SHOPPING BY EXAMPLE
A New Shopping Paradigm in Next Generation Retail Stores

Ashish Khare, Hiranmay Ghosh
TCS Innovation Labs, Tata Consultancy Services Ltd
Plot No 249 D&E, Udyog Vihar, Phase – IV, Gurgaon, India

Jaideep Shankar Jagannathan
Freescale SemiconductorIndia Ltd, Express Trade Tower
Floor5,6,7, Plot 15 & 16, Sector 16-A, Noida, India

Keywords: Online shopping, Query-by-Example, Content-based Image Retrieval, SIFT, PCA-SIFT, Image Features Indexing, R-Tree.

Abstract: In this paper, we present a new example based approach to search for a particular product based on its visual properties. A user can take a photo of a product package with a cell-phone or webcam and submit it to an online shopping portal for finding the product details. We search a product image database for the distinctive visual features on the query image to locate the desired product. We use PCA-SIFT feature for robust retrieval, to account for possible imperfections in the query image due to uncontrolled user environment. We use Oracle Java R-Tree to index image features to realize a scalable system. We establish robustness and scalability of our approach by conducting several experiments on fairly large prototype implementations.

1 INTRODUCTION

Over the past few years the online shopping trend has continued to grow at a healthy clip. According to survey conducted by Nielsen (Nielsen, 2008) over 85 percent of world's online population has shopped online in last two years. The reasons are obvious – 24 x 7 virtual online stores, no pushy sales staff, no busy checkout line-ups, no parking hassles etc. Today, several successful online stores proliferate over the Internet, ebay.com, amazon.com, etc being a few examples.

Despite the advantages online shopping faces several challenges. Significant among them is the difficulty in identifying a particular product when its exact details are not known or difficult to specify for a customer. For instance, a certain brand of chocolate which may come in several different flavours and it may be difficult for a customer to remember the exact one he/she wants. However the variants of the chocolate are uniquely distinguished by the visual appearance of their packaging and a user generally looks for the visual pattern while exploring the shelves of a brick-and-mortar retail store. Such facility has not yet been made available on the online shopping portals. We present a new shopping solution that enables a user search a shopping portal for the desired product with a photo of the product package snapped using a digital camera. The solution employs well-known "query-by-example" paradigm of content-based image retrieval. We call the solution "*Shopping by Example*" to reflect the query paradigm.

While visual pattern matching technique seems quite intuitive for product selection, it provides quite a few challenges. Content-based image retrieval techniques are based on inexact match or similarity measures, rather than exact match principles used in the database search. The query images in a shopping application are snapped by a user in an uncontrolled environment resulting in significant distortions. Exact identification of the intended product is very difficult in this noisy environment. Use of content-based retrieval in shopping has been reported in (Ghosh & Chaudhury, 2002) and (Tollmar *et al*, 2007). Both of these systems aim at exploring the

Khare A., Ghosh H. and Jagannathan J. (2009).
SHOPPING BY EXAMPLE - A New Shopping Paradigm in Next Generation Retail Stores
.
In *Proceedings of the Fourth International Conference on Computer Vision Theory and Applications*, pages 48-53
DOI: 10.5220/0001788600480053

product space with example images (and other descriptors) rather than uniquely identifying a specific product.

Our main contribution is to implement a practical shopping solution that can be used for example based search for a specific product in a large product database. There are three major aspects in this work: (a) selection of a suitable image feature that can uniquely identify a product from a large collection despite the undesired transformations of the query image, (b) to achieve scalability by using a suitable indexing scheme for the image feature and (c) to devise a method to use these methods to create a practical system. We have been constrained in selecting the image feature and the indexing methods by the fact that there are not many methods that scale up to cope up with thousands of unique products that may exist in a retail store.

The rest of the paper is organized as follows - Section 2 provides the critical review of CBIR methods towards present application. Section 3 describes the system in detail. Section 4 describes some experimental results that establish robustness and scalability of the system. Section 5 concludes the paper.

