Sketch-based Object Recognition

Bo Zhu[*]
Stanford University

Ed Quigley[†]
Stanford University

Abstract

We present a system that performs sketch-based image retrieval and classification. The system enables the user to draw sketches that are used as queries against a dataset of 20,000 sketched images spanning 250 categories. The system returns top matches and uses a weighted voting scheme to estimate the most likely classifications of the user's sketch based on those matches. This work demonstrates the potential of the computer to compare and classify hand-drawn images in addition to the more frequently researched digital photographs.

Keywords: sketch recognition, image retrieval, histogram of gradients

1 Introduction

Image-based search is a valuable tool that is closely tied to the computer vision problem of object recognition and classification. In a system that performs image searches, it is desirable both to find close image matches and to classify a query image. Often, however, a user may not have a digital photograph of an object for which he or she desires to perform a search. In this case, it is desirable that the user be able to perform the search using a rough sketch of the desired object.

Our system is focused on the specific problems of matching sketched query images to a database of sketches and classifying those query images. The work is meaningful because it demonstrates the potential of the computer to classify and compare hand-drawn sketches, which are imperfect and sparse relative to actual photos. We use techniques developed in [Eitz et al. 2012a] and [Eitz et al. 2011] to implement a system that performs sketch-based object recognition and matching.

2 Background

In this section we reference relevant work in the areas of sketch-based image search and recognition and summarize the contributions made by our project.

2.1 Previous Work

Problems related to object recognition constitute a broad category of computer vision. Recently, the ability to recognize real world objects using rough hand-drawn sketches has been of particular interest. The MindFinder system, described in [Cao et al. 2010] and [Wang et al. 2010], performs image search based on a rough sketch provided by the user. [Sun et al. 2013] is a another image searching system that allows the user to provide both edge and color information in a sketch. [Chen et al. 2009] uses annotated sketches as a blueprint for creating composited scenes using images available online. A method for returning three dimensional objects based on query sketches is described in [Eitz et al. 2012b]. [Cole et al. 2008] presents the results of a study on how artists sketch objects. A sketch-based image retrieval tool that can be used on mobile devices due to its low memory consumption is detailed in [Tseng et al. 2012]. [Ribeiro and Igarashi 2012] considers sketches in the context of games. We base our project primarily on [Eitz et al. 2012a], which allows a user to draw an object while interactively providing ranked guesses at the object's category.

2.2 Our Contribution

Our work does not add new research to the body of work referred to in the previous section. Instead, we attempt to synthesize the functionality of the categorization and image search systems in order to provide both "closest match" image results as well as ranked guesses at the object portrayed in a query sketch. Further, by creating a system that successfully accomplishes these goals, we verify that the techniques described in [Eitz et al. 2011] and [Eitz et al. 2012a] are able to perform sketch recognition tasks. This verification of reproducability is an important factor in the progress of any scientific field.

3 User Interface

We provide the user with an interface for creating sketches and performing searches based on those sketches. The system interface is depicted in Figure 1. The upper-left section of the interface provides a canvas on which the user can sketch a query image. Controls to the right of the canvas include, from top to bottom, paint brush, eraser, clear screen, and search. When the user performs a search, the right side of the window is populated with the 30 closest matches from the sketch dataset. The closest match is in the upper left, and images are listed in decreasing order from left to right, top to bottom. Below the canvas, the top five most likely object classes are listed in decreasing order from top to bottom.

4 Algorithm

In this section we describe the implementation of the sketch-based object matching system. The overall system pipeline is illustrated in Figure 2.

4.1 Technical Summary

Our implementation consists of matching HoG features of a query image to a dictionary of codewords. This dictionary is generated by clustering the histograms of gradients computed from our training set, which is a group of 20,000 human-sketched images in 250 categories provided by [Eitz et al. 2012a]. The details of this method are provided in Section 4.2.

