance comparisons. However, entropy is not the best way to describe these histograms. While they are frequency histograms, they are not probability distributions and an image with low or high entropy does not say anything about its innate probability of being the most similar item in a bag.

The similar item approximation can be easily extended to apply to sets of images with multiple common parts. For example, a set of faces will have eyes, ears, nose, and mouth all in common. By running the algorithm multiple times on the set of faces with some small adjustments to the items $B_1, \ldots, B_k$, we can determine multiple common parts of image.

Consider a final star $\hat{x}$, which is derived from the min-star generated by B_r and thus has root $\hat{x_r}$. Let $x_m = \text{argmax}_{k \in Vr} d(\hat{x}_r, \hat{x}_k)$. That is, $\hat{x}_m$ is the furthest distance from the center of the star. Let us refer to that distance as f. For every item $x_i^k \in B_k, \forall k \in V \backslash \{m\}, \forall i$ we know that if $d(x_i^k, \hat{x}_r) \leq f$, x_i^k is more similar to $\hat{x_r}$ than any item in B_m. Thus, we can assume no new information is given by those items. They cannot represent a new aspect of the images, as they are closer to the root of the star than other items that are in the star. Thus, for each bag, we remove all elements that are are closer to $\hat{x}_r$ than $\hat{x}_m$.

for the algorithm to fail on those images, so accuracy estimations may be artificially inflated.

[7] The LSH support code [12] only accepts l_1 and l_2 distance functions.

Figure 9.4: Sample of results from running algorithm on 100 side-view images of cars. On the left are original images and on the right are the selected items

We now have a new set of bags $B_1^{'}, \ldots, B_K^{'}$ that have no items close to the similar item that is described by the star $\hat{x}$ with root $\hat{x}_r$. Thus, we can run the algorithm again on our new set of bags to find a new distinct set of items described by a completely different star $\hat{x}^{'}$, whose root has at least a distance of f from the root of our original star $\hat{x}$.

Consider a set of L distinct stars represented by $\hat{x}_1, \ldots, \hat{x}_L$. Each star represents a distinct similar part that's shared between bags $B_1, \ldots, B_K$. These stars could potentially serve as the sub-parts of a new object grammar.

Conclusion

This thesis has shown that a 2-approximation algorithm can be implemented to estimate similar items among a set of images. SIFT features and visual words aid in describing the image beyond pixel-by-pixel values. More exploration needs to be done in different distance functions and feature descriptors to improve the algorithm's accuracy on images with semantic similarity but very different image representations. Future applications of this research include building object grammars. Moreover, by applying this research to non-visual datasets such as public speech transcripts, similar-part approximation can describe the thematic emphases by honing in on similar patterns among a series of speeches. Finally, if applying this algorithm to different scenes, similar-part approximation can help define the canonical aspects of a given environment or location.

References

[1] Girshick, R., P. Felzenszwalb and D. McAllester. "Object Detection with Grammar Models." 2011 Advances in Neural Information Processing Systems 24 (NIPS), 2011

[2] Prioletti, A., A. Møgelmose, P. Grisleri, M. Manubhai Trivedi, A. Broggi, and T. B. Moeslund. "Part-Based Pedestrian Detection and Feature-Based Tracking for Driver Assistance: Real-Time, Robust Algorithms, and Evaluation." IEEE Transactions on Intelligent Transportation Systems 14(3):1346-1359, 2013

[3] Gerónimo, D., A. M. López, A. D. Sappa, and T. Graf. "Survey of Pedestrian Detection for Advanced Driver Assistance Systems." IEEE Transactions on Pattern Analysis & Machine Intelligence, 2010

[4] Felzenszwalb, P. Personal communication.

[5] Agarwal, S., and D. Roth. "Learning a Sparse Representation for Object Detection." ECCV Lecture Notes in Computer Science, 2002

[6] Lowe, D. "Object Recognition from Local Scale-invariant Features." Proceedings of the Seventh IEEE International Conference on Computer Vision, 1999

[7] Vedaldi, A. and B. Fulkerson. “VLFeat: An Open and Portable Library of Computer Vision Algorithms.” http://www.vlfeat.org

[8] Csurka G, C Dance, L Fan, J Willamowski, C Bray. “Visual categorization with bags of key points.” EECV Workshop on Statistical Learning in Computer Vision, 2004Gionis, A., P. Indyk, and R. Motwani. “Similarity search in high dimensions via hashing.” Proceedings of the 25th International Conference on Very Large Data Bases (VLDB), 1999

[9] Everingham, M., L. Van-Gool, C. K. I. Williams, J. Winn and A. Zisserman. “The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results". http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html

[10] Leibe, B., and B. Schiele. "Interleaved Object Categorization and Segmentation." Proceedings of the British Machine Vision Conference, 2003

[11] Dalal, N., and B. Triggs. "Histograms of Oriented Gradients for Human Detection." IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005

[12] Gionis, A., P. Indyk, and R. Motwani. “Similarity search in high dimensions via hashing.” Proceedings of the 25th International Conference on Very Large Data Bases (VLDB), 1999

[13] Andoni, A., and P. Indyk. “LSH Algorithm and Implementation (E2LSH)” http://www.mit.edu/~andoni/LSH/

[14] Belhumeur, P., J. Hespanha, and D. Kriegman. “Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection,” IEEE Transactions on Pattern Analysis and Machine Intelligence, July 1997, pp. 711-720.