2 A CRITICAL REVIEW OF CBIR

Shopping-by-Example is essentially a Content Based Image Retrieval (CBIR) implementation. It searches a product image database with a query image and produces the best matching result. The technical challenge in the system is to retrieve the desired product image uniquely, even when a query image suffers significant distortion. Another challenge is to scale the solution for a large product database. To address the first we have used PCA-SIFT (Ke & Sukthankar, 2004) algorithm which is an advanced version of Scale Invariant Feature Transform (SIFT) (Lowe, 2004) algorithm. SIFT extracts a set of distinctive image features that are invariant to several distortions and can be used to perform robust matching of images representing different views of an object. But a drawback of SIFT features is that huge number of keypoints are generated corresponding to an image resulting in large storage and computational overheads. PCA-SIFT remove this deficiency by using Principal Component Analysis (PCA) on keypoint descriptors. After significant experimentation with SIFT, PCA-SIFT and other image features, we have selected PCA-SIFT as the feature descriptor to characterize the distinctive product marks for their robustness against image distortions, relatively compact representation and reduced comparison time requirements. As retail stores generally contain thousands of products, it is not practical to compare a query image with all the images in the product database to find the best matching result. There are two distinct techniques to ensure fast access to desired images in the database. In one approach, the database can be clustered by using some unsupervised learning mechanism, such as k-means clustering (MacQueen, 1967) or DBScan clustering (Ester *et al*, 1996). This approach works well when there are natural well-separated clusters in the data. The PCA-SIFT keypoints for the product images used in our solution do not exhibit such natural clustering behavior. The second approach, involves indexing of multidimensional data. After significant experimentation with different variants of R-Tree (Guttman, 1984) and X-Tree (Berchtold *et al, 1996*) algorithms, we have selected Oracle Java R-Tree implementation (Oracle, 2007) for indexing PCA-SIFT keypoints. A major advantage of this memory resident implementation is very fast access time.

3 SYSTEM DESCRIPTION

3.1 Creating Product Database

The traditional product parameters, such as the name, product code, quantity and the price are stored in a conventional relational database in Shopping-By-Example system. In addition, we create an image feature index of the product packages. This is done in two steps.

Feature Extraction. At this step, we extract the PCA-SIFT features of the images. The product images which populate the product database can be of different sizes. Selection of a proper image size before feature extraction is necessary. Using too large images result in unstable key points whereas using too small images relevant key points are not selected. We scale all product images to an optimal size as an image pre-processing stage. Next, we convert the images to gray-scale (PGM format) from which PCA-SIFT features are extracted.

Index Construction. The PCA-SIFT feature of every product in the database comprises several keypoints, each of which is a high-dimensional vector. We index each of these keypoints using Oracle Java R-Tree implementation. Since all the keypoints are pre-computed and known in advance, we have bulk inserted them in the database. Bulk

insertion not only reduces insertion time, but also forms a well balanced R-Tree structure (Bercken *et al*, 1997) thereby improving search performance. In general, a data item in Oracle Java R-Tree is represented by a hyper-solid in the *n*-dimensional space by specifying the lower and the upper bound coordinates. However, we have defined each key-point as an ideal point, i.e. with identical upper and lower bound coordinates. The R-Tree implementation assigns a unique *id* to every indexed keypoint. We create a hash-table to create a *many-to-one* map of these keypoints to their respective product *id*s.

3.2 Query Processing

A basic query processing model of our work is shown in the Figure 1. The whole process can be broadly described by the following steps: -

Feature Extraction. The query image submitted by the user is normalized to a specific size, converted into gray-scale format before the PCA-SIFT keypoints are extracted as in the case of populating the database.

R-Tree Search. The keypoints in the query image do not have identical descriptors as in the corresponding database images because of distortions in the query image. However, it is assumed that many of them are sufficiently close in the feature space to be retrieved with a range search in the product image key-point database.

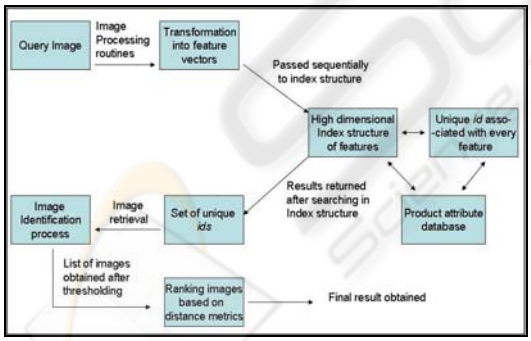

Figure 1: A basic query processing model.

