FRUIT IMAGE CLASSIFICATION USING COLOR AND GEOMETRIC FEATURES

Isac Waern Kyrk
Gothenburg University
guswaeis@student.gu.se

Wafia Adouane
Gothenburg University
gusadowa@student.gu.se

ABSTRACT

This project deals with image classification using color and geometric features. The main goal is to investigate whether composing color and geometric features could increase the distinctiveness of features. So, we extracted geometric features using different techniques and compose them with color information. We also implemented different methods to compare images based on the extracted features (compute the distance between vectors). At the end, we designed a toy system which is meant to teach users about different fruits by showing images and/or descriptions.

1. INTRODUCTION

Image classification consists in finding reliable similarities between images that belong to the same category or represent the same object. According to Martin Szummer, Rosalind W Picard, image classification is a challenging problem that lies on reliably finding similarities among images that represent the same object based on objects' descriptors or in other words describing an image based on the semantic scene it represents (Bay, et al., 2006) (Szummer & Picard, 1998). A variety of techniques have been introduced to deal with this task. Perhaps, the most used techniques in Computer Vision and Image Processing field are the use of SIFT (Scale-invariant feature transform) and SURF (Speeded Up Robust Features) features which are based on extracting invariant feature descriptors. However, both of these techniques are applied to grayscale images. Therefore, color information which plays an important distinctive role in matching and classifying images (at least for humans) is lost. In this project, we used both color and geometric features separately then we combined them.

2. IMPLEMENTATION

We can divide what we did into three main phases: image dataset building, feature detection/extraction and image comparison.

2.1 Image dataset building

Collect manually a set of 180 images of 12 fruits (10-17 images for each fruit) and

annotate them. We will use this image dataset as reference for each category (fruit in our case).

2.2 Features

A set of information which can be anything about an object depending on the task to solve. They can be about the color, size, center, average color, orientation, minimum and maximum intensity,...etc. of the object.

2.2.1 Types of features

The basic idea is to consider image classification as grouping images into semantic classes based on some features. There are two types of features:

1. Visual features: we used OpenCV and SimpleCV libraries to extract invariant representative features (information) which are able to distinguish an image from other images. These features should be invariant to different image transformations such as rotation, illumination, scale, viewpoint, noise,... etc. We need to convert these visual features (vectors) into a human-readable format (a.k.a. semantic classes). This is done by assigning an appropriate semantic class based on the presence of certain features.

2. Lexical features: we based our work on López-Franco's idea "...image classification is associating labels to images based on the existence of an object" (López-Franco, et al., 2014). By the lexical feature or semantic class we mean the label we assign to each image for instance: apricots, grapes...etc.

2.2.2 Feature detection/extraction

We first extracted different feature set which can be grouped as follows:

1. Geometry features: structural Analysis and Shape Descriptors, which are mainly related to Edges, SIFT, SURF, Contours, Skeleton, Canny Edge and Laplacian features.

2. Color information: Hue and HSV histograms

2.2.3 Feature combination

we composed both geometry features and color information in a single feature vector:

-Hue histogram-Edges

-Hue histogram-SIFT

-Hue histogram-SURF

-Contours features

-Hue histogram-Contours

The extracted features are vectors, of different dimensionality. Depending on the features, these vectors are either a simple vector like hue/edge features:

[0.04738562091503268, ….0.00016651104886399005, 0.00023498288204170558 …... 0.01682695300342359, 0.005669156551509492 4, ……, 0.0, 0.0, 0.0, 0.0, 0.0]

or a vector of vectors like SIFT/SURF features:

```
(array([[ 91.,  27.,   2., ...,   1.,   0.,   1.],
        [  4.,   3.,   4., ...,   1.,   0.,   4.],
        [  2.,   1.,   1., ...,   1.,   6.,   3.], ...,
        [  0.,   0.,   0., ...,   0.,   0.,   0.],
        [  0.,   0.,   0., ...,   0.,   0.,   0.]], dtype=float32),)
```

To compose different features, we used two methods:

1. Clustering: reduce each vector of vectors to a representative value or values. In our case, we computed the mean of each vector, i.e. turned a vector in a single number. We used two kinds of means: arithmetic and geometric means. However, in image processing, the K-means is usually used for clustering or computing the visual words. We simplified things and say we just did the same but took k = 1.