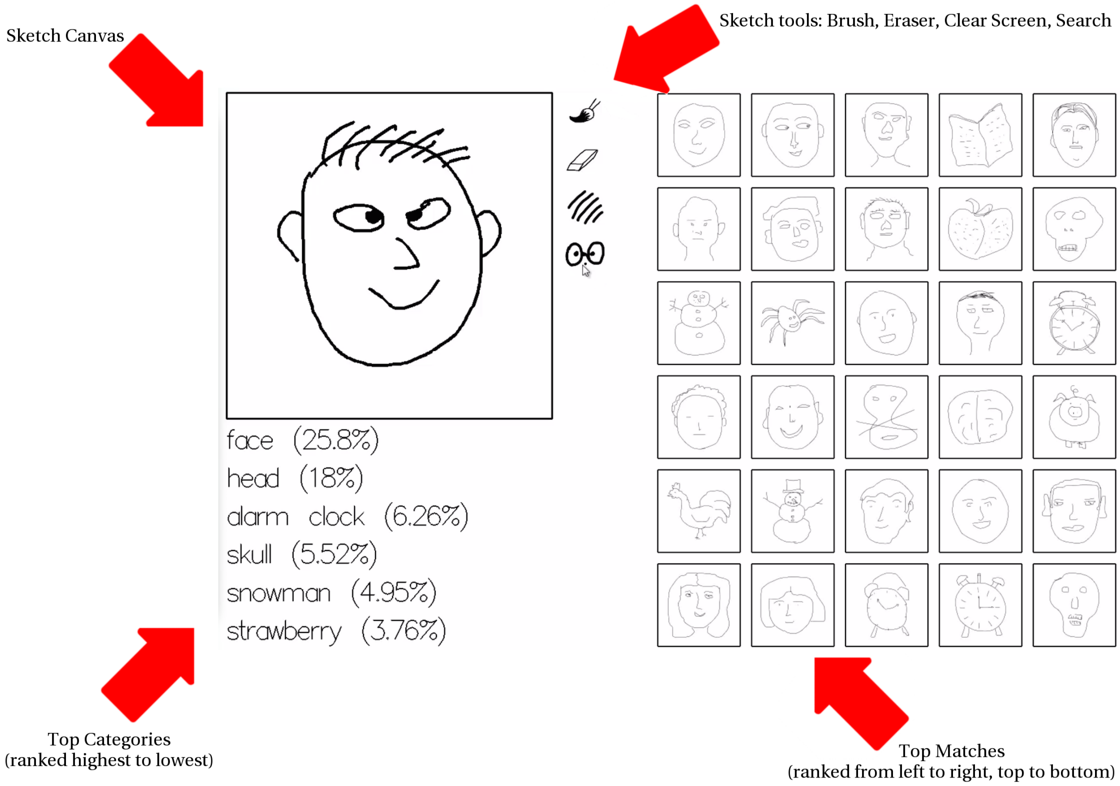
4.2 Technical Details

In the following subsections we describe the implementation details of our system.

4.2.1 Dataset

We use the dataset provided by [Eitz et al. 2012a] which contains 20,000 manually drawn sketches collected from Amazon Mechan-

[*] e-mail: boolzhu@stanford.edu
[†] e-mail: equigley@stanford.edu



Sketch Canvas

Sketch tools: Brush, Eraser, Clear Screen, Search

face (25.8%)
head (18%)
alarm clock (6.26%)
skull (5.52%)
snowman (4.95%)
strawberry (3.76%)

Top Categories
(ranked highest to lowest)

Top Matches
(ranked from left to right, top to bottom)

Figure 1: *User interface, with labeled components*

ical Turk. The dataset includes 250 categories of objects ranging from simple (e.g., apple, sun, mug) to complex (e.g., clock, angle, face). Each category has 80 images, and the images contain only the sketch lines of the object (i.e., no other scene context is contained within the sketches). All images shown as search results in this report come from this dataset.

4.2.2 Feature Extraction

Our system extracts features from a sketch using HoG descriptors, based on the method described in [Eitz et al. 2012a]. We first map each input sketch onto a 256×256 image by isotropically rescaling it using the ratio of the length of the longest axis of the bounding box of the sketch and the edge length of the rescaled image. This step ensures the scale and translation invariance of the feature descriptor for each sketch, though we note that our algorithm does not ensure invariance to rotation.