In order to search the keypoints that lie close to the query keypoints, we define a hyper-solid around each query point and submit it as a query in the R-Tree implementation. This hyper-solid is referred to as the Minimum Bounding Hyper-solid (MBH). In a large product database, several other keypoints, besides the intended one, are likely to lie close to the query key-point within this hyper-solid. All keypoints which fall in the hyper-solid are retrieved during the search. The situation is shown in Figure 2 by taking example in two dimensions. In Figure 2, each of the dots represents a key-point in the database. The dots drawn with a particular style belongs to the same product image. Thus a search with every keypoint in the query image yields a set of other keypoints belonging to the different product images. Mapping the retrieved keypoints to the corresponding products provides a set of product images P_i for every keypoint K_i in the query image. Since at most one keypoint for a product image may map to one keypoint of the query image, we remove any duplicates (e.g. see diamonds in Figure 2) from this set. Thus, for all keypoints in the query image we get a list of set of matching keypoints $<M_1, M_2 ... M_N>$ where N denotes the number of keypoints in the query image. The union of these sets $\cup_i M_i$ denote the set of candidate products.

Filtering. For every product $m_i \in \cup_i M_i$, we count the number of its occurrence n_i in the list $<M_1, M_2 ... M_n>$, which signifies the number of matching keypoints for that product image with the query image. We sort the product images in descending order of matches to get a list $<(m_i,n_i)>$, where $m_i \in \cup_i M_i$ and $n_1 > n_2 > n_3$ ... and so on.

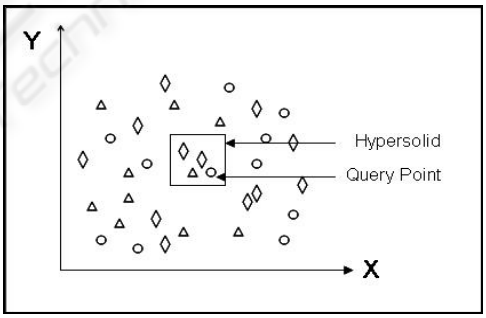

Figure 2: Key points in 2-d region are shown.

A naive solution could produce the product from the top of the list as the solution, since the maximum number of its keypoints can be assumed to match with the image of the target product. However, condition may not be satisfied in a large product database and distorted query image. Identification of the target product requires comparison of the query image with each of the candidate solutions. The list $<(m_i,n_i)>$ can be quite long in a large product database. In order to restrict the expensive comparison algorithm, we drop the spurious candidate products, which are characterized by significantly less number of matching keypoints using some heuristics. We compare every n_i with n_{i+1}

in the list, and remove all images m_i for i> k, when $n_{k+i} < \mu.n_k$. Thus we get a smaller list $< (m_i,n_i)_{i=1,k}>$ for processing.

Ranking. The final stage of query processing involves comparing the product images in the list $<(m_i,n_i)_{i=1,k}>$ with the query image by PCA-SIFT feature comparison algorithm. A product is retained in the list only if the match value exceeds a certain threshold η. If none of the images qualify at this stage, we declare the search to be unsuccessful. The PCA-SIFT features are distinctive enough to make a good discrimination in the matching scores of the target and the other product images. However, if two or more products are characterized by very similar product-marks, it may not be possible to distinguish between the two. Thus, we use the following heuristics to identify the solution. As in the last stage, we arrange the product images in order of descending match scores s_i and define a cut-off at $s_{k+i} < \tau.s_k$ and select the product images m_i as the final solution. If there are too many products in this list, we assume that the system could not identify the product and declare the search to be unsuccessful.

Discussions. The query image could be compared with all product images in the database using PCA-SIFT based feature to give the same results. However, PCA-SIFT based comparison algorithm is very complex ($O(n^2)$) and it is not possible to build a scalable system using this method. The R-Tree indexing technique results in short-listing of a set of candidate solutions, which significantly reduces the overall search time. The multi-stage filtering of the candidate solutions further optimizes the search time. If the cardinality of candidate solution list falls to one at any stage of filtering, that product is treated as the solution without any further processing. Similarly, if none of the database images qualify at any of the filtering stages, we conclude that the product is not in the database and the search terminates. The optimal threshold values like μ, η and τ are found from several experimental observations.

The choice of the MBH size is critical for the performance of the system. A large value of MBH generates a lot of spurious keypoints, which results in larger processing time in subsequent filtering and ranking stages. If the value of MBH is too small, relevant key points are not selected, resulting in poor search performance. We measure the search performance of the system with Mean Reciprocal Rank (MRR) over a number of queries. Reciprocal Rank is defined as the multiplicative inverse of the rank of the first correct answer. The MRR is the average of the reciprocal ranks of results over a number of queries. Figure 3 shows the variation of MRR and Search Time against MBH. Note that while search time monotonically increases with MBH, the MRR saturates to a maximum value close to 100 percent. An optimal value of MBH has been selected in our system based on this observation.

