suitable threshold, images that are semantically nearer are retrieved from the database and displayed as a thumbnail. The system use case realization is shown in Figure 9 and the CCBIR procedure is given below.

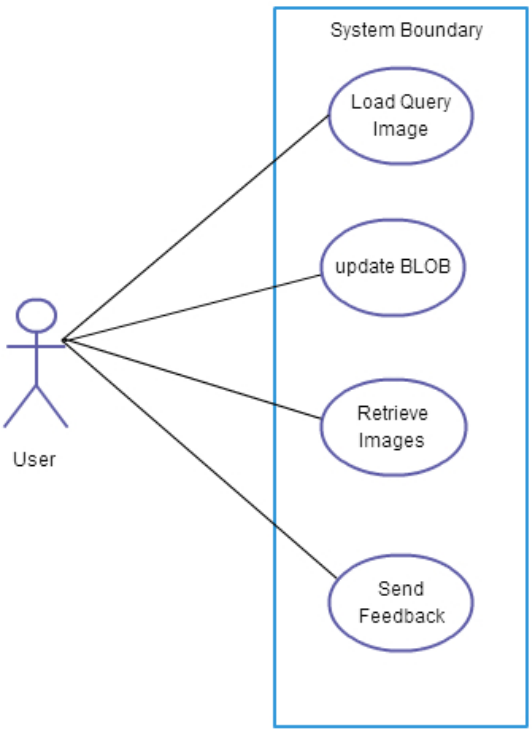


Fig. 9: System Use Case realization

Procedure for CCBIR Implementation:

1. Send the query image through the web role handling the application, for computation, to a worker node.
2. Worker node sends back the structure containing the image features of the query image.
3. Send the request for computations on the images stored in the BLOBs to worker nodes, with query image features as parameter.
4. Worker nodes send back a structure containing the image name and the similarity index for each image on which they operate (making use of the procedure to compute image features for these images).
5. Rank images based on the similarity index and create thumbnails for the images and display

Procedure to compute features of an image:

SEGMENTATION

1. Segment the image such that it becomes a 6*6 matrix.
2. Compare each segment of the query image with the images in the repository until a match is found.
3. The matched segment is flagged to avoid future computations with it.
4. This is repeated for all the segments of the image and the image with >30 segments, is tagged as one of the similar images.

The Algorithms for Segmentation and Segment comparison are shown in Table1 and Table2 respectively:

Table 1
ALGORITHM FOR SEGMENTATION

Algorithm: Segmentation

Get the Boundary of the image along Y-axis
Get the Boundary of the image along X-axis
//Segment the bitmap image into a 6*6 matrix
for every boundary of the divided image along Y-axis
for every boundary of the divided image along X-axis
Get the segment boundary
Copy segment
end for
end for
return Segmented_Images;

Table 2
ALGORITHM FOR SEGMENT COMPARISON

Algorithm: Segment Comparison

Initialize a flag array of size 36
for every segment of query image
for every segment from repository
if(the flag is set AND the segments are similar)
Increment SimilarityCount;
end if
end for
end for
if(SimilarityCount >30)
Tag image to be similar
end if

FEATURE COMPUTATION

1. For each region compute histogram on the segments and select features of those segments that are most frequently appearing inside that region (this is done by defining a threshold to indicate similarity).
2. Extract the texture features for the selected fragments in the last step.

3. Include the information in a structure that represents the particular region.
4. Combine information from all regions into a structure to represent the features for the entire image.

