A Day at the Museum

Catie Chang, Mary Etezadi-Amoli, and Michelle Hewlett

I. INTRODUCTION

Our objective was to identify 33 different paintings from the Cantor Arts Center, based on a database of images depicting the paintings at various angles, orientations, and illuminations.

We considered two main approaches for solving this image processing problem. The first used color histograms, which are advantageous because they do not change significantly across minor variations in scales and angles. Furthermore, we found that warping or rotations were not necessary for achieving high performance. One disadvantage of using this color histogram method is that lighting and noise can greatly affect the color content of the images. Our second approach was based on scale-invariant feature detection. However, the performance of the feature detection was sensitive to the choice of several parameters, and further optimization over these parameters may have been necessary to achieve perfect performance. In addition, our implementation of feature detection was more computationally expensive than the color histogram approach. After testing both algorithms, the color histogram method achieved higher accuracy and faster runtime.

Our color histogram approach consists of four main steps (see Figure 1). First, the painting is segmented from the background. Then, the colors in this painting are quantized based on a custom color palette, which we designed in order to be representative of the colors across all of the training images in our set. Next, the image is partitioned into four quadrants, and normalized histograms of color values are computed. Finally, the color histograms of this image are compared against those of the training images. We use a histogram intersection metric to quantify the similarity between histograms. The mean of the histogram intersection values across all four quadrants serves as our final measure of similarity.

To classify a novel image, we compute its similarity with every other image in the database. We identify the novel image as the painting corresponding to the most similar training image.

We spent a day at the Cantor Arts Center and took three more pictures of each painting. Our training set was thus comprised of 198 images.

![Fig. 1. Flowchart of algorithm.](image)

II. SEGMENTATION

Our first task was to segment the painting from the remainder of the image. Our algorithm converts the RGB image to grayscale (Figure 2), and then filters the grayscale image using median filtering to reduce much of the noise present in the original image. We then convert this filtered grayscale image to black and white using a fixed threshold of 0.45, which worked well given the consistent lighting of the museum (Figure 3). We also implemented an adaptive thresholding method, but it was more computationally expensive and, for our training set, did not offer signif-
significant improvement over the fixed thresholding.

![Grayscale image](image1.png)

**Fig. 2.** Grayscale image

![Black and white image](image2.png)

**Fig. 3.** Black and white image, thresholded at 0.45.

After obtaining this black and white image and creating a mask, we perform region labeling on the white regions. Within all of the training images, the painting falls in the center of the image; therefore, to ensure that we select regions corresponding to the painting, we create a square around the center of the mask and choose all regions overlapping with this square. From these, we choose the region having the greatest number of pixels (Figure 4).

![Central region](image3.png)

**Fig. 4.** Central region with the largest number of pixels.

We then create a new image that has ones at all pixels corresponding to this label, and zeros elsewhere. This captures most of the painting, but contains a few small holes. To fill in the holes, we invert this image and perform region labeling on the background. At this point, we are interested in finding the region corresponding to the background/wall. In order to do this, we find the labels of the four corners of the image, and remove the pixels having these labels. A final mask is created by zeroing out these background pixels. Finally, the mask is applied to the original image (Figure 5).

![Final mask](image4.png)

**Final segmentation algorithm performs very well on the training images. The algorithm successfully segments out the wall, other paintings, statues, and any other unnecessary objects that may appear in the images. Our algorithm also ensures that the painting we extract from the original image lies in the center of the image, which was a valid constraint for our training and test sets. While shadows around the frames are sometimes retained in the segmented image, this did not seem to affect the overall performance of our classifier.**

We tried several different approaches before finalizing our segmentation algorithm. Our first
approach was to convert the image from RGB to HSV coordinates and use the saturation value to threshold the image. However, a problem with this approach was that the saturation values for the images were very noisy and could not be remedied even with filtering. Secondly, we tried edge detection, but this technique did not perform very well because several of the frames were quite fancy and did not have clear edges. A third approach, which we refer to as the “wall thresholding” method (Figure 6), was to find all of the R, G, and B pixels whose values lie within a window around the average R, G, and B values. Because these are shades of gray, we assumed that these pixels are part of the wall, and thus classified them as background pixels. On most of the training images, this method performed beautifully and even got rid of the shadows on some of the paintings. However, it failed badly on a few images, because both lighting and noise tended to affect the R,G,B values.

III. Color Histogram Matching

Overview

We used a color histogram approach to solve the painting identification problem. The image is first quantized to 100 colors, and histograms are computed over four quadrants to provide spatial localization. Histogram intersection is used as the similarity metric. This method demonstrated the best performance of all the methods we considered.