We sample the rescaled image at $1,024$ points regularly spaced in a square grid on the image. Each of these points is used as the center of a window whose area is approximately nine percent of the total image area. Neighboring windows have a high degree of overlap because of our dense sampling. Each window is divided into 16 bins, four along the x-axis and four along the y-axis. For each pixel of each bin, the gradient is determined by applying the filters $[-1\ 0\ 1]$ and $[-1\ 0\ 1]^T$. This results in one of eight possible gradient values, because we ignore the ninth possible gradient that corresponds to no color change, $(0,0)$. Each bin is given the value of the most frequently occuring gradient within the bin. Thus each image has $1,024$ feature vectors, each of length $4\ x\ bins\ \times\ 4\ y\ bins\ \times\ 8\ orientations = 128$. An illustration of a single window is given by Figure 3.

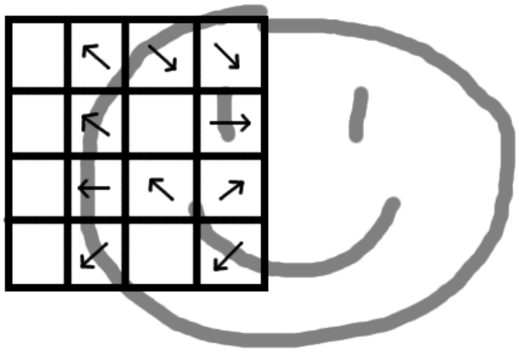


Figure 3: *A* 4×4 *window corresponding to a single feature vector, with nonzero gradients displayed.*