The Algorithm for Feature Computation is shown in Table 3:

Table 3
ALGORITHM FOR FEATURE COMPUTATION

Algorithm: Feature Computation
for all the pixels in the image Get a pointer to each pixel of the image Store the RED, GREEN, BLUE in an array **end for** Sort the RED, GREEN, BLUE array in ascending order **for** first 64 values of each array Feature= Feature+ Number of pixels/(1+Intensity of pixel); Feature= Feature/ 64 **end for** **return** Feature;

FEATURE COMPARISON

1. The features are often treated as vectors, so the difference turns to be the distance between these two vectors. It's naturally to define the distance in terms of Euclidean norms.
2. Given two images, the difference between the two features measures the similarity of these two images. The features are often treated as vectors, so the difference turns to be the distance between these two vectors. It's naturally to define the distance in terms of Euclidean norms. But the absolute distance is not quite suit for this task. For example, we have two pairs of images (a, a'), (b, b'). The feature of these images: f (a) = 1000, f (a') =1050, f (b) =100, f (b') =150. The absolute distances are same for both cases, but it's obviously that the difference in the second case is more significant.
3. So we use the relative measure of distance given in equation (2):

$$d_{(r,s)} = \frac{|r-s|}{1+r+s} \tag{2}$$

Where r and s are the features to be compared and d is the distance between them. The distance is used as the similarity index that is used for ranking the images (lesser the distance more similar are the images being compared).

The Algorithm for Feature Comparison is shown in Table 4:

Table 4
ALGORITHM FOR FEATURE COMPARISON

Algorithm : **Feature Comparison**
Get the Feature of Query image
Get the Feature of Image from repository
Calculate the difference between the above features

6 Experimental Results

Database:

The dataset consisting of the classes mentioned in Table1 and some assorted images. This is considered as our test environment on cloud. The training was on less than half the testing data. The Table5 shows the general performance of the system on image database where images are classified by humans roughly based on objects present in them. It can be observed that the performance of the system depends widely on the kind and variety of objects in the image. Totally, we have considered one thousand assorted images, randomly taken from the web. So, from the image database of one thousand images, we have calculated precision and recall for varied image queries.

The recall is defined as the proportion of relevant documents retrieved and precision is defined as the proportion of retrieved documents that is relevant. The accuracy is defined as the difference between the proportion of retrieved documents that are relevant and the proportion of retrieved documents that are irrelevant. The recall, precision and the accuracy for the database is shown in Table5.

The common evaluation measures used in CBIR systems are precision and recall defined as equation (3) & (4) respectively.

$$\text{Precision}(P) = \frac{r}{n} \tag{3}$$

$$\text{Recall}(R) = \frac{r}{R} \tag{4}$$

Where: r: number of relevant documents retrieved
n: number of documents retrieved
R: total number of relevant documents.

The average precision is calculated as the ratio of difference between the number of relevant pictures retrieved (r_i) and the number of irrelevant pictures (ir_i) to the total number of relevant pictures in the database (tr). The average is calculated over query of all the images in the class and is given in equation (5).

$$\text{Average Precision} = \frac{(r_i - ir_i)}{tr} \quad (5)$$

The accuracy is calculated by sum of precision and recall and then divided by two and is given in equation (6).

$$\text{Accuracy} = \frac{(\text{Precision} + \text{Recall})}{2} \quad (6)$$

Canberra and Euclidean distance given as equation (7) & (8), both measures were used for finding similar images. Euclidean distance is used between the standard deviations of the query image and the images in the database.

$$\text{Euclidean Distance} = \sqrt{\sum (a_i - b_i)^2} \quad (7)$$

$$\text{Canberra Distance} = \sum \frac{|a_i - b_i|}{|a_i| + |b_i|} \quad (8)$$

The performance of CCBIR system with respect to average precision, precision, recall and accuracy using Euclidean Distance (Method1) is given in Figure 10.

