IJCSN International Journal of Computer Science and Network, Volume 5, Issue 1, February 2016
ISSN (Online) : 2277-5420 www.IJCSN.org
Impact Factor: 1.02

Large Scale Image Searching Using Two Dimensional Binary Trees Combined with Feature Selection

[1] Radhakrishnan B., [2] Anver Muhammed K.M

[1] Assistant Professor, Baselios Mathews 2 College of Engineering
Muthupilakkadu P.O, Sasthamcotta

[2] System Administrator, Baselios Mathews 2 College of Engineering
Muthupilakkadu P.O, Sasthamcotta

Abstract - The challenging task in image retrieval is to find similar images based on the distance between images. Content based image retrieval system is used to identify the similarity between multidimensional image feature vectors. If the database is large, searching the image linearly will be too slow. We have proposed the two dimensional binary tree approach for searching the images based on feature detectors and they have improve the retrieval efficiency significantly.

Keywords – *CBIR, Feature Detectors, Hessian Detector, SIFT.*

1. Introduction

The growing amount of digital images caused by the more and more ubiquitous presence of digital cameras and social networking sites confronts the users with new problems. Images are a fundamental part of our daily communication. The huge amount of data available is not easy to manage and effective retrieval of the images is really a difficult task. By entering the keywords in Google, related documents in the internet can be searched for but in the case of image search the results are not satisfactory. Rather than saving the image textual description of the image are stored in the database. The images can be searched in the database by using text based information retrieval methods, based on the textual descriptions. But it is a tedious task to be done practically because of the rich information in each images, so no textual description can absolutely define an image. On the other hand indexing each image have to be done manually and is not feasible in the case of billions of images in the internet. Content-based image retrieval (CBIR) is another image retrieval approach. Instead of using text annotations it works through the direct use of image content. With this, CBIR systems may be fully automated in the indexing of the image database and free from human dependence.

2. CBIR

Content-based image retrieval combines many disciplines. In one case there are image recognition and pattern analysis which describe the images in a way that makes it easy to distinguish between similar and dissimilar images. On the other hand there is the database part which tries to store the images and features to allow for efficient access to the data. In general, users search for images from internet by submitting a query image. CBIR system evaluates the similarity between the query image and database images and returns those with good similarity. CBIR systems uses features like color and texture for image indexing and retrieval as well as shape information, spatial relationship between regions, salient points and so on. Feature detection is a low-level image processing operation. A feature is defined as an interesting part of an image, and features are used as a starting point for many computer vision algorithms. It is usually performed as the first operation on an image, and examines every pixel to see if there is a feature present at that pixel, then later on searching algorithms will only examine the region of the features. As a built-in pre-requisite to feature detection, the input image is usually smoothed by a Gaussian kernel in a scale-space representation.

2.1 Edges

Edges are points where there is a boundary (or an edge) between two image regions. In general, an edge can be of almost arbitrary shape, and may include junctions. In

practice, edges are usually defined as sets of points in the image which have a strong gradient magnitude.

2.2 Corners / Interest Points

The terms corners and interest points are used somewhat interchangeably and refer to point-like features in an image, which have a local two dimensional structure. The name "Corner" arose since early algorithms first performed edge detection, and then analyzed the edges to find rapid changes in direction.

2.3 Blobs / Regions of Interest or Interest Points

Blobs provide a complementary description of image structures in terms of regions, as opposed to corners that are more point-like. Nevertheless, blob descriptors may often contain a preferred point (a local maximum of an operator response or a center of gravity) which means that many blob detectors may also be regarded as interest point operators. Blob detectors can detect areas in an image which are too smooth to be detected by a corner detector.

3. Feature Detection

1. Global features: where the image is represented by one multi-dimensional feature, describing the information in the image. The information can be color histograms, edge magnitude or orientation histograms or a specific descriptor extracted from some filters applied to the image, such as GIST features. The global features are advantages in the sense that they are fast and easy to compute and generally require small amounts of memory. However, when compared with its counterpart Local Features its performance is worse.

2. Local features: where the image is represented by a set of local feature descriptors extracted from a set of regions around the image. There are generally two components for local features: a feature detector and a feature descriptor. A feature detector detects interesting locations in the image, for example corners and edges. A feature descriptor describes the image patch around that interest point, usually by histograms of gradients or orientation. There are different kinds of feature detectors, but most commonly used are Difference of Gaussians (DoG), Hessian- Affine, and Harris Affine. Similarly there are various feature descriptors, but by far SIFT and HOG are the most widely used. The main advantage of local features is their superior performance when compared with global features. But each image have hundreds of local features so they need large memory usage.

