A Shape Based Image Search Technique

Aratrika Sarkar[1]
[1]Department of Computer Science and Engineering
Institute of engineering and Management,
West Bengal University of Technology,
Kolkata,India.
.

PallabiBhatttacharjee[1]
[1]Department of Computer Science and Engineering
Institute of engineering and Management,
West Bengal University of Technology,
Kolkata,India

***Abstract*—This paper describes an interactive application we have developed based on shaped-based image retrieval technique. The key concepts described in the project are, i)matching of images based on contour matching; ii)matching of images based on edge matching; iii)matching of images based on pixel matching of colours. Further, the application facilitates the matching of images invariant of transformations like i) translation ; ii) rotation; iii) scaling. The key factor of the system is, the system shows the percentage unmatched of the image uploaded with respect to the images already existing in the database graphically, whereas, the integrity of the system lies on the unique matching techniques used for optimum result. This increases the accuracy of the system. For example, when a user uploads an image say, an image of a mango leaf, then the application shows all mango leaves present in the database as well other leaves matching the colour and shape of the mango leaf uploaded.**

***Keywords*—shape-based image retrieval; contour matching; edge matching; pixel matching**

I. Motivation

Humans can often recognize objects using shape information alone. This has proven to be a challenging task for computer vision systems. One of the main difficulties is, developing representations that can effectively capture important shape variations. We want to compare different objects such as leaves of a tree and to detect two different leaves. The computational complexity of these tasks and the recognition accuracy obtained are highly dependent on the choice of a shape representation and comparison among leaves of various shapes.

II. Introduction

In this 21st century where “searching” on the internet in a part and parcel of life, text and voice search are dominant whereas image search is still lagging behind. So we have tried to develop an application which shows optimum results on Image Searching.

Recognition [1] relies upon the existence of a set of predefined objectsContent-based image retrieval[2] is prevalent since 1992 for automatic retrieval of images from a database, based on the colors and shapes present. Since then, the term has been used to describe the process of retrieving desired images from a large collection on the basis of syntactical image features[3]. The techniques, tools, and algorithms that are used originate from fields like statistics, pattern recognition, signal processing, and computer vision[4].

The problem of recognition of objects represented in images is the problem of identifying homologous elements in shapes, which are usually defined by groups of points.Our approach focuses on finding the optimum matching of the images taking contour[5] as the key feature of the image. Contour matching[6] is an important issue and a difficult problem of image processing. The accuracy and the efficiency of the algorithms are the most two critical factors.Contour representation defines the boundary of an object. We must keep in mind that the object must be identified even if it undergoes some geometric transformations. We aim to find the output as images which match the input image in terms of maximum percentage matching. The user has options to find out results in terms of EDGE MATCHING[7],CONTOUR MATCHING,COLOUR MATCHING[8].In section III we discuss the related work, followed by the Methodology presented in section IV. Section V presents some snapshots while Section VI shows the experimental results and finally in Section VII we conclude the discussions.

III. Related Work

Many methodologies have been proposed to analyze plant leaves in an automated fashion. A large percentage of such works utilize shape recognition techniques to model and represent the contour shapes of leaves, however additionally, color and texture of leaves have also been taken into consideration to improve recognition accuracies. One of the earliest works employs geometrical parameters like area, perimeter, maximum length, maximum width, elongation to differentiate between four types of rice grains, with accuracies around 95% [9]. Use of statistical discriminant analysis along with color based clustering and neural networks have been used for classification of a flowered plant and a cactus plant. The authors use the Curvature Scale Space (CSS) technique [10] and k-NN classifiers[11] to classify chrysanthemum leaves. Both color and geometrical features have been reported to detect weeds in crop fields employing k-NN classifiers. The authors propose a hierarchical technique of representing leaf shapes first by their polygonal approximations and then introducing more and more local details in subsequent steps. Fuzzy logic [12] decision making has been utilized to detect weeds in an agricultural field. The authors propose a two-step approach of using a shape characterization function called centroid-contour distance curve[13] and the object eccentricity[14] for leaf image retrieval. The centroid-contour distance (CCD) curve and eccentricity along with an angle code histogram (ACH) [15] have been used for plant recognition.