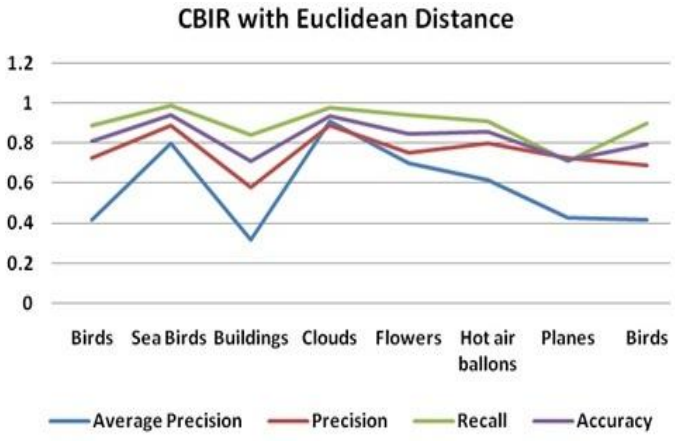


Fig. 10: Performance of the CBIR system using Euclidean Distance (Method1)

The performance of normal CBIR system using for Each Category of Images for Canberra Distance with respect to average precision, precision, recall and accuracy is given in table 5.

Table 5
PERCENTAGE OF PERFORMANCE OF THE CBIR SYSTEM USING CANBERRA DISTANCE (METHOD2)

S. No.	Category	Average Precision (%)	Precision (%)	Recall (%)	Accuracy (%)
1.	Birds	0.468	0.734	0.93	0.601
2.	Sea Birds	0.88	0.94	1	0.91
3.	Buildings	0.36	0.68	0.86	0.52
4.	Clouds	0.94	0.97	1	0.955
5.	Flowers	0.716	0.858	0.97	0.787
6.	Hot air ballons	0.63	0.815	1	0.7225
7.	Planes	0.467	0.7335	0.75	0.60025
8.	Birds	0.342	0.671	0.71	0.5065

The average performance of accuracy of CBIR with Euclidean Distance and Canberra Distance is given in Figure 11.

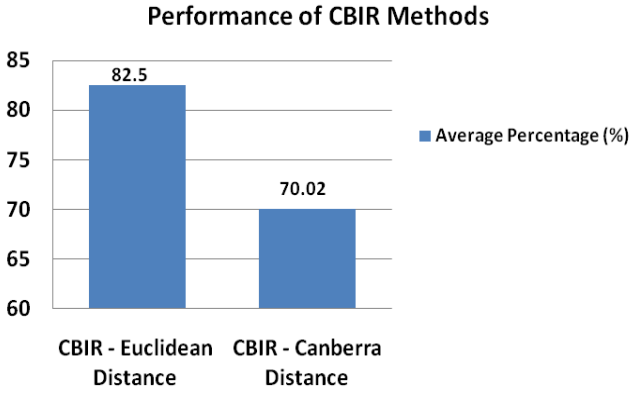


Fig. 11 Average Percentage of Accuracy of all CBIR methods

Performance:

Since the proposed approach is implemented on the cloud platform there is a considerable increase in the speed of computation. Azure provides a set of Worker roles that can perform parallel computations thus decreasing the overall processing speed. And also the following non-functional features are provided.

Fault tolerance

To an application developer, Windows Azure consists of the Compute service and the Storage service. However neither one might function exclusive of the Windows Azure Fabric. By means of disapproval simultaneously a data center filled of machines into an

articulate whole, the Fabric provides a basis for the whole thing else.

The fabric controller owns all resources in a particular Windows Azure data center. It's also conscientious for handing over instances of both applications and storage to physical machines. Doing this intelligently is important. A developer requests for a number of Web role instances and Worker role instances for his/her application. A naive assignment might situate every one of these instances on machines in the same rack serviced by the identical network switch. If either the rack or the switch failed, the entire application would no longer be available. Known the high accessibility goals of Windows Azure, building an application dependent on single points of failure like these would not end up achieving high accessibility. To avoid this, the fabric controller groups the machines it owns into an amount of liability domains. Every fault domain is a part of the data center where a single failure can shut down access to everything in that domain.

The application portrayed in the figure above is running just two Web role instances, and the data center is divided into two fault domains. While the fabric controller deploys this application, it places one Web role instance in each of the fault domains. This arrangement means that a single hardware failure in the data center can't take down the entire application. The fabric controller sees Windows Azure Storage as just another application - the controller doesn't handle data replication. Instead, the Storage application does this itself, making sure those replicas of any blobs, tables, and queues used by this application are placed in different fault domains.