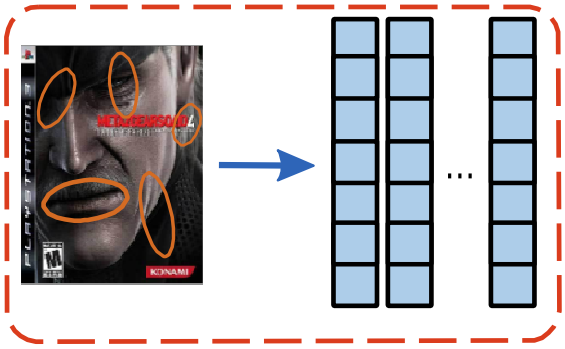

Fig 1:Local features represent an image by a set of data structures

The purpose of local invariant features is to provide a representation that allows to efficiently match local structures between images. That is, we want to obtain a sparse set of local measurements that capture the essence of the underlying input images and that encode their *interesting* structure. To meet this goal, the feature extractors must fulfill two important criteria:

- The feature extraction process should be *repeatable and precise*, so that the same features are extracted on two images showing the same object.
- At the same time, the features should be *distinctive*, so that different image structures can be told apart from each other.

The steps involved are:

1. Find a set of *distinctive key points*.
2. Define a region around each key point in a *scale- or affine-invariant* manner.
3. Extract and *normalize* the region content.
4. Compute a *descriptor* from the normalized region.
5. Match the local descriptors.

The extraction procedure should yield the same feature locations if the input image is translated or rotated. It is obvious that those criteria cannot be met for all image points. For instance, if we consider a point lying in a uniform region, we cannot determine its exact motion, since we cannot distinguish the point from its neighbors. Similarly, if we consider a point on a straight line, we can only measure its motion perpendicular to the line. This motivates us to focus on a particular subset of points, namely those exhibiting signal changes in two directions. We have selected a key point detector that employs

different criteria for finding such regions: the *Hessian detector*.

3.1 The Hessian Detector

The *Hessian detector* searches for image locations that exhibit strong derivatives in two orthogonal directions. It is based on the matrix of second derivatives, the so-called Hessian:

$$H(X,\sigma)=\begin{pmatrix} I_{xx}(X,\sigma) & I_{xy}(X,\sigma) \\ I_{xy}(X,\sigma) & I_{yy}(X,\sigma) \end{pmatrix} \quad (1)$$

The detector computes the second derivatives I_{xx}, I_{xy}, and I_{yy} for each image point and then searches for points where the determinant of the Hessian becomes maximal:

$$\det(H) = I_{xx}I_{yy} - I^2{}_{xy} \quad (2)$$

Once a set of interest regions has been extracted from an image, their content needs to be encoded in a descriptor that is suitable for discriminative matching. The most popular choice for this step is the SIFT descriptor.

3.2 The SIFT Descriptor

The *Scale Invariant Feature Transform* (SIFT) was originally introduced by Lowe as combination of a DoG interest region detector and a corresponding feature descriptor. This descriptor aims to achieve robustness to lighting variations and small positional shifts by encoding the image information in a localized set of gradient orientation histograms. The descriptor computation starts from a scale and rotation normalized region extracted from the detector. As a first step, the image gradient magnitude and orientation is sampled around the key point location using the region scale to select the level of Gaussian. Sampling is performed in a regular grid of 16×16 locations covering the interest region. For each sampled location, the gradient orientation is entered into a coarser 4×4 grid of gradient orientation histograms with 8 orientation bins each, weighted by the corresponding pixel's gradient magnitude and by a circular Gaussian weighting function with a σ of half the region size. The purpose of this Gaussian window is to give higher weights to pixels closer to the middle of the region, which are less affected by positional shifts.

The motivation for this choice of representation is that the coarse spatial binning allows for small shifts due to registration errors without overly affecting the descriptor. At the same time, the high-dimensional representation provides enough discriminative power to reliably distinguish a large number of key points. When computing the descriptor, it is important to avoid all boundary effects, both with respect to spatial shifts and to small orientation changes. Thus, when entering a sampled pixel's gradient information into the 3-dimensional spatial orientation histogram, its contribution should be smoothly distributed among the adjoining histogram bins using trilinear interpolation. Once all orientation histogram entries have been completed, those entries are concatenated to form a single 4 × 4 × 8 = 128 dimensional feature vector. A final illumination normalization completes the extraction procedure. For this, the vector is first normalized to unit length, thus adjusting for changing image contrast. Then all feature dimensions are thresholded to a maximum value of 0.2 and the vector is again normalized to unit length. This last step compensates for non-linear illumination changes due to camera saturation or similar effects. The definition of what constitutes points as being "close" is critical. One commonly-used family of metrics are the Lp norms. For points a and b of dimensionality D, these are defined as . The Lp norms for p = 1 (Manhattan distance) and p = 2 (Euclidean distance) are especially common when comparing image patches. Here, we take advantage of the fact that images typically contain 8-bit quantized values. We use a 2-dimensional lookup table to store the result of $|a - b|^p$ for all possible values of a and b. This can be done during program initialization and saves D subtractions and D · (p−1) multiplications per distance function evaluation. Likewise, the expensive p-th root calculation can be stored in a lookup table if D and p are small enough.