IV. METHODOLOGY

Fig 1 shows the flowchart for the algorithm developed. There are several functions which the application can perform. It allows the user to input an image of his/her choice. Then the user has the facility of matching edge, contour which will take the image as input and perform several functions. For instance "Find edges" is a function, which will take an image as input, and find out the matching images as output whose edges match the edge of the input image. Contour representation defines the boundary of an object. The object has to be identified even if it undergoes some geometric transformation and our application succeds in finding out the matching of the input image even if it undergoes geometric transformations [16] like rotation, scaling etc.

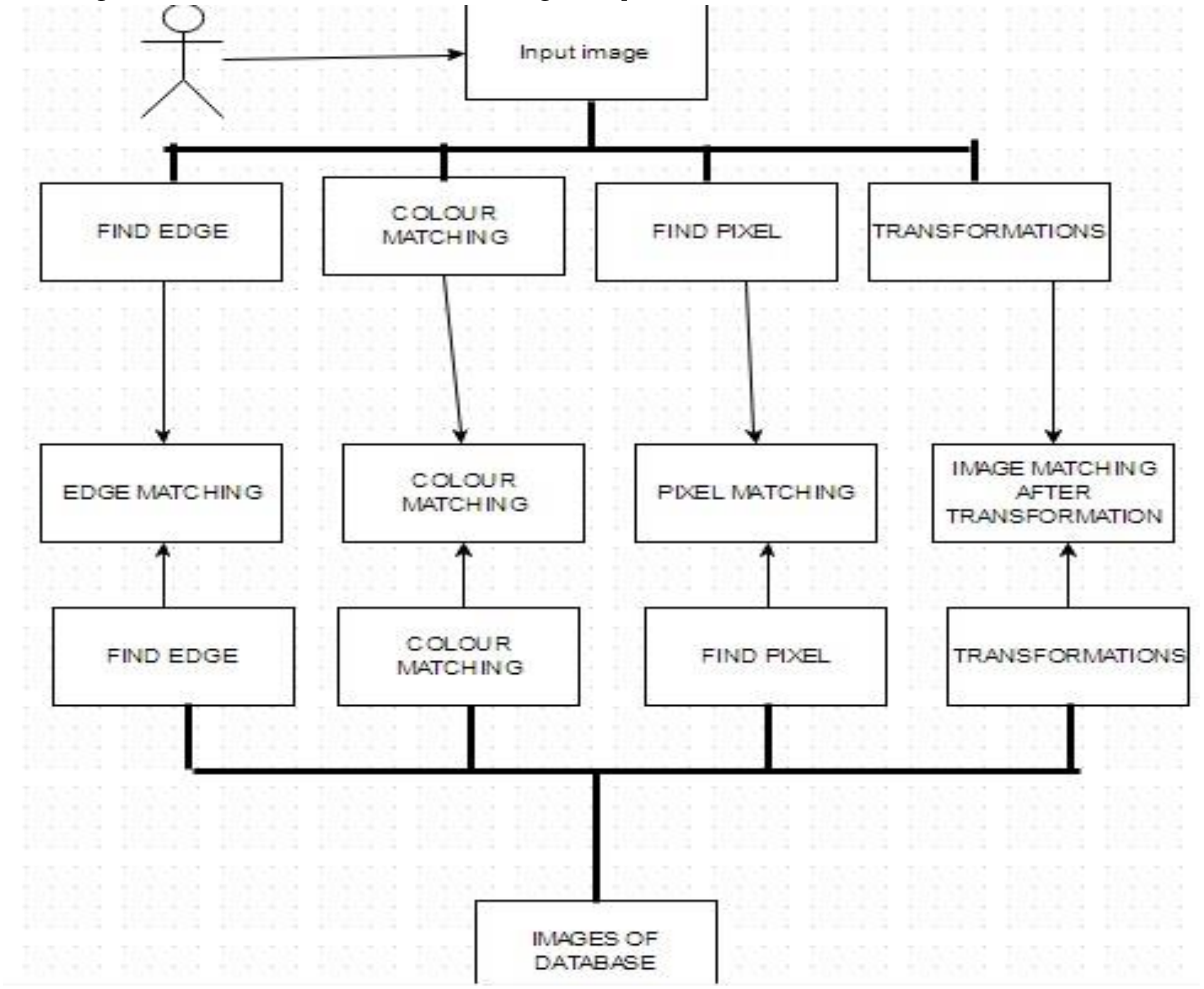


Fig. 1. Flowchart for image search algorithm

A. STEP ALGORITHM FOR EDGE MATCHING:

1) Upload image

2) Find edges of the image by using "find edges" algorithm.