Scalability

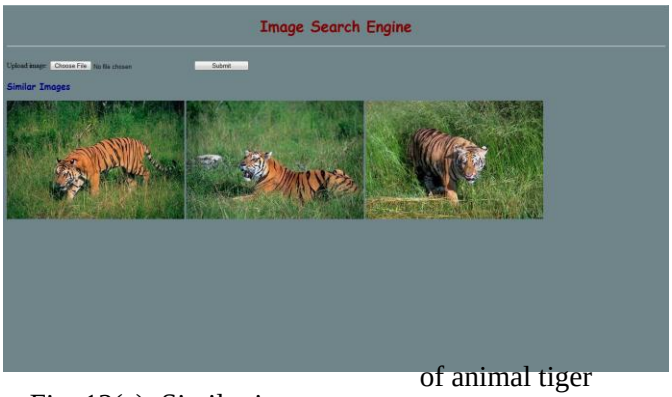
Scalability can be achieved in two ways: Computing scalability and Data storage scalability. Computing scalability is achieved by dynamically scaling out the number of the Worker role instances as and when the requirement arises. Data storage scalability is achieved by the usage of BLOBs for storing the images which are scalable by nature.

Security

To use Windows Azure Storage a user must build a storage account. This can be done via the Windows Azure portal internet interface. The user can receive a 256-bit secret key once the account is created. Further the above secret key is used to substantiate user requests to the storage system. Specifically, a HMAC SHA256 signature for the request is produced by using the above secret key. The signature is accepted among every client requirements by request to authenticate the verifying the HMAC signature.

Snapshots of CCBIR System:

Snapshot is that the state of a system at a selected purpose in time. Snapshot can refer to a genuine copy of the state of a system or to a potential provided by certain systems. These snapshots output a set of similar images to a query image specified. The following Figures 12(a) to 12(h) shows various results of different image cases of CCBIR system.