Methods

Before settling on our final color histogram algorithm, we tried many variations. We initially tried comparing the 1-D histograms of the R, G, and B components, using mean square error and correlation as performance metrics. This approach did not work well, however, because histogramming the R, G, and B components separately does not preserve information about the color as a whole. Since color is determined by an (R,G,B) triple, there are many different hues that have the same R value, for example, so separate histograms of R, G, and B do not accurately represent the color palette of an image.

One way to get around this issue would be to use a multidimensional histogram that counts the number of pixels that have each possible (R,G,B) triplet value. The R, G, and B values would have to be coarsely quantized so that the number of bins is computationally reasonable. An alternative option would be to quantize the number of
colors in the image based on a color palette, and translate each (R,G,B) triple into a single index into this palette. We chose this second option.

In building a color palette, it is important to pick a good, representative set of colors. Ideally, such a palette would consist of the minimum number of colors needed to accurately represent a set of images. Our approach was to use the rgb2ind function to quantize each of the images in the training set to 256 colors using minimum variance quantization. We stored the combined colors from each image into one color map, and then rounded and sampled to remove duplicates. The resulting set of 100 colors is shown in Figure 7.

Fig. 7. Color palette.

Another important factor to consider when using histograms for image comparison is spatial localization. A color histogram gives no information about the spatial distribution of colors in the image, so a method that relies on color histograms for identification would likely benefit from incorporating spatial information. Gavrielides et al. and Cinque et al. [1,2] describe methods that use spatial chromatic histograms. For each color $i$ in the quantized image, a vector $(h_i, b_i, \sigma_i)$ is defined, where $h_i$ is the number of pixels in the image with color $i$, $b_i$ is the coordinates of the centroid of color $i$, and $\sigma_i$ is a measure of the spread about this centroid of the pixels having color $i$. The metric used for comparing a query image with a training set image is histogram intersection, scaled by centroid and spread information. Since a vector has to be computed for each color in the quantized image, this method can be computationally slow. Also, since we are dealing with histograms over the segmented regions of the paintings, and since these regions are not all the same size, the centroid and spread measurements would have to be computed relative to local coordinates of the mask and scaled appropriately so they can be meaningfully compared.

Since the goal of spatial chromatic histograms is simply to encode some spatial information about the colors, we decided to use a simpler method of dividing the mask area into four quadrants (Figure 8), and computing histograms over those quadrants. This method works better than computing histograms over the entire mask (no spatial localization), and it performed comparably to division into nonants (9 sections).

Fig. 8. Illustration of quadrants for image in Figure 5.

We used histogram intersection [3] as our similarity metric. Given a test image histogram $T$ and a model training image histogram $M$, the intersection of these histograms is defined as:
\[
\sum_{j=1}^{m} \min(T_j, M_j)
\]
where \( j \) is an index over colors in the palette. The intersection gives the number of pixels in common between the model and test histograms. To form our final similarity metric, we normalized this intersection by the number of pixels in the model image. This value lies between 0 and 1, where 1 indicates a perfect match:

\[
H(T, M) = \frac{\sum_{j=1}^{m} \min(T_j, M_j)}{\sum_{j=1}^{m} M_j}
\]

Since the segmented paintings have different sizes, we scaled the test image histogram to have the same number of pixels as the model image histogram. This metric weights all color bins equally, without any consideration of how perceptually similar the colors are. An alternative metric, histogram quadratic distance [4], takes this weighting into consideration, but did not perform as well as histogram intersection. The formula is as follows:

\[
d_{\text{min}}(x, y) = (x - y)^T A(x - y)
\]
where \( a_{ij} = (1 - d_{ij}/d_{\text{max}}) \), \( d_{ij} \) is the Euclidean distance between colors \( i \) and \( j \) in the colormap, and \( d_{\text{max}} \) is the maximum distance between any two colors in the colormap.

Histogram intersection also performed better than mean-squared error, correlation, infinity norm, and a combination of histogram intersection and mean square error.

**Results and Discussion**

Using histogram intersection with four quadrants and our 100-color palette, we obtained 98.5\% accuracy. Table 1 shows the number of errors obtained with our final method, as well as the number of errors obtained with three other similarity metrics. Since we have six pictures of each of the 33 paintings, we divided the entire training set into six sets (each containing all 33 paintings). To evaluate our algorithm, we trained on each possible combination of five image sets and tested the remaining image set.

Even though the museum lighting is controlled, variations in lighting still exist across the painting area, and the color appears to vary somewhat depending on the distance and the angle at which the picture is taken. We tried a form of gray-world color balancing to see if this would help, but the results were actually worse.

