Image Search Engine with Different Image Similarity Semantics

Wiwit Rifa'i
School of Electrical Engineering and Informatics
Institut Teknologi Bandung
Bandung, Indonesia
wiwitrifai@gmail.com

Dr. Ir. Rinaldi Munir, M.T.
School of Electrical Engineering and Informatics
Institut Teknologi Bandung
Bandung, Indonesia
rinaldi.munir@itb.ac.id

***Abstract*— The increasing number of image collections makes image search engines increasingly needed to efficiently search for images in large image collections. Image collections that have descriptive text can utilize the descriptive text to be used in text-based search engines, but for text-free image collections it is necessary to use a content-based image search engine (CBIR) that is only based on visual information on the image. In CBIR, image similarity is based on features that can be extracted from images such as color, texture and shape. One of the main problems in CBIR is the semantic gap, the gap that occurs because of the limitations of extraction features in describing the expected semantics. Semantics that are used to compare the similarities of two images are very dependent on user's perspective that tend to be subjective. Therefore, the similarity semantic of the image can vary greatly depending on the judgment and intent of the user. One approach in overcoming semantic gap is by weighting the image extraction feature. The weighting of these features determines the features that are considered more dominant in comparing image based on the intended image similarity semantic. Relevance feedback can be used to calculate the weighting of features based on feedback from users in order to get closer to the intended image similarity semantic. The relevance feedback method used in weighting calculations is Self Order Feature Reweighting. The Inverted Multi-Index is also used as an index data structure so that the image search process becomes more efficient. The test results show that the application of relevance feedback can provide increased accuracy but not too significant. The Inverted Multi-Index data structure is able to increase the search speed for a small number of images taken. The weakness of the Inverted Multi-Index has the effect of decreasing accuracy and does not work well if the number of images taken is too much.**

Keywords—image, similarity semantic, relevance feedback, inverted multi-index

I. INTRODUCTION

The ease in making, storing and disseminating digital images makes the number of images increase over time. The amount of digital image that is too much can make it difficult for people to find the desired image. One solution to solve these problems is with the image search engine. With a digital image search engine, users only need to enter several queries stating the criteria of the desired image. Then the query will be processed by a digital image search engine and will output the results of several images that correspond to the query entered by the user.

Query used in digital image search can be text or image. The advantage of digital images rather than text is that images sometimes have more information that cannot be expressed in a text so that queries in the form of digital images can provide more information than text. In addition, the shortcomings of queries in text form are that not all image collections have descriptive text that can be used as a comparison with the query text. Descriptive text is generally obtained when the image retrieval process from an article or web page. So that image collections that do not have descriptive text are only able to use a Content Based Image Retrieval (CBIR).

Digital images can store a variety of visual information that is quite complex such as the composition of colors, object shapes, textures and so on. To process queries in the form of digital images, low-level feature information is extracted from these images which will then be used as information to describe query criteria. But such low-level information is sometimes too limited or difficult to describe semantics at the conceptual level. [1] refer to these problems as "semantic gap" and are the most challenging problem in CBIR.

When someone uses digital image as a query, semantic gap problems also arise in interpreting the image criteria that they want to search based on the given query image. In general, the criteria of the image you want to look for when using an image as a query is how similar the image is to the query image. According to the user's point of view, the assessment of similarity between images is subjective and depends on the semantic similarity used. Similarity semantic is semantics or the meaning of the user's assessment of how similar the images are compared. For example for a collection of images with a mixture of various landscape images, building images, batik pattern images, etc.,, when someone uses a landscape image as a query the similarity semantics that tend to be used are based on the scene's atmosphere where the color composition feature in the image is more important than other visual features. Whereas when someone uses the image of batik as a query, the similarity semantic that tends to be used is based on the characteristic of the batik pattern of the image where the texture feature becomes the most dominant. For the same query image sometimes several different semantics are needed, for example when someone uses an image of batik-patterned clothing there is a possibility that the person wants to look for an image with the same pattern or maybe based on the shape

model. So that the definition of image similarity can vary greatly depending on the purpose of the user.

II. Related Works

A. Relevance Feedback

Relevance feedback is a method used to improve the results of an information retrieval based on user feedback on the results provided. Users will provide feedback whether every information displayed is relevant or not. Relevance feedback will change a parameter in information retrieval based on the feedback provided.