3) Save the image with a name.

4) Save the image containing the edge into an array of pixels.

5) Match each pixel of the above obtained array in RGB [17] configuration of images in the database and find their difference and percentage difference

B. STEP ALGORITHM FOR CONOUR MATCHING:

1) Upload the image

2) Find edges of the image by using "find CONTOUR" algorithm.

3) Save the image with a name.

4) Save the image containing the CONTOUR into an array of pixels.

5) Match each pixel in RGB configuration of the images in the database and find their difference and percentage difference and also show the difference graphically.

C. STEP ALGORITHM FOR PIXEL MATCHING:

1) Upload the image

2) Find the pixel array of the colored version.

3) Save the image containing the CONTOUR into an array of pixels.

4) Math each pixel in RGB configuration and find the difference and percentage difference and also show the difference graphically.

D. STEP ALGORITHM FOR PIXEL TRANSFORMATION:

1) If the user wants to find a matching between his/her uploaded image and any image in the database where he or she has performed transformation in the uploaded image.

E. OUR APPROACH

In this approach, we follow several steps one after another each of which is explained after the steps.

- User can upload any image of his or her choice.
- Detect the edge of the image.
- Store the RGB (Red, Green, Blue) configuration pixel by pixel for each pixel, in an array. Let us call it pixel array, say pa1.
- Repeat step 2 and 3 for each image stored in the database.
- Now, we compare each pixel array ,i.e pa2, pa3 , pa4 with the pixel array of the uploaded image and find the difference and the percentage unmatched.
- Based on the percentage unmatched, we find out if the images are similar or not.
- We show the difference in a graph.
- Thus we find which images match the most and display them.

Each step is described below in detail:

STEP 1

User can upload any image of his or her choice.

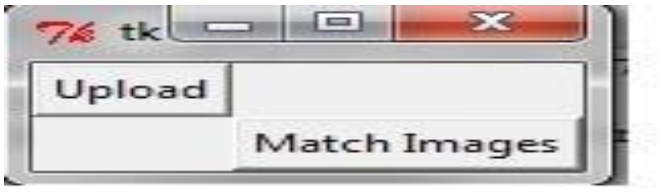


Fig. 2. User can upload any image

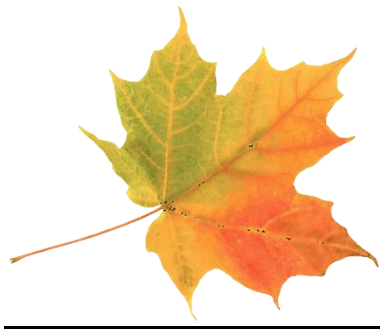


Fig. 3. Input image

STEP 2

Next, we detect the edge of the image.

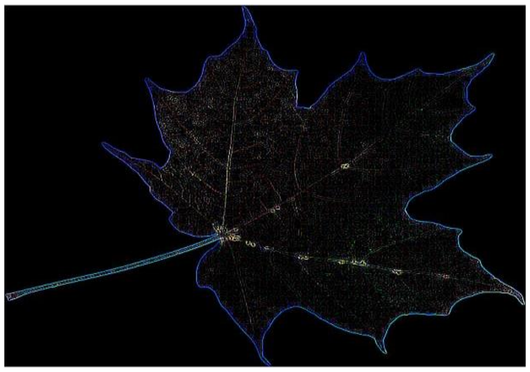


Fig. 4. Image result after Find edges

STEP 3

Next, we find the RGB (Red, Green, Blue) configuration pixel by pixel for each pixel and store it in an array. Let us call it pixel array, say pa1.

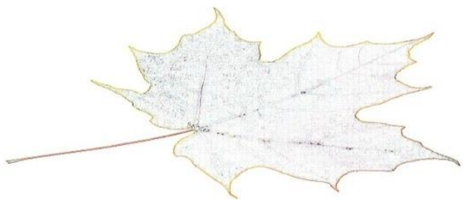


Fig. 5. Result image after contour in found

STEP 4

Repeat step 2 and 3 for each image stored in the database, that is we find out the edge and pixel arrays for each image. Let us name the pixel arrays as pa2, pa3, pa4 and so on.