**IV. Feature-based Recognition**

**Overview**

We also implemented a feature-based approach to recognition, based on the the concept of SIFT Features [5,6]. We identified scale-invariant features by finding local extrema in Difference-of-Gaussians (DoG) image pyramids, and built a local descriptor for each feature based on orientation histograms. We then experimented with several metrics for evaluating the similarity between features of two images.

**Methods Description**

Successively higher levels in the Gaussian image pyramid were obtained by smoothing the Gaussian image (\( \sigma = 1.5 \)) and then downsampling (bilinear interpolation) by a factor of 0.8. At each level, two Gaussian images (differing by a smoothing factor of \( \sqrt{2} \)) were produced and subtracted to obtain the DoG image. Extrema (peaks) were detected by identifying pixels that were greater than or less than all of their 8 neighbors. To reduce the number of noisy features, we only retained peaks whose magnitudes exceeded 60\% of the maximum magnitude in the DoG image. In addition, peaks that were likely to form parts of edges were eliminated by excluding those pixels whose ratio of principal curvatures exceeded a threshold value.

For each feature surviving the above criteria, we constructed a local descriptor by building orientation histograms, as described in [6]. Within a 16x16 window around the feature, gradient magnitudes were weighted by a Gaussian window (\( \sigma = 7 \)) and accumulated into 8-bin orientation
TABLE I

ERROR COMPARISON ACROSS SIMILARITY METRICS FOR COLOR HISTOGRAMS

<table>
<thead>
<tr>
<th>Image Set</th>
<th>Histogram Intersection</th>
<th>MSE</th>
<th>Correlation</th>
<th>Histogram Quadratic Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>Total Errors (out of 198)</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>Percent Correct</td>
<td>98.5</td>
<td>98.0</td>
<td>97.4</td>
<td>95.0</td>
</tr>
</tbody>
</table>

histograms. To save computation time, the gradient magnitude and orientation images at all levels were pre-computed. Just as in [6], we made a different histogram for each of four 4x4 pixel regions within the 16x16 window. The values of each histogram bin were concatenated into a 128-entry vector, which was then normalized to unit length in order to increase the robustness against changes in illumination.

To identify a new test image, we computed a pairwise similarity metric between the feature set belonging to the test image and the features sets of each of the images in the training set. The training set image with the highest similarity determined the class (painting) of the test image. Since we were interested in recognizing entire images, rather than specific objects within an image, we only needed to be concerned with the overall similarity between features sets of two images rather than explicitly finding the correspondences.

The similarity between two features \( f_1 \) and \( f_2 \) was defined as \( \sqrt{2} - d(f_1, f_2) \), where \( d(f_1, f_2) \) is the Euclidean distance. The nearest neighbor of a feature in Image 1 is defined as the feature in Image 2 having the highest similarity.

We tried several ways of defining the similarity between sets of features from two different images, including:

1. For each feature in Image 1, compute the similarity with its nearest neighbor in Image 2. Take the sum of these similarities.
2. Same as (1), but sum over only the \( N \)-percent highest similarity values. \( N \) between 25 and 50 usually gave the best results.
3. For each feature in Image 1, find the ratio between the similarities of its nearest neighbor and the second-nearest neighbor. Then, take the sum of these ratios.

**Results and Discussion**

At best, this approach misclassified 10 of the 99 images in the original training set – worse than the best results using color histograms. In addition, the computation time was about 10 seconds slower than the color histogram method. Therefore, we chose not to use the feature-detection approach in our final submission.

There are several reasons why this feature-based approach did not achieve perfect accuracy. One reason concerns the choice of optimal parameters for the algorithm. We tried a range of values for parameters such as (1) the distance between scales on the image pyramid (0.8 usually gave the best results) and (2) the threshold for the magnitude of peaks retained in the peak-detection step (60% ended up being a good value). However, we did not do a very fine-grained search for optimal parameters, and this may have affected the accuracy significantly. Indeed, we found that different sets of parameters had a large impact on the accuracy.

For the similarity metrics, options (1) and (2) performed significantly better than (3).
The lack of rotation-invariance in our implementation may have also made a difference. Furthermore, for simplicity we also chose not to implement several other optimizations described in Lowe’s paper, such as fitting quadratic functions to localize the features in both image- and scale-space with greater accuracy.

V. Conclusion

We successfully ID’d 98.5% images in our training set using a color-histogram approach. Here is a beautiful reflection of our success, found at the Cantor Arts Center:

![Image](image-url)

Fig. 9. idididididid.

VI. References


VII. Group Participation

Catie focused on coding the feature detection algorithm, and Mary & Michelle jointly concentrated on programming the color histogram approach. Everything else was equally divided.