Recognizing pictures at an exhibition using SIFT

Rahul Choudhury
Biomedical Informatics Department
Stanford University
EE 368 Project Report

Abstract—**To aid visitors of art centers in automatically identifying an object of interest, a computer vision algorithm known as SIFT has been implemented using MATLAB. Salient aspects of the algorithm have been discussed in light of image recognition applications. Test methods to validate the algorithm have been detailed out. Performance improvement for SIFT also has been proposed.**

1. INTRODUCTION

Museums and art centers present objects of interests, which contain important domain-specific information that visitors are interested in. Art exhibition centers usually provide a guide or a booklet, which visitors manually scan to find out general information about an object of interest. However this manual and passive method of information retrieval is tedious and interrupts visitor's art-viewing experience. The goal of this project is to develop an automated software-based technique, which can recognize paintings on display in an art-center based on snapshots taken with a camera-phone.

2. BACKGROUND

Comparing images in order to establish a degree of similarity is an important computer vision problem and has application in various domains such as interactive museum guide, robot localization, content-based medical image retrieval, and image registration. Comparing images remains a challenging task because of issues such as variation in illumination conditions, partial occlusion of objects, differences in image orientation etc. Global image characteristics such as color histograms, responses to filter banks etc. are usually not effective in solving real-life image matching problems.

Researchers have recently turned their attention to local features in an image, which are invariant to common image transformations and variations. Usually two broad steps are found in any local feature-based image-matching scheme. The first step involves detecting features (also referred to as keypoints or interest points) in an image in a repeatable way. Repeatability is important in this step because robust matching cannot be performed if the detected locations of keypoints on an object vary from image to image. The second step involves computing descriptors for each detected interest point. These descriptors are useful to distinguish between two keypoints.

The goal is to design a highly distinctive descriptor for each interest point to facilitate meaningful matches, while simultaneously ensuring that a given interest point will have the same descriptor regardless of the object's pose, the lighting in the environment, etc. Thus both detection and description steps rely on invariance of various properties for effective image matching.

Image matching techniques based on local features are not new in the computer vision field. Van Gool [5] introduced the generalized color moments to describe the shape and the intensities of different color channels in a local region. Sven Siggelkow [2] used feature histograms for content-based image retrieval. These methods have achieved relative success with 2D object extraction and image matching. Mikolajczyk and Schmid [3] used the differential descriptors to approximate a point neighborhood for image matching and retrieval. Schaffalitzky and Zisserman [4] used Euclidean distance between orthogonal complex filters to provide a lower bound on the Squared Sum Differences (SSD) between corresponding image patches. David Lowe proposed [1] Scale Invariant Feature Transforms (SIFT), which are robustly resilient to several common image transforms. Mikolajczyk and Schmid [6] reported an experimental evaluation of several different descriptors where they found that the SIFT descriptors obtain the best matching results.

3. SELECTION OF LOCAL FEATURES

The following requirements were key in selecting a suitable local-feature for images used in this project:

(a) Invariance: The feature should be resilient to changes in illumination, image noise, uniform scaling, rotation, and minor changes in viewing direction
(b) Highly Distinctive: The feature should allow for correct object identification with low probability of

mismatch.

(c) Performance: Given the nature of the image recognition problem for an art center, it should be relatively easy and fast to extract the features and compare them against a large database of local features.

4. BRIEF DESCRIPTION OF THE ALGORITHM

Based on the requirements mentioned in the previous section and the reported robustness in [6], I selected the Scale Invariant Feature Transform (SIFT) [1] approach for this project. SIFT is an approach for detecting and extracting local feature descriptors that are reasonably invariant to changes in illumination, image noise, rotation, scaling, and small changes in viewpoint. Before performing any multi-resolution transformation via SIFT, the image is first converted to grayscale color representation.

A complete description of SIFT can be found in [1]. An overview of the algorithm is presented here. The algorithm has four major stages as mentioned below:

- Scale-space extrema detection: The first stage searches over scale space using a Difference of Gaussian function to identify potential interest points.
- Keypoint localization: The location and scale of each candidate point is determined and keypoints are selected based on measures of stability.
- Orientation assignment: One or more orientations are assigned to each keypoint based on local image gradients.
- Keypoint descriptor: A descriptor is generated for each keypoint from local image gradients information at the scale found in the second stage.

Each one of the above-mentioned stages is elaborated further in the following sections.

A. *Scale-space Extrema Detection*

The SIFT feature algorithm is based upon finding locations (called keypoints) within the scale space of an image which can be reliably extracted. These keypoints correspond to local extrema of difference-of-Gaussian (DoG) filters at different scales. The DoG image is represented as $D(x, y, \sigma)$ and can be computed from the difference of two nearby scaled images separated by a multiplicative factor k:

$$D(x, y, \sigma) = \left(G(x, y, k\sigma) - G(x, y, \sigma)\right) * I(x, y)$$
$$= L(x, y, k\sigma) - L(x, y, \sigma)$$

where $L(x, y, \sigma)$ is the scale space of an image, built by convolving the image $I(x, y)$ with the Gaussian kernel $G(x, y, \sigma)$. The convolved images are grouped by octave. An octave corresponds to doubling the value of σ, and the value of k is selected so that a fixed number of blurred images are generated per octave. This also ensures the same number of DoG images are obtained per octave. Keypoints are identified as local maxima or minima of the DoG images across scales. Each pixel in a DoG image is compared to its 8 neighbors at the same scale, and the 9 corresponding neighbors at neighboring scales. If the pixel is a local maximum or minimum, it is selected as a candidate keypoint.