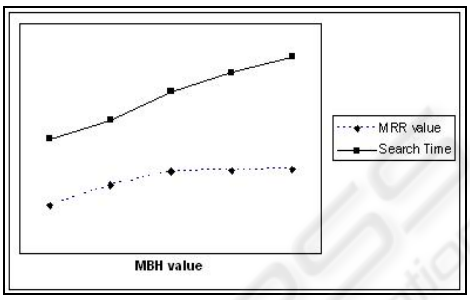

Figure 3: Variation of MRR & search time w.r.t. MBH.

Illustrative Example. We illustrate query processing through an actual query example. Figure 4 shows the sample query image.

Figure 4: Sample query image.

A search in the product image database with the PCA-SIFT keypoints of this image produces a set of candidate products. Figure 5 shows the results with the products sorted in descending order of matching keypoints (n_i). Images 1-4 have been selected from this list after the filtering stage. Note the sharp fall in number of matching keypoints between images 4 and 5.

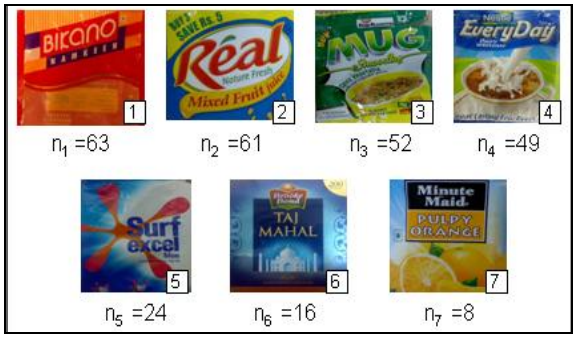

Figure 5: Intermediate result obtained.

The ranking algorithm is applied on these selected images. Figure 6 shows the ranked results

sorted in descending order of the matching scores. Note the re-ranking of the results and sharp contrast in the matching scores resulting in unique selection of product 1. Figure 7 shows the output from the system in response to this query.

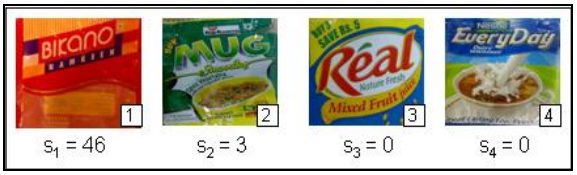

Figure 6: Images sorted w.r.t. s_i.

Figure 7: Product uniquely identified.

4 EXPERIMENTAL RESULTS

We have conducted a number of experiments to check the robustness and scalability of the system. We have implemented two shopping portal prototypes, one with grocery items and the other for music/video/games CD/DVDs as proof of concept of our approach. The first prototype is built with more than 100 grocery items, such as beverages, chocolates, dairy products, etc. The cartons/ wrappers of these products have been photographed manually to create the database. Since the packages have been physically available with us, we could create a variety of query images by photographing them in different lighting conditions as well as with different distances, angles and perspectives to study the robustness of the system. The second prototype, created with more than 1000 images of CD/DVD jackets, has been used to establish scalability of the system. We tweaked these images with image processing tools to create as realistic query images as possible. However, they were derived from the original image and truly not a different image instance of the same jackets.

Robustness. We define robustness as the capability to identify a product uniquely despite query image deformations. We use the grocery database for these experiments, since we could photograph the available wrappers in different lighting conditions, orientations and perspectives. Figure 8 shows some of the query images. Figure 9 shows some of the images present in grocery image database. These images contain product marks which have fancy font styles (Eg image 2, 5, 11), different languages (Eg image 3, 4 and 9) and uneven text size (Eg image 1 and 6).

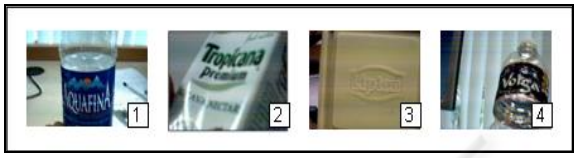

Figure 8: Sample query images containing noise and distortions.

Figure 9: Sample images from grocery image database.

We used the metrics (a) %age of times a product is uniquely identified (b) %age of time the product is identified but not be uniquely and (c) %age of times the search failed or produced wrong result to establish the robustness of the system. We used 155 different query images belonging to 31 products for this purpose. The result is summarized in Table 1.

Table 1: Experimental Results for robustness.