of animal tiger

Fig. 12(a): Similar image output

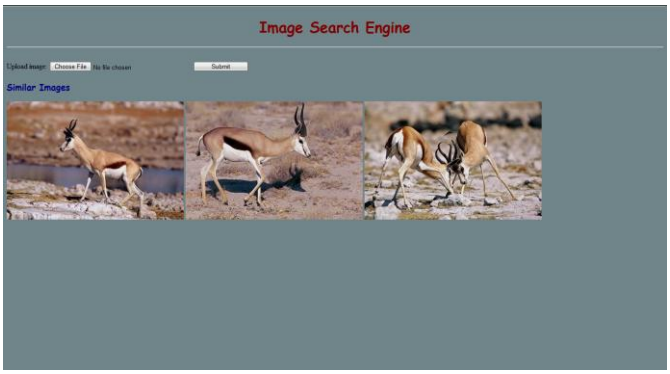


Fig. 12(b): Similar image output of animal deer

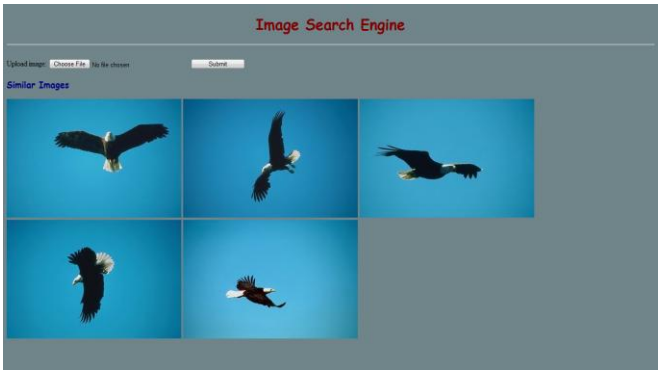


Fig. 12(c): Similar image output of birds

Fig. 12(d): Similar image output of flowers

Fig. 12(e): Similar image output of planes

Fig. 12(f): Similar image output of clouds

Fig. 12(g): Similar image output of combination of images

Fig. 12(h): Similar image output of combination of images

7 Conclusion

The need to discover a desired image from a group is shared by several professional teams, including design engineers, journalists and art historians. Whereas the necessities of image users will vary significantly, these are often useful to characterize image queries into three levels of abstraction: primitive features such as shape or color, logical features such as the individuality of objects given away and abstract attributes such as the importance of the scenes depicted. Whereas CBIR systems presently function efficiently simply at the lowest of these levels, the majority users require higher levels of retrieval.

Users need to retrieve images from a set come from a diversity of domains, including medicine, architecture, crime prevention, publishing and fashion. Extremely minute has yet been available on the approach such users seek for and use images, though makes attempts are being created to classify users' behavior within the hope that this can permit their requirements to be better met within the future.

Existing indexing carry out for images depends mostly on classification codes or text descriptors supported in several cases by text retrieval correspondence intended or made to order especially to handle images. Yet again, extremely tiny substantiation on the efficiency of such systems has been published. Client fulfillment with such systems appears to differ significantly.

CBIR operates on a completely different principle from keyword indexing. Primitive features characterize image content, such as color, textures, and shape, are computed for both stored and query images, and used to identify the twenty stored images most strongly corresponding the query. Semantic features comparable to the kind of object present within the image more durable to extract, although this remains an active research topic. Video retrieval may be a topic of increasing importance. CBIR techniques are moreover used to fragment extensive videos into entity shots, extract still key frames shortening the content of every shot, and explore video clips containing particular types of movement.

The effectiveness of all current CBIR systems is intrinsically restricted by the fact that they can operate only at the primitive feature level. None of them will search effectively for, say, a photograph of a dog, although some semantic queries may be handled by specifying them in terms of primitives. A beach scene, for instance, is retrieved by specifying huge areas of blue at the top of the image, and yellow at the bottom. There is substantiation that combining primitive image features with hyperlinks or text keywords will conquer variety of these troubles, however modest is known with relation to however such features will best be combined for retrieval.

References:

[1] Remco C. Veltkamp, Mirela Tanase(2002). "Content-Based Image Retrieval Systems: A Survey", *UU-CS*-2000-34, 2002.

[2] "Windows Azure Platform Home Page", *http://www.microsoft.com/windowsazure*

[3] David Chappell. "Introducing the Windows Azure Platform". *http://go.microsoft.com/fwlink/?LinkId=158011*.

[4] W. Y. Ma (1997). "NETRA: A Toolbox for Navigating Large Image Databases". *Ph.D thesis, Dept. of Electrical and Computer Engineering, University of California* at Santa Barbara, June 1997.

[5] Wei-Ying Ma and B. S. Manjunath (1999). "Netra: A toolbox for navigating large image databases". *Multimedia Systems*, 7(3):184–198, 1999.

[6] J. Fournier, M. Cord, and S. Philipp-Foliguet (2001). "Retin: A content-based image indexing and retrieval system". *Pattern Analysis and Applications*, 4(2/3):153–173, 2001.

[7] E. Loupias and S. Bres (2001). "Key point-based indexing for pre-attentive similarities: The kiwi system". *Pattern Analysis and Applications*, 4(2/3):200-214, 2001.

[8] Etienne Loupias and Nicu Sebe (2000). "Wavelet-based salient points: Applications to image retrieval using color and texture features. In Advances in Visual Information Systems" . *Proceedings of the 4th International Conference, VISUAL 2000, Lecture Notes in Computer Science* 1929, pages 223–232. Springer, 2000.

[9] Gaurav Aggarwal, Pradeep Dubey, Sugata Ghosal, Ashutosh Kulshreshtha, and Abhinanda Sarkar (2000). "iPure: Perceptual and user-friendly retrieval of images". *In Proceedings of IEEE Conference on Multimedia and Exposition (ICME 2000)*, July 2000.