[2] is one example of relevance feedback that is used to change the weighting of features in image retrieval. They named this method Self Order Feature Reweighting (SOFRW). Feedback provided is whether each image displayed is relevant to the query or not. Then the weighting feature will be recalculated every time the feedback is given. Feature weighting is calculated based on formula (1).

$$W_i^{k+1} = \frac{\left(\epsilon + \sigma^k_{N_{r,i}}\right)}{\left(\epsilon + \sigma^k_{N_{rel,i}}\right)} \quad (1)$$

In formula (1), weighting is calculated based on the standard deviation of each feature component. $\sigma^k_{N_{r,i}}$ is the standard deviation for the N_r image retrieved. $\sigma^k_{N_{rel,i}}$ is the standard deviation of relevant images in the k iteration. ϵ is a very small decimal to prevent division by 0.

$$overall\ distance = \sum_{i=1}^{n} w_i f_i(query, DBImg) \quad (2)$$

From the weight that has been calculated in the formula (1), the calculation of the overall distance value of query image and *DBImg* image is as in formula (2). The *f* function in formula (2) is the distance function used to calculate the distance between two features on the same component. Usually the *f* function used is euclidean distance.

B. Index Data Structure

The index data structure is a data structure to store information about image collections in the form of image extraction features and manage these features so that image search can be performed efficiently. But storage in memory may be larger because it is used to store structural information to organize stored data. There are several types of index data structures such as Partition Tree [3], Neighbors Graph [3], Compact Codes [3], Inverted Index [4], and Inverted Multi-Index [4].

III. Proposed Method

A. Image Features

The image extraction feature is used to be a basic characteristic that distinguishes one image from another. There are several types of features such as features based on color, texture and shape. In CBIR, one or more extraction features are used to describe the image. The more features used, the better the ability to compare between images, but too many features will make long time performance.

In order for the semantics that can be achieved based on the image extraction feature to be greater, the selected image extraction features must have a different semantic tendency. With these considerations, one method of extraction features will be selected for each category (color, texture and shape). So that the extraction feature that will be used is HSV with color moments as a color feature, the gabor filter as a texture feature, and the GIST as a shape feature.

B. Index Data Structure

Changes in the index data structure must be carried out efficiently. In the search engine that is built, changes in the data structure occur when adding new image similarity semantics or making improvements to a image similarity semantic with relevance feedback. Partition tree and neighborhood graph structures are not suitable for storing more than one image similarity semantic because changes from relevance feedback to the structure data are too large. Standard Inverted Index will produce very different clustering for different similarity semantics. But the standard Inverted Index can be combined with a compact code so that clustering is only done on each feature that tends to be different. So that weighting can be done on each feature. The structure data is called the Inverted Multi-Index [4].

C. Relevance Feedback

The approach used in representing image similarity semantic is weighting on features. Therefore the relevance feedback used must be able to calculate the weighting based on the feedback provided. The chosen relevance feedback is Self Order Feature Reweighting (SOFRW). SOFRW was chosen because [2] claimed that this weighting method provided better results than previous similar researches.

D. Architecture

The process in search engines in general can be divided into 2 separate processes i.e. indexing and search. The process flow in the search engine built is as in Figure 1 and Figure 2. Based on the process flow, the architecture of the search engine is as shown in Figure 3 and Figure 4.

The architecture is divided into 5 modules: interface module, extractor module, indexer module, searcher module, and improver module. Interface module are used to handle interactions with users. Extractor module is responsible for extracting image features. Indexer module is a module that is tasked with storing information on a collection of images along with image features that have been extracted with the Extractor module into an index data structure. Searcher module is a module that is tasked with finding a number of images in an index that are considered to be most similar to query images based on a specific feature weighting. Improver module is a module that is responsible for improving search results and processing relevance feedback.