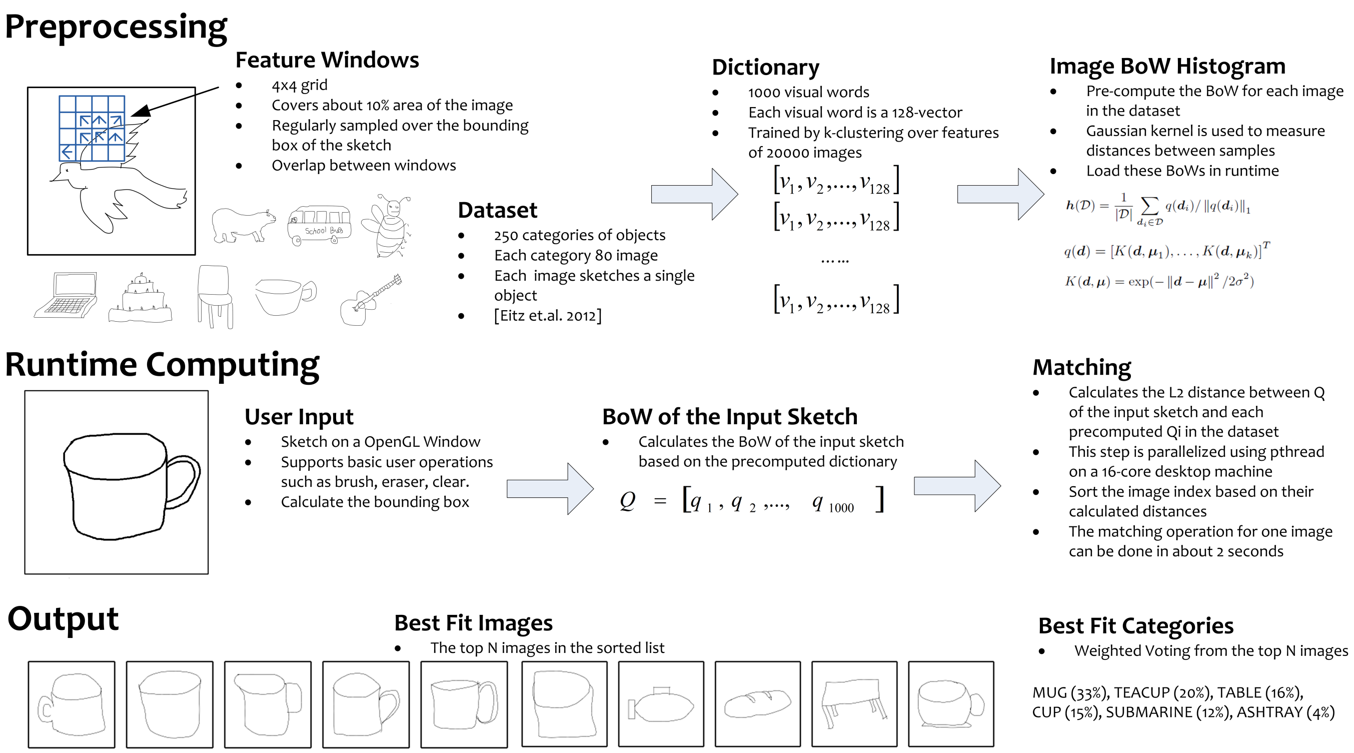

Preprocessing
Feature Windows
4x4 grid
Covers about 10% area of the image
Regularly sampled over the bounding box of the sketch
Overlap between windows
School Bus
Dataset
250 categories of objects
Each category 80 image
Each image sketches a single object
[Eitz et.al. 2012]
Dictionary
1000 visual words
Each visual word is a 128-vector
Trained by k-clustering over features of 20000 images
$[v_1, v_2, ..., v_{128}]$
$[v_1, v_2, ..., v_{128}]$
……
$[v_1, v_2, ..., v_{128}]$
Image BoW Histogram
Pre-compute the BoW for each image in the dataset
Gaussian kernel is used to measure distances between samples
Load these BoWs in runtime
$h(\mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{d_i \in \mathcal{D}} q(d_i) / \|q(d_i)\|_1$
$q(d) = [K(d, \mu_1), \ldots, K(d, \mu_k)]^T$
$K(d, \mu) = \exp(-\|d - \mu\|^2 / 2\sigma^2)$
Runtime Computing
User Input
Sketch on a OpenGL Window
Supports basic user operations such as brush, eraser, clear.
Calculate the bounding box
BoW of the Input Sketch
Calculates the BoW of the input sketch based on the precomputed dictionary
$Q = [q_1, q_2, ..., q_{1000}]$
Matching
Calculates the L2 distance between Q of the input sketch and each precomputed Qi in the dataset
This step is parallelized using pthread on a 16-core desktop machine
Sort the image index based on their calculated distances
The matching operation for one image can be done in about 2 seconds
Output
Best Fit Images
The top N images in the sorted list
Best Fit Categories
Weighted Voting from the top N images
MUG (33%), TEACUP (20%), TABLE (16%), CUP (15%), SUBMARINE (12%), ASHTRAY (4%)

Figure 2: *Overview of sketch matching and classification algorithm*

4.2.3 Dictionary

By performing the feature extraction described in Section 4.2.2 for each image in the dataset, we get $20,000$ matrices with dimensions 1024×128. In order to perform image searches in a reasonable amount of time, we want to reduce the size of these image descriptions. To accomplish this, we use a bag of words model. Once we have computed the HoG features for each training image, we generate a dictionary of codewords by clustering the HoG descriptors using the k-means algorithm. One would expect the accuracy of the method to vary directly with the number of codewords, and the speed of the search to vary inversely with the number of codewords. Based on the findings reported by [Eitz et al. 2011], we chose to use a dictionary of $1,000$ codewords.