[10] J. Kreyss, M. R¨oper, P. Alshuth, Th. Hermes, and O. Herzog (19997). "Video retrieval by still image analysis with ImageMiner". *In Proceedings of IS&T/SPIE's Symposium on Electronic Imaging: Science & Technologie,* 8-14 Feb. '97, San Jose, CA, 1997.

[11] Sameer Antani, L. Rodney Long, George R. Thoma (2004). "Content-Based Image Retrieval for Large Biomedical Image Archives" *MEDINFO 2004*.

[12] S. Nandagopalan, Dr. B. S. Adiga, and N. Deepak (2008). "A Universal Model for Content-Based Image Retrieval". *World Academy of Science, Engineering and Technology*, Vol.2, 2008.

[13] Michael Eziashi Osadebey (2006). "Integrated Content Based Image Retrieval using texture, shape, spatial information", *Master Thesis Report in Media Signal Processing*, Department of Applied Physics and Electronics, Umea University, Umea, Sweden, February 2006.

[14] Chang, S K et al (1988). "An intelligent image database system". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(5), 681-688.

[15] Eakins J. P, Boardman J. M and Graham M. E (1998). "Similarity retrieval of trade mark images". *IEEE Multimedia*, 5(2), 53-63.

[16] N.Vasconcelos and A. Lippman (2000). "Feature Representations for Image Retrieval: Beyond the Color Histogram". *Proceedings of the International Conference on Multimedia and Expo*, New York, 2000.

[17] John P. Eakins and Margaret E. Graham (1999). "Content-based Image Retrieval". *A Report to the JISC Technology Applications Program*, University of Northumbria at Newcastle.

[18] Eakins J P, Graham M E and Boardman J M (1997). "Evaluation of a trademark retrieval system", *in 19th BCS IRSG Research Colloquium on Information Retrieval*, Robert Gordon University, Aberdeen.

[19] Shengjiu Wang (2001). "A Robust CBIR Approach Using Local Color Histograms*",TR 01-13*, Canada, October 2001.

[20] Lu, G. J. Phillips and S. Rahman (1998). "Techinques to Improve Color Based Image Retrieval Performance*". Proceedings of International Symposium on Audio, Video, Image Processing and Intelligent Applications*, Baden Baden, Germany. pp 57-61. August 17-21, 1998.

[21] Nuno Vasconcelos (2007). "From Pixels to Semantic Spaces: Advances in Content-Based Image Retrieval", *IEEE Computer*, 2007.

[22] Karande S.J, Maral V (2013). "Semantic content based image retrieval technique using cloud computing", *Computational Intelligence and Computing Research (ICCIC), 2013 IEEE International Conference on*, 26-28 Dec. 2013, Enathi.

[23] Belloulata K, Belallouche L, Belalia A, Kpalma K (2014). "Region based image retrieval using Shape-Adaptive DCT", *Signal and Information Processing (ChinaSIP), 2014 IEEE China Summit*

& International Conference on, 9-13 July 2014, Xi'an.

[24] Choudhary R, Raina N, Chaudhary N, Chauhan R, Goudar R.H (2014). "An integrated approach to Content Based Image Retrieval", *Advances in Computing, Communications and Informatics (ICACCI, 2014 International Conference on*, 24-27 Sept. 2014, New Delhi.

[25] Yang Di, Liao, Jianxin, Wang, Jingyu Qi, Qi Sun, Haifeng (2014). "Multiple features for image retrieval in distributed datacenter", *Network Operations and Management Symposium (APNOMS), 2014 16th Asia-Pacific*, 17-19 Sept. 2014, Hsinchu, Taiwan.

[26] Di Yang, Jianxin Liao, Qi Qi, Jingyu Wang, Tonghong Li (2014). "An image retrieval framework for distributed datacenters", *Local Computer Networks (LCN), 2014 IEEE 39th Conference on*, 8-11 Sept. 2014, Edmonton, AB.