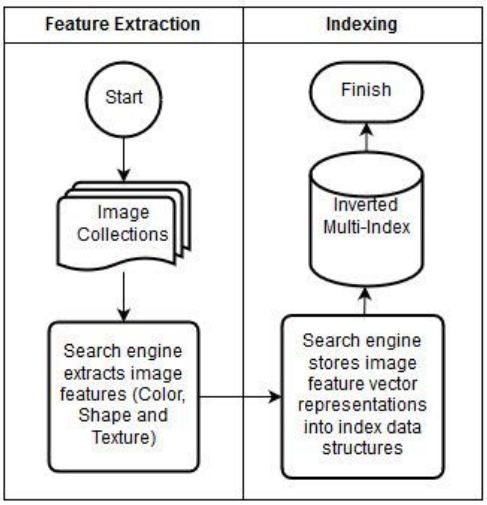

Fig. 1. Indexing Flowchart

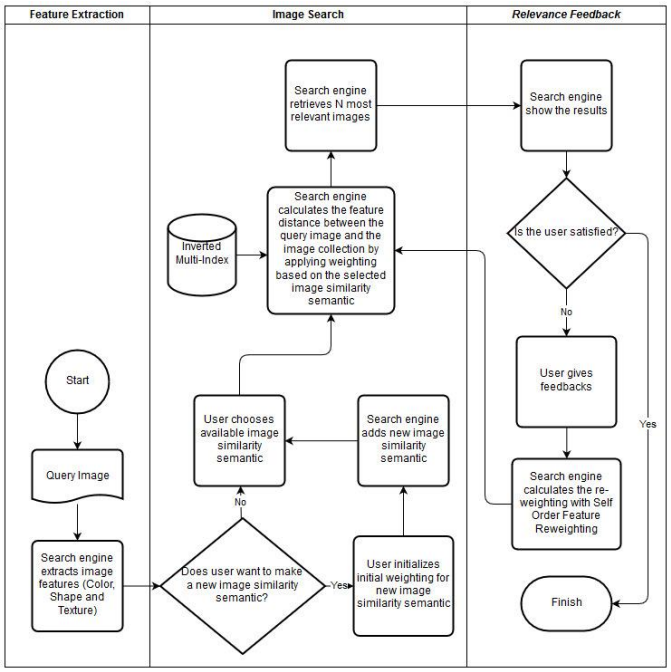

Fig. 2. Searching Flowchart

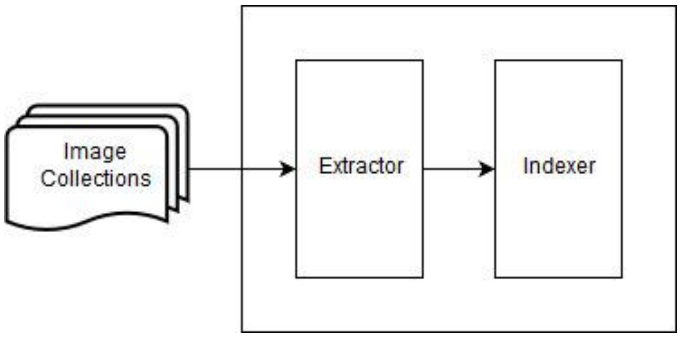

Fig. 3. Indexing Architecture

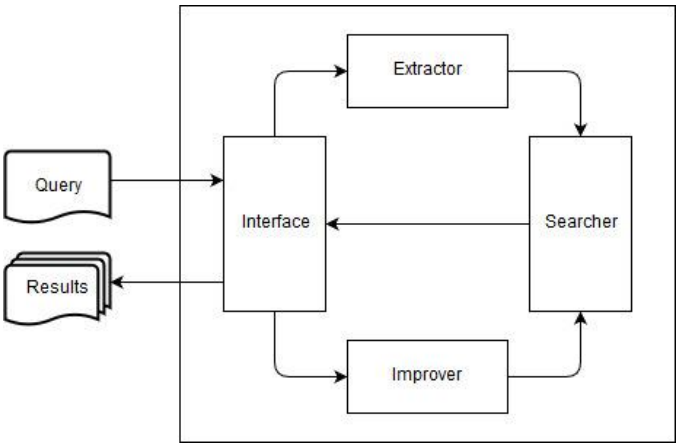

Fig. 4. Searching Architecture

IV. Implementation and Experiments

Implementation and testing are carried out on the same computer that has specifications as in Table I. The tests are carried out with Holiday [5] and Mirflikr [6] datasets. Image relevance is chosen based on the relevance that has been given or the same tag in the dataset. Some queries are randomly selected.