STEP 5

Now, we compare each pixel array ,i.e pa2, pa3 , pa4 with the pixel array of the uploaded image and find the difference and the percentage unmatched.

STEP 6

Now based on the percentage unmatched, we determine if the images are similar or not. If percentage unmatched is

- less than 20% , then the images are considered to be similar
- greater than 20% but less than 60% , then we rotate the image and find any image matching the transformed, i.e rotated image.
- Greater than 60% , then the images are said to be Unmatched.

STEP 7

Next, we find out the difference between first 20 pixels of two images and plot a graph against pixel number and assume the same pattern to continue for the rest of the pixels.

STEP 8

Thus we find which images match the most, that is find out the images for which the percentage unmatched is the least, and display those images.

V. SCREENSHOTS

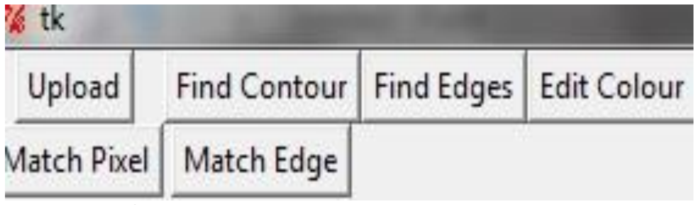


Fig. 6. GUI of the application developed

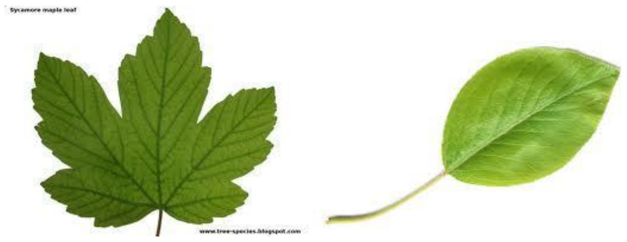


Fig. 7. Output Result images. Percentage differences with input images are also shown for Fig 3 as input.

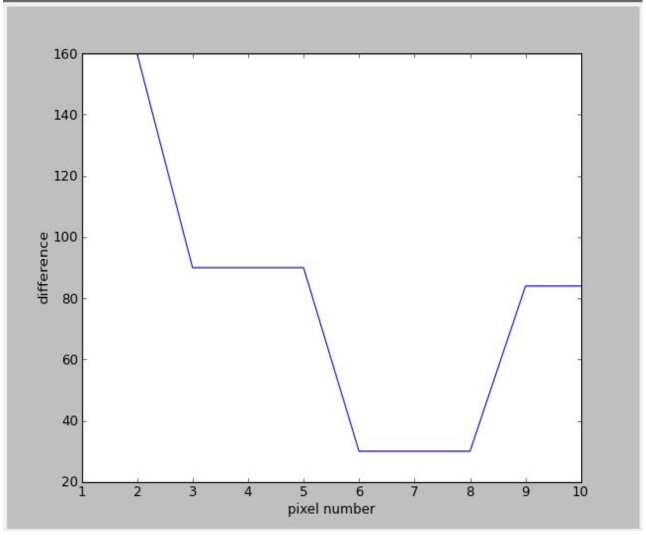


Fig. 8. Graph showing the differences between the two images.

X AXIS: Pixel Number
Y AXIS: Difference.

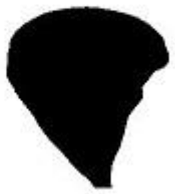


Fig. 9. Input Image

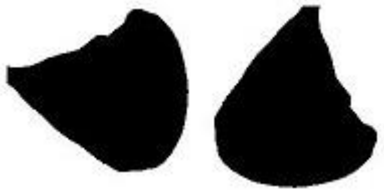


Fig 10(a) Fig 10(b)

Fig 10 (c)

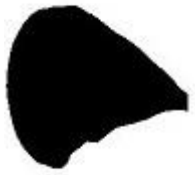


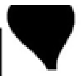
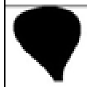
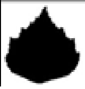
Fig 10 (d)