4. Searching Using Two Dimensional Search Trees

Consider the 2-dimensional rectangular range searching problem. Let P be the set of n points in the plane. The basic assumption is no two point have same x-coordinate, and no two points have same y-coordinate. Let's consider the following recursive definition of the binary search tree : the set of (1-dimensional) points is split into two subsets of roughly equal size, one subset contains the point smaller than or equal to splitting value, the other contains the points larger than splitting value. The splitting value is stored at the root and the two subsets are stored recursively in two sub trees. Each point has its x-coordinate and y-coordinate. Therefore we first split on x-coordinate and then on y-coordinate, then again on x-coordinate, and so on. At the root we split the set P with

vertical line l into two subsets of roughly equal size. This is done by finding the median x- coordinate of the points and drawing the vertical line through it. The splitting line is stored at the root. P_{left}, the subset of points to left is stored in the left sub tree and P_{right}, the subset of points to right is stored in the right sub tree. At the left child of the root we split the P_{left} into two subsets with a horizontal line. This is done by finding the median y-coordinate if the points in P_{left}. The points below or on it are stored in the left sub tree, and the points above are stored in right sub tree. The left child itself stores the splitting line. Similarly P_{right} is split with a horizontal line, which are stored in the left and right sub tree of the right child. At the grandchildren of the root, we split again with a vertical line. In general, we split with a vertical line at nodes whose depth is even, and we split with horizontal line whose depth is odd. Considering a collection of feature vectors as point set $A=l_1,l_2\ldots.l_n$ in the search tree, retrieving image involves searching for images similar to a given query. The nearest neighbor is retrieved by finding point l_k in A which is most similar to the given query that means whose distance measure is the minimum

$$d(a,b)=\left(\sum_{i}^{D} \left| a_i - b_i \right|^{p}\right)^{1/p} \quad (3)$$

We measured the response time of the system to the user and then gave a breakdown of the time into initialization ie the program set up time before queries can be made; time spent searching the tree and the response time (time including initialization, search and display of results). In table 1 the times are given using linear search and the two dimensional search trees respectively.

Table1: Search Response Time: Average results of 20 test queries per algorithm.

Method	Feature	Initialization	Search	Sum	Response Time
Linear Search	Color	0.21	7.1	7.3	7.5
	Texture1	0.21	10.7	10.9	12.3
Tree Search	Color	0.68	2.3	3.0	3.1
	Texture1	0.68	3.0	3.7	3.9

5. Conclusion

Solution of finding the similar image items requires solving the nearest neighbor problem. Two dimensional binary search trees are an efficient method for finding nearest neighbors. In this paper we have adapted the search trees to the problem of image retrieval and found the best parameters regarding minimizing the access time. Threshold value can be approximated by a logarithmic function of the number of feature vectors.

References

[1] Mohamed Aly. Online Learning for Parameter Selection in Large Scale Image Search. In CVPR Workshop OLCV, June 2010.

[2] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. Commun. ACM, 51(1):117–122, 2008.

[3] S. Arya, D.M. Mount, N.S. Netanyahu, R. Silverman, and A.Y. Wu. An optimal algorithm for approximate nearest neighbor searching. Journal of the ACM, 45:891–923, 1998.

[4] R. Baeza-Yates and B. Ribeiro-Neto. Modern Information Retrieval. ACM Press,1999.

[5] Andrei Broder, Moses Charikar, and Michael Mizenmacher. Min-wise independent permutations. Journal of Computer and System Sciences, 60:630–659, 2000.

[6] Andrei Z. Broder, Steven C. Glassman, Mark S. Manasse, and Geoffrey Zweig. Syn- tactic clustering of the web. Computer Networks and ISDN Systems, 29:8–13, 1997.

[7] A.Z. Broder. On the resemblance and containment of documents. In Proc. Compres- sion and Complexity of Sequences 1997, pages 21–29, 1997.

[8] J. Sivic and A. Zisserman. Video google: A text retrieval approach to object matching in videos. In ICCV, 2003.

[9] M. Muja and D. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In VISAPP, 2009.

Authors Profile

Radhakrishnan B is working as Asst. Professor in Computer Science department. He has more than 14 years experience in teaching and has published papers on data mining and image processing. His research interests include image processing, data mining, and image mining

Anver Muhammed K.M is working as System Administrator in Computer Science department. He has more than 10 years experience in System Administration and Implementation. His research interests include image processing and network security.