Total Query Images	155	
Query images in which product is identified uniquely	139	89.6%
Query images in which a product is identified, but not uniquely	11	7.0%
Query images in which no product is identified	5	3.2%
MRR		96%

We note that in about 97% of cases, the system could retrieve the results satisfactorily. In the few

cases (e.g. query image 1 in Figure 8), where multiple products were retrieved, there had been two more products in the database with very close visual characteristics (e.g. product images 1 and 7 in Figure 9). It is possible to confuse between the products on manual selection unless one scrutinizes the labels carefully. In the rare cases, where the desired product was not retrieved, the query image had too much distortion or occlusion (e.g. query image 3 in Figure 8).

Scalability. To show scalability of the system we have used an image database of 1000 plus CD/DVD jackets. In this experiment we applied some morphological operations on the database images to create the query images. We were constrained to use this option since we did not have the jackets physically with us. We studied the variation of the MRR and total search time over the number of database items and the results are shown in Figure 10 and Figure 11 respectively. We note that the MRR remains constant and the search time increases marginally on increasing the database size. The total search time of the system increased by merely 18 percent on five-fold increase of data size, from 200 to 1000. As Figure 11 shows, the rise has been attributed to the Java R-Tree implementation.

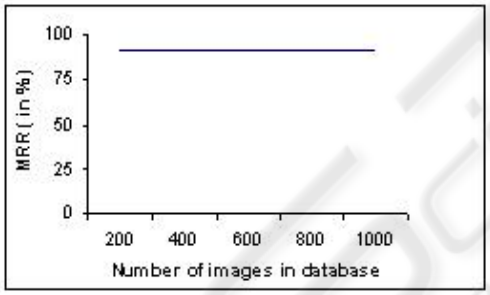

Figure 10: MRR of the system.

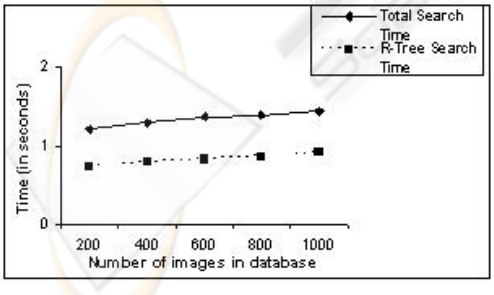

Figure 11: Time performance graph of the system.

5 CONCLUSIONS

In this paper we have presented a novel solution for Internet shopping with image examples of products. The solution can be used for a wide range of packaged products characterized by distinct visual designs or bearing distinct product-marks, such as grocery items, music, video (CD/DVDs), computer games and many of the books and magazines.

ACKNOWLEDGEMENTS

We would like to thank Yan Ke of CMU for his consent to use PCA-SIFT code given by him. In addition we would also like to thank several members of Oracle team – Steven Serra, Siva Ravada, Jack Wang, Ning An for their technical assistance with Oracle Java R-Tree.

REFERENCES

Nielsen, 2008. *Trends in Online Shopping,* A Global Nielsen consumer report, February 2008.

Ghosh, H., Chaudhury, S., 2002. WindowShopper: Guided Shopping in e-market, *In Proc. of Intl. Conf. KBCS-2002,* National Centre of Software Technology(NCST).

Tollmar, K., *et al,* 2007. A picture is worth a thousand keywords: Exploring mobile image-based web search, *In Proc. of 9th Intl. Conf. on Human-Computer Interaction with Mobile devices,* Singapore.

Ke, Y., Sukthankar, R., 2004. PCA-SIFT: A more distinctive representation for local image descriptors, *In Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR.).*

Lowe, D., G., 2004. Distinctive image features from scale invariant keypoints, *In Intl. Journal of computer vision, Vol. 60, Issue 2.*

MacQueen, J., B., 1967. Some Methods for classification and analysis of Multivariate Observations, *In Proc. of 5th Berkeley Symposium on Mathematical Statistics & Probability,* Berkeley, University of California.

Ester, M., *et al*, 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. *In 2nd Intl. Conf. on knowledge discovery and data mining.*

Guttman, A., 1984. R-Trees: A dynamic index structure for spatial searching, *In Proc. of 1984 ACM SIGMOD Intl. Conf. on Management of data.*

Berchtold S., *et al,* 1996. The X-Tree: An index structure for high dimensional data, *In Proc. of 22nd VLDB Conf,* Mumbai, India.

Oracle 11g, 2007. Oracle Spatial Java API Reference, URL http://download.oracle.com/docs/cd/B28359_01/appdev.111/b28401/overview-summary.html, last retrieved on 26th Sept'08.

Bercken, J., V., *et al, 1997.* A generic approach to bulk loading Multidimensional index structures, *In Proc. of the 23rd VLDB Conf*, Athens, Greece.