TABLE I. Implementation & Testing Environment

Aspect	Specification
Processor	Intel® Core™ i7-4720HQ CPU @ 2.60GHz × 8
RAM	11.6 GiB
Operating System	Fedora 28 64-bit
Programming Language	C++17 and Python 3.6.5
Image Processing Library	Open CV 3.4.1
Web Framework	Flask 1.0.2

A. Implementation

Extractor, indexer, searcher, and improver modules are implemented with the C++17. The extractor module uses OpenCV library to perform basic operations in image processing. These modules are built into a shared object library. The shared object library will be loaded by the interface module so that its functions can be called directly. The interface module is implemented in Python using Framework Flask as a web service.

B. Accuracy Testing

In Table II it can be seen that the resulting accuracy is small. This happens because of the incompatibility of the image extraction feature implemented in the search engine against several queries and semantic similarities used in testing. If you look at the dataset used, the query and the basis of similarity tend to depend on the shape features of objects in the image (local features) and ignore the remaining objects while the shape features used in search engines are features of the shape of the overall image (global features). So that the extraction feature used is less able to describe the query given.

TABLE II. MAP SCORES OF IMAGE SEARCH ENGINE

Iteration	*Mean Average Precision* (MAP)			
	Inverted Multi-Index		*Exhaustive Search*	
	50 Feedbacks	*500 Feedbacks*	*50 Feedbacks*	*500 Feedbacks*
0	0,002234	0,002234	0,086127	0,086127
1	0,007545	0,009398	0,096627	0,116024
2	0,008055	0,010056	0,175251	0,173096
3	0,008458	0,010134	0,175628	0,172274
4	0,008416	0,010189	0,175982	0,172185

Although there is a query mismatch with the extraction feature used, the experiment shows an increase in accuracy by relevance feedback. From Table II, compare the MAP values for each iteration with the previous iteration. MAP tends to increase, especially from the 0 iteration to the 1st iteration even up to 4.2 times. But there is also a decrease in accuracy which is not too significant to 0.995 times, this can be due to the noise that causes weighting to be less accurate. The decrease can also be due to the distribution of features from the given feedback increasingly mixing the relevant and irrelevant feedback so that it becomes increasingly difficult to separate it by weighting.

Then by comparing the experiments using 50 feedback per iteration with experiments using 500 feedback per iteration it was found that accuracy tends to be better for the number of feedback per iteration of 500. But in the experiment without Inverted Multi-Index in the 2nd, 3rd, and 4th iterations shows the opposite result. The cause of this can also be due to noise or increasingly mixed features distribution between relevant feedback and irrelevant ones.

The use of the Inverted Multi-Index has the effect of reducing accuracy. If you look at Table II and compare each MAP value between those who use the Inverted Multi-Index and those who only use Exhaustive Search, then the average is calculated, the decrease in accuracy is around 5.3% of the initial accuracy. This decrease occurs because image collection in search using Inverted Multi-Index is done in clusters and image comparison is done to the center of the cluster first to get the desired number of images. Therefore, the accuracy of Inverted Multi-Index depends on the clustering process performed. Accuracy in the Inverted Multi-Index also depends on the division of dimensions of the extraction feature because the dimensions need to be divided into sections first before clustering is carried out. After that, the smaller parts are clustered. Therefore the Inverted Multi-Index also depends on the dimension division of the extraction feature used. The division of dimensions will give better results if the interrelationship between parts of the small dimension becomes smaller.

C. Time Performance Testing

Based on Table III, the search time in the Inverted Multi-Index is longer than the exhaustive search when the comparison of the number of images taken is too much. But if the comparison of the number of images taken with the number of images in the dataset is very small, then the search in the Inverted Multi-Index will also run quickly. This happens because the search for too many images, the Inverted Multi-Index checks almost all clusters while the total number of clusters is more than the number of images in the dataset.

When the number of images taken is small, many clusters need to be checked to be very small. The advantage of Inverted Multi-Index is to find the next cluster that is closest to the query can be done quickly. Therefore, if the number of images taken is small, the search can be done quickly.