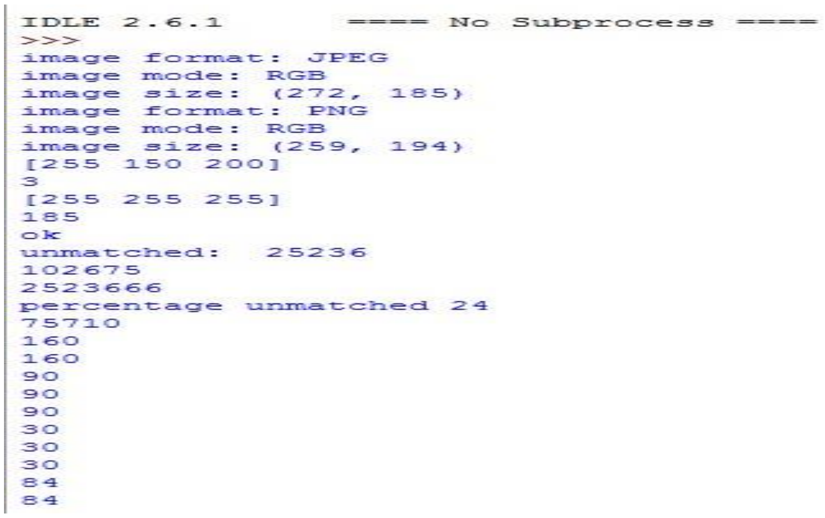
Fig. 10. (a),10(b),10(c),10(d):Output Images of the input image in Fig (9).Input image does not lie in the database. But the rotated versions are present. They are searched and displayed.

Fig. 10. (e)Scaled down version of the input image (Fig 9) occurs in the database and it is displayed

Fig. 10. (f) : Scaled up version of the input image (Fig 9) occurs in the database and it is displayed

TABLE I. PERCENTAGE DIFFERENCE BETWEEN INPUT IMAGE AND DATABASE IMAGES

INPUT IMAG	DATABASE IMAGES 		
	-44.13	-28.25	-54.28
	-28.25	-44.13	-29.56



```
IDLE 2.6.1       ==== No Subprocess ====
>>>
image format: JPEG
image mode: RGB
image size: (272, 185)
image format: PNG
image mode: RGB
image size: (259, 194)
[255 150 200]
3
[255 255 255]
185
ok
unmatched:  25236
102675
2523666
percentage unmatched 24
75710
160
160
90
90
90
30
30
30
84
84
```

Fig. 11. Snapshot of output where percentage differences between input and output images are shown

VI. EXPERIMENTAL EVALUATION

Table 1 shows the percentage differences between the input image and the database images.

VII. DISCUSSION AND CONCLUSIONS

In this study , we have developed an algorithm for shape based image retrieval and image search. We have used an approach where an user uploads an image and first edge detection is done, contour matching is done after contour detection, next pixels are found and stored in an array .Similar steps are performed on database images and percentage differences are found and images are displayed.

Future work will be to improve the algorithm so that the skeleton of images can be found, for example finding leaf skeletons will help in leaf categorizations.

REFERENCES

[1] www.www2008.org/**papers**/pdf/p307-jingA.pdf
[2] http://infolab.stanford.edu/~wangz/project/imsearch/review/JOUR/
[3] en.wikipedia.org/wiki/**Syntactic**_pattern_recognition
[4] en.wikipedia.org/wiki/**Computer_vision**
[5] www.cs.berkeley.edu/~arbelaez/publications/amfm_pami2011.pdf
[6] https://vision.in.tum.de/_media/spezial/bib/rosenhahn_iwcia06.pdf
[7] www.geocomputation.org/1998/99/gc_99.htm
[8] www.dsp.toronto.edu/~kostas/.../pub/.../2000-SpringerMonograph.pdf
[9] http://www.ncbi.nlm.nih.gov/pmc/articles/PMC1390657/
[10] www.springer.com/?SGWID=0-102-1297-33673609-0
[11] www.springer.com/?SGWID=0-102-1297-33673609-0
[12] www.ijarcsse.com/docs/**papers**/Volume_3/11.../V3I11-0501.pdf
[13] www.math.uci.edu/icamp/summer/research_11/bhonsle/cdfd.pdf
[14] en.wikipedia.org/wiki/Orbital_eccentricity
[15] www.mathworks.in/help/matlab/ref/rose.html
[16] www.ascilite.org.au/ajet/ajet28/guven.pd
[17] en.wikipedia.org/wiki/**RGB**_color_model