4.2.4 BoW Histogram

Using the trained dictionary, each sketch can be described by a bag of words (BoW) histogram. The histogram is calculated using a Gaussian kernel as in [Eitz et al. 2012a]. Each feature descriptor f from the input sketch is compared with each codeword d in the dictionary using the equation

$$S_{f,d} = e^{-|f-d|^2/2\delta^2} \tag{1}$$

where δ is the length scale of the Gaussian kernel (we use $\delta = 0.1$ as in [Eitz et al. 2012a]). The histogram of feature f is then given by the n dimensional vector

$$F = (S_{f,d_1}, S_{f,d_2}, ..., S_{f,d_n}) \tag{2}$$

in which $d_1, d_2, ..., d_n$ are n visual words in the dictionary. The Gaussian kernel allows a smoothed distribution of a feature in the feature space spanned by the n visual words. This histogram is normalized as $\hat{F} = F/|F|$ to eliminate the influence of the local window size.

The BoW histogram of a sketch is calculated as the average of its n extracted features, $H = \sum_{i=1}^{n} \hat{F}_i/n$. Thus H is also an n dimensional vector. We precompute H for the 20,000 dataset images and store them as a binary file. At runtime this file is loaded into memory for computing the smallest distances to query sketches. This method provides a compact representation of a sketch image.

4.2.5 Matching and Voting

When a search is performed, a histogram of codeword occurances for the query sketch is computed in the same way as the histogram for a training image. The closest n dataset images (we use $n = 100$) are then found by comparing the query histogram to the dataset histograms using their Euclidean distance. The top 30 closest matches are presented to the user. All 100 of the matches, sorted by their distance to the query sketch, are used to vote for the most likely class of the query sketch. In general, the class of the sorted image i is given a weighted vote of $n - i + 1$. Thus the first image votes for its class with a weight of 100, the second votes with a weight of 99, and so on. The top five class results, along with their vote percentages, are reported to the user.

5 Experiments

Most of our experiments involved testing the ability of the system to classify a set of images drawn by our team members. In these experiments, we found that the method was effective both at distinguishing very different classes (e.g., shirt vs face) as well as discriminating between similar classes (e.g., apple, tomato, and pear).

In the following subsections we discuss specific results produced by our system.

5.1 User Sketch Recognition

We tested our system by sketching a variety of different objects. A subset of our sketches along with the top ten closest matches from the dataset and the top object class is given in Figure 5.

5.2 Incremental Sketch Editing

Our system supports incrementally editing the sketch to improve the accuracy of the search results. As shown in Figure 4, the recognized best fit category can be corrected from “pear” to “strawberry” by incrementally adding more local features (seeds).

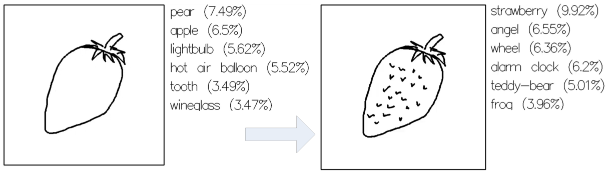


Figure 4: *Incremental sketch editing.*

5.3 Quantitative Analysis

In order to quantify our results, we split our dataset into a training subset and a test subset. We generated a new codeword dictionary based on the first 75% of the sketches from each category, then used the remaining 25% of the dataset as query sketches. We found that in 82% of these cases, the correct object class was within the top five guesses produced by our weighted voting scheme.

6 Limitations and Discussion

One of the major limitations of a sketch based recognition system is its accuracy. Although our system can achieve about 80% accuracy within the top five categories (as discussed in Section 5.3), there are still failure cases that may be worthwhile to explore and improve. Based on our experiment results, we group the failure cases into four different categories: inter-class ambiguity, intra-class ambiguity, perspective, and stroke style.

6.1 Inter-class ambiguity

Different categories of objects may share similar shapes or features, which introduces inter-class ambiguity to our recognition system. As shown in Figure 6, objects like a fork, watch, carrot, leaf, microphone, etc. have similar global shapes. Although there are differences in the local features of these objects, the dominant feature in the diagonal direction overwhelms the local differences and creates an inter-class ambiguity. This ambiguity due to similar global shapes may be solved by using a higher resolution window with a smaller area for feature extraction. However, given the requirement of interactive performance, our system uses a window that achieves results with an acceptable tradeoff between accuracy and speed.