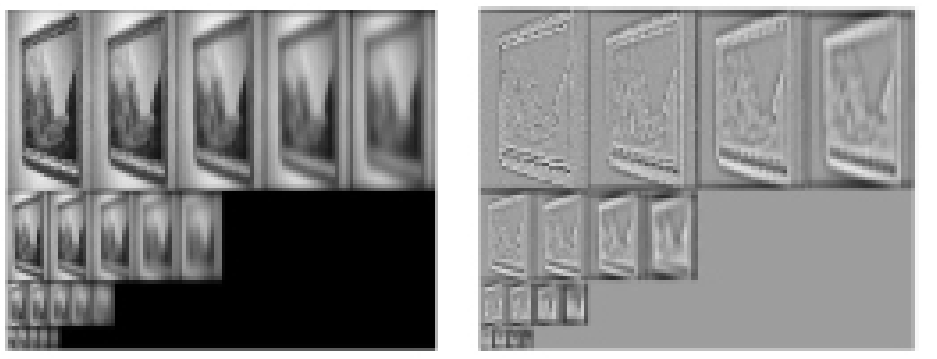

Figure 1: Gauss Pyramid (left) and DoG (right) for a sample training image

B. *Keypoint Localization*

In this step the keypoints are filtered so that only stable and more localized key points are retained. First a 3D quadratic function is fitted to the local sample points to determine the location of the maximum. If it is found that the extremum lies closer to a different sample point, the sample point is changed and the interpolation performed instead about that point. The function value at the extremum is used for rejecting unstable extrema with low contrast.

The DoG operator has a strong response along edges present in an image, which give rise to unstable key points. A poorly defined peak in the DoG function will have a large principal curvature across the edge but a small principal curvature in the perpendicular direction. The principal curvatures can be computed from a 2x2 Hessian matrix H, computed at the location and scale of the keypoint. H is given by $H = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{yx} & D_{yy} \end{bmatrix}$. The eigenvalues of H are proportional to the principal curvatures of D. However eigenvalues are not explicitly computed; instead trace and determinants of H are used to reject those keypoints for which the ratio between principal curvatures is greater than a threshold.

Figure 2: Keypoints after removing low-contrast interest points

Figure 3: Stable keypoints after removing the edge elements

C. Orienation Assignment

In this step each keypoint is assigned an orientation to make the descriptor invariant to rotation. This keypoint orientation is calculated from an orientation histogram of local gradients from the closest smoothed image $L(x, y, \sigma)$. For each image sample $L(x, y)$ at this scale, the gradient magnitude $m(x, y)$ and orientation $\theta(x, y)$ is computed using pixel differences:

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}$$

$$\theta(x,y) = \tan^{-1}\left(\frac{L(x,y+1) - L(x,y-1)}{L(x+1,y) - L(x-1,y)}\right)$$

The orientation histogram has 36 bins covering the 360 degrees of major orientation bins. Each point is added to the histogram weighted by the gradient magnitude $m(x, y)$ and by a circular Gaussian with σ that is 1.5 times the scale of the keypoint. Additional keypoints are generated for keypoints with multiple local dominant peaks whose magnitude is within 80% of each other. The dominant peaks in the histogram are interpolated with their neighbors for a more accurate orientation assignment.

D. Keypoint Descriptor

The local gradient data from the closest smoothed image $L(x, y, \sigma)$ is also used to create the keypoint descriptor. This gradient information is first rotated to align it with the assigned orientation of the keypoint and then weighted by a Gaussian with σ that is 1.5 times the scale of the keypoint. The weighted data is used to create a nominated number of histograms over a set window around the keypoint. Typical keypoint descriptors use 16 orientation histograms aligned in a 4x4 grid. Each histogram has 8 orientation bins each created over a support window of 4x4 pixels. The resulting feature vectors are 128 elements with a support window of 16x16 scaled pixels.

5. KEYPOINTS MATCHING

Given a test image, each one of its keypoint is compared with keypoints of every image present in the training database. Euclidean distance between each invariant feature descriptor of the test image and each invariant feature descriptor of a database image is computed at first. However two keypoints with the minimum Euclidean distance (the closest neighbors) cannot necessarily be matched because many features from an image may not have any correct match in the training database (either because of background clutter or may be the feature was not detected at all in the training images). Instead the ratio of distance between closest neighbors and distance between the second closest neighbor is computed. If the ratio of distances is greater than 0.6, then that match is rejected. Once the comparison is performed with all test images in the database, the title of the database image with the maximum number of matches is returned as the recognized image.

6. IMPLEMENTATION

The approach described above has been implemented using MATLAB. The implementation has two aspects: training and inference. During the training phase locally invariant features (keypoints, orientation, scales and descriptors) from all training images are retrieved using