TABLE III. SEARCH PERFORMANCE TIME

Image Collections	Retrieved Images	Searching Time (s)	
		Inverted Multi-Index	*Exhaustive Search*
10000	10	0,001721	0,047213
10000	100	0,466761	0,048655
10000	1000	6,340240	0,050830
100000	10	0,001395	0,456822
100000	100	0,003245	0,488266
100000	1000	0,904204	0,507607
100000	10000	14,642767	0,531176
1001491	10	0,001294	4,560423
1001491	100	0,002037	4,845413
1001491	1000	0,008494	5,085050
1001491	10000	0,531673	5,442233
1001491	100000	7,103843	5,576297

In Table IV, the relevance feedback process runs quickly because it only takes 0.024257 seconds with a total feedback of 1024. Based on the graph, it appears that the time needed in relevance feedback is proportional to the amount of feedback provided. This happens because the calculation of relevance feedback is light so it is suitable for use in image search engines to be real-time.

TABLE IV. RELEVANCE FEEDBACK PERFORMANCE TIME

Feedback Count	Time (s)
2	0,000076
4	0,000126
16	0,000452
64	0,001497
256	0,006659
1024	0,024257
4096	0,111431
16384	0,400705
65536	1,61677

V. CONCLUSIONS

Image search engine was successfully built using relevance feedback to weight the image features based on user feedback and use the Inverted Multi-Index as an index data structure. The image extraction feature used is the color moment with the HSV color system, gabor texture, and GIST. The image search engine built is able to accept images as queries and can choose semantic similarities that are expressed as feature weighting.

The re-weighting calculation method used for relevance feedback in the search engine built is Self Order Feature Reweighting. The relevance feedback is successfully used to approach the semantic similarity that the user wants because accuracy tends to increase with relevance feedback to calculate feature weighting.

Accuracy in the test shows a small value because there is an incompatibility of extraction features used to query and semantic similarities used in testing. The mismatch occurs because the query used depends heavily on the shape features of the objects in the image (local features) while the form features used are the shape features of the overall image (global features). But the tests carried out successfully showed an increase in accuracy by relevance feedback. The increase depends on the number of iterations and feedback provided. Sometimes the addition of feedback can also reduce the insignificant accuracy that can be caused by noise in the feedback provided or the more mixed features distribution between relevant feedback that is not relevant.

Calculation of weighting in relevance feedback also requires a fast time of 0.024257 seconds for 1024 pieces of feedback and is linearly proportional to the large number of feedback. The Inverted Multi-Index works better than exhaustive search if the number of images taken is much smaller than the number of images in the dataset. Conversely, the Inverted Multi-Index will run poorly for the number of images taken too large. The Inverted Multi-Index was also successfully integrated with the weighting of Relevance Feedback, but the use of the Inverted Multi-Index has the effect of reducing accuracy to 5.3% of accuracy using exhaustive search. Therefore, the Inverted Multi-Index is only better used if the number of images is so large that the exhaustive search is considered too long.

VI. FUTURE WORK

Research on this image search engine must still be developed and further researched. The selection of datasets and queries in the test should pay more attention to their compatibility with the extraction features used. Accuracy may be enhanced by using a different method of relevance feedback or by giving different feedback, for example, there is a relevance weight for each image. Image similarity semantic can also be approached by other methods such as nonlinear transformations in order to better describe complex image similarity semantic. Research on the use of a better index data structure but does not reduce accuracy is also needed in the future.

REFERENCES

[1] Gevers, Th, and A. W. M. Smeulders. "Image search engines: An overview." *Emerging Topics in Computer Vision* (2004): 1-54.

[2] Kumar, K. Kranthi, and T. Venu Gopal. "A novel approach to self order feature reweighting in CBIR to reduce semantic gap using Relevance Feedback." In *Circuit, Power and Computing Technologies (ICCPCT), 2014 International Conference on*, pp. 1437-1442. IEEE, 2014.

[3] Wang, Jing, Jingdong Wang, Gang Zeng, Rui Gan, Shipeng Li, and Baining Guo. "Fast neighborhood graph search using cartesian concatenation." In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2128-2135. 2013.

[4] Babenko, Artem, and Victor Lempitsky. "The inverted multi-index." In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pp. 3069-3076. IEEE, 2012.

[5] Jegou, Herve, Matthijs Douze, and Cordelia Schmid. "Hamming embedding and weak geometric consistency for large scale image search." In *European conference on computer vision*, pp. 304-317. Springer, Berlin, Heidelberg, 2008.

[6] Huiskes, Mark J., Bart Thomee, and Michael S. Lew. "New trends and ideas in visual concept detection: the MIR flickr retrieval evaluation initiative." In *Proceedings of the international conference on Multimedia information retrieval*, pp. 527-536. ACM, 2010.