In addition to global shape ambiguities due to global features, local features may introduce inter-class ambiguities. As shown in Figure 7, our algorithm mistakenly classifies the input sketch (on the left of the figure) as a “sponge bob” instead of a “house” based on the voting results. This ambiguity comes from the local square shapes in both the “house” and the “sponge bob” categories. Since methods based on local features consider only the similarity between local descriptors and the relationships between different features in the entire image is not taken into account, repetitive local features in the feature space may overwhelm other important global features in the image space. To solve this problem, a more advanced recognition algorithm taking the spatial relationships into account (e.g. [Savarese and Fei-Fei 2007]) would be needed.

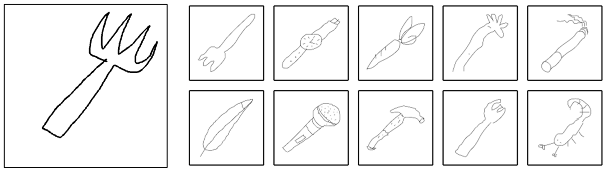


Figure 6: *Example of inter-class ambiguity due to global shape similarities.*

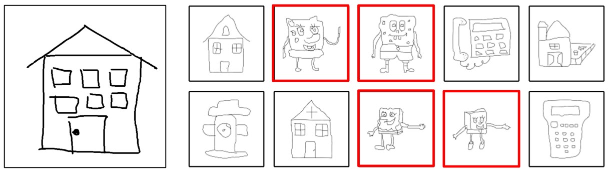


Figure 7: *Example of inter-class ambiguity due to local shape similarities.*

6.2 Intra-class ambiguity

Intra-class ambiguity comes mainly from the multiple representatives of the same category in our dataset. For example, Figure 8 shows eight of the different representations of bread in the dataset [Eitz et al. 2012a]. Even if a query sketch matches one of these eight entries perfectly, the voting results may still be wrong due to an insufficient number of matches within the “bread” category. A simple solution to this problem is to significantly expand the number of images in the dataset.

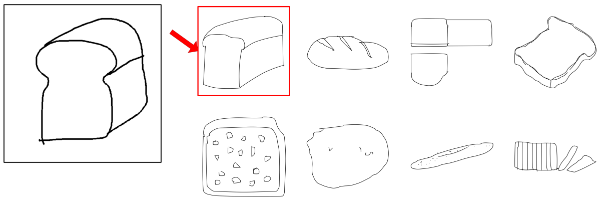


Figure 8: *Examples of intra-class ambiguity.*

6.3 Perspective

Orientation and perspective of a sketch may cause problems in recognition since our algorithm is not invariant to rotation. As shown in Figure 9, a query sketch can match an axe drawn facing in one direction while missing the axes drawn in all other directions. The insufficient number of samples in the “axe” category facing the same direction may lead to incorrect voting results. In addition to two dimensional orientation, three dimensional perspective changes of a sketch also make recognition more complex.