2. Use all information: we considered all the information (descriptors) and flatten/ stack[1] them using NumPy.

It is known that the number of detected descriptors differs depending on the image. First, we detected and extracted only 200 descriptors for each image and limited their dimensionality to 50. Then, we extracted all possible descriptors for each image without limiting their dimensionality (128 for each SIFT descriptor and 64 for each SURF descriptor). However, to be able to apply any machine learning method, we need to have equal length vectors. To do so, we computed the number of descriptors for each image in the dataset then conducted three experiments:

a. Take the smallest number of descriptors (minimum length = lenMin) and reduce the number of descriptors, for each image, to that number.

b. Take the biggest number of descriptors (maximum length = lenMax) and enlarge the number of descriptors, for each image, to that number. This is done by creating new array V_0 of length L filled with zeros.

For each vector V_i of length L_i:

$$\mathbf{L = lenMax - L_i}$$

We implemented this using NumPy as follows:

$$\mathbf{V_0 = zeros((lenMax - L_i))}$$

Then concatenate V_i and the new created array V_0 to create a new array V as follows:

$$\mathbf{V = np.concatenate((V_i, V_0))}$$

c. Take a random number of descriptors

2.3 Image comparison

So far, we detected and extracted different visual features in a format of vector features of different dimensions and mapped each of them to a lexical feature. Now, we take this information as a representative feature of each category. What we need to do after is to find how to assign a new image to a particular category. In another words, how to match two images? To do so, we implemented different methods based on computing distance between two feature vectors.

2.3.1 Distance measures

[1] We used two NumPy methods: np.dstack() and np.array().flatten() The first takes a sequence of arrays and stack them to make a single array and the second returns a copy of the array collapsed into one dimension.

Comparing images can be done either by computing their similarity or difference. To do so, we simply compare their feature vectors using the different methods implemented in OpenCV, namely "Correlation", "Chi-Squared" and "Intersection", "Bhattacharyya". In addition to this, we played also with some other methods Scipy offers, namely "Euclidean", "Manhattan", "Chebysev".

We also implemented a confidence degree method and set its threshold to 50%, meaning to consider the system output, its confidence should be at least 50%. This is done by implementing the Levenshtein distance method which is basically used for string sequence but we adopted it to our needs where we converted each vector into a string. This method works well in most cases. Moreover, we implemented Cosine similarity matrix.

For fun, we implemented our own simple method which is just taking the difference between two vectors. For instance: say we have two NumPy arrays A and B of the same dimension. We used a NumPy class called **np.linalg.norm** to compute the norm matrix using linear algebra. The difference is computed as follows:

diff = np.linalg.norm(np.array(A)-np.array(B))

To our surprise, this works just fine!

3. Results and discussions

As described above, we extracted different features (color and geometry), implemented different methods for manipulating these features (flattening, stacking and computing means) and implemented various ways for computing the distance between images (vectors).

Next, we used supervised machine learning technique for classification. We trained some Sklearn classifiers namely SVM, KNNs, DecisionTree, MultinomialNB, Bagging Classifier using different classifier combination, RandomForest Classifier. In our case, RandomForest Classifier got the best results. We used 80% of our image dataset for training and 20% for testing (all images of the test dataset have not been seen yet). It is amazing that when we trained on 50% and test on the rest 50% we got an accuracy (using Hue features) of 54%. We played on the training/test size and found that the training/test size did not have really an effect as all images are unique. In all cases, when we trained and test on the same dataset we got an accuracy of 100%.

We conducted some classification experiments using color and geometry features separately then combined. The obtained results are shown in Table 1

Features	Method	Accuracy
Hue features	None	70%
Edge features	None	22%
SIFT features	Geometric mean	47%
SIFT features	Arithmetic mean	5.5%
SIFT features	All descriptors	28%
SURF features	Geometric mean	42%
HUE-SIFT	Geometric mean	55%
SIFT-EDGE	Geometric mean	39%
HUE-SIFT	All descriptors	25%
Contours	All descriptors	17%
Contours_HUE	All descriptors	11%

Table 1: Results

In the above table, we meant by 'Method', the method used to collapse vector of vectors into one dimension vector (stack, cluster or compute the mean). 'None' means no method was applied namely for Hue and Edge features because we applied SimpleCV methods (HueHistogramFeatureExtractor and EdgeHistogramFeatureExtractor) which return