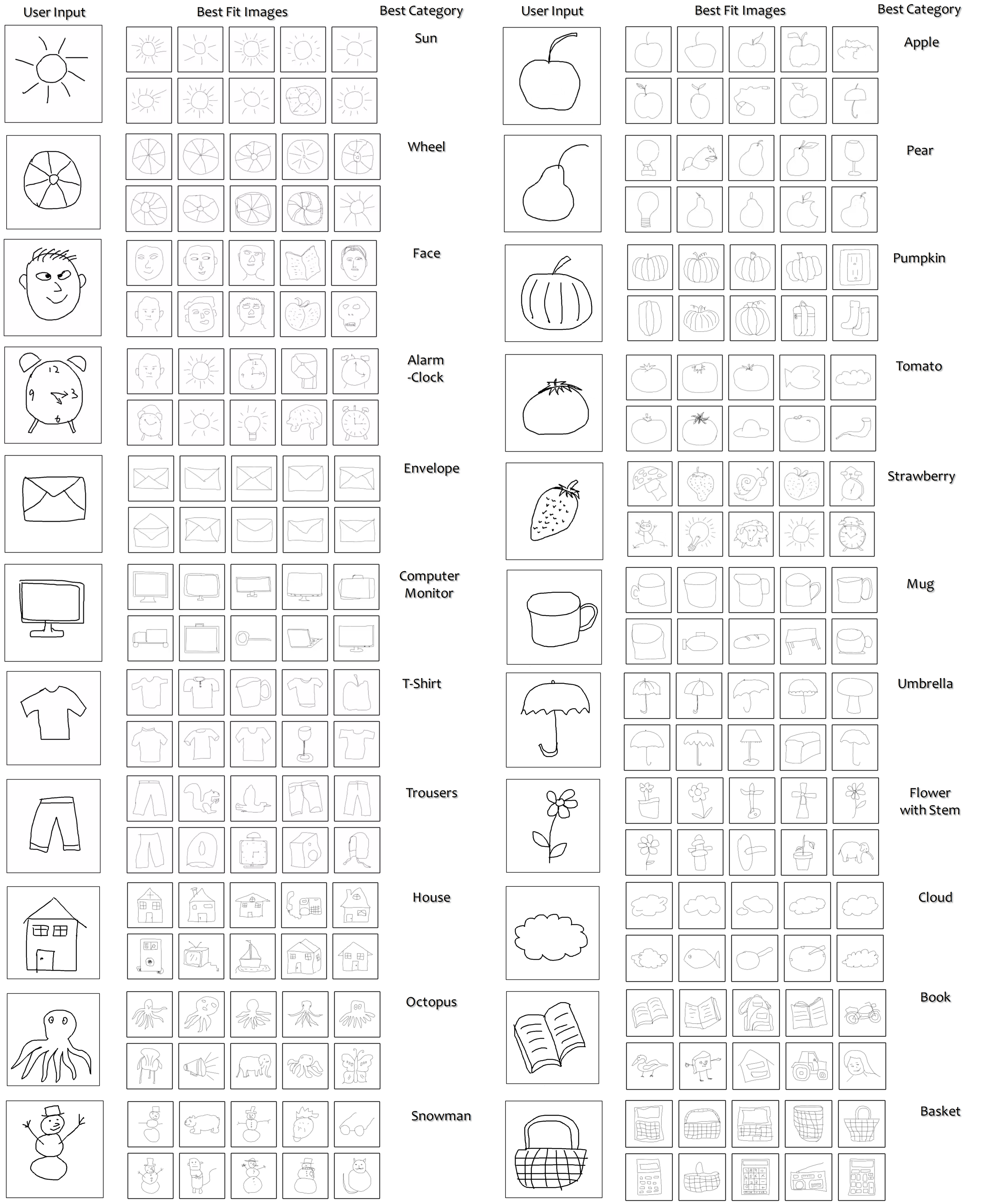


Figure 5: *Results.*

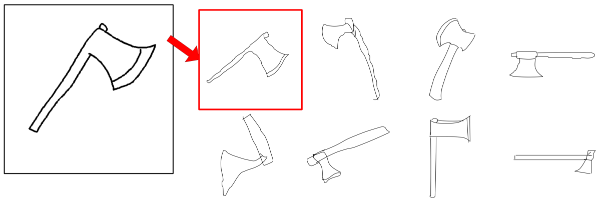


Figure 9: *Failure case due to variable orientations.*

6.4 Stroke styles

Figure 10 shows an interesting failure case due to different stroke styles. The main difference between the two query sketches is the style of the corners (sharp vs round), which leads to a significant difference in the accuracy of the search results. The round corner style, which is prevalent in many other categories (e.g., cactus, elephant, camel, etc.), negatively affects the accuracy of the search results while the sharp corner style, which is rare in other categories, lead to successful matching. This difference is understandable since our matching algorithm is sensitive to local features. However, considering that different users may have different drawing styles, the effect of the difference should ideally be made as small as possible.

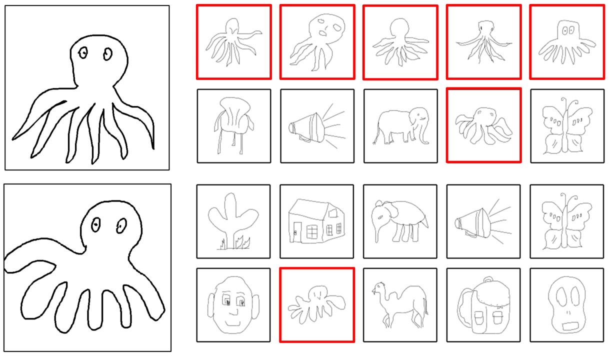


Figure 10: *Different stroke styles affect the search results.*

7 Conclusions

Sketch processing, recognition, and search are interesting lines of work in current computer vision research. The computer's ability to recognize human drawings has potential applications in the areas of image search and educational gaming. For example, image search could potentially be performed by only providing a rough sketch of the desired object. In the area of education, young children could learn to draw objects in a computer game that automatically evaluates the category of the sketched object.

We find that using a bag of words algorithm with HoG descriptors is a reasonable technique for solving the problem of classifying and matching human-drawn sketches. However, since a sketched image is very different from a digital photograph (e.g., it contains no color information, is sparser than a photograph, varies with a human's drawing style, etc), it may be that alternative methods that are ill-suited to actual photos may be effectively employed in processing sketches. Thus, problems related to sketched images are a fruitful avenue for continued computer vision research.

7.1 Source Code

Our source code is available for download at `https://bitbucket.org/equigley/cs231a_project/get/master.zip`.

References

CAO, Y., WANG, H., WANG, C., LI, Z., ZHANG, L., AND ZHANG, L. 2010. Mindfinder: interactive sketch-based image search on millions of images. In *Proceedings of the international conference on Multimedia*, ACM, 1605–1608.

CHEN, T., CHENG, M.-M., TAN, P., SHAMIR, A., AND HU, S.-M. 2009. Sketch2photo: internet image montage. In *ACM Transactions on Graphics (TOG)*, vol. 28, ACM, 124.

COLE, F., GOLOVINSKIY, A., LIMPAECHER, A., BARROS, H. S., FINKELSTEIN, A., FUNKHOUSER, T., AND RUSINKIEWICZ, S. 2008. Where do people draw lines? In *ACM Transactions on Graphics (TOG)*, vol. 27, ACM, 88.

EITZ, M., HILDEBRAND, K., BOUBEKEUR, T., AND ALEXA, M. 2011. Sketch-based image retrieval: Benchmark and bag-of-features descriptors. *Visualization and Computer Graphics, IEEE Transactions on 17*, 11, 1624–1636.

EITZ, M., HAYS, J., AND ALEXA, M. 2012. How do humans sketch objects? *ACM Transactions on Graphics (TOG) 31*, 4, 44.

EITZ, M., RICHTER, R., BOUBEKEUR, T., HILDEBRAND, K., AND ALEXA, M. 2012. Sketch-based shape retrieval. *ACM Transactions on Graphics (TOG) 31*, 4, 31.

RIBEIRO, A., AND IGARASHI, T. 2012. Sketch-editing games: human-machine communication, game theory and applications. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*, ACM, 287–298.

SAVARESE, S., AND FEI-FEI, L. 2007. 3d generic object categorization, localization and pose estimation. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, IEEE, 1–8.

SUN, X., WANG, C., SUD, A., XU, C., AND ZHANG, L. 2013. Magicbrush: image search by color sketch. In *Proceedings of the 21st ACM international conference on Multimedia*, ACM, 475–476.

TSENG, K.-Y., LIN, Y.-L., CHEN, Y.-H., AND HSU, W. H. 2012. Sketch-based image retrieval on mobile devices using compact hash bits. In *Proceedings of the 20th ACM international conference on Multimedia*, ACM, 913–916.

WANG, C., LI, Z., AND ZHANG, L. 2010. Mindfinder: image search by interactive sketching and tagging. In *Proceedings of the 19th international conference on World wide web*, ACM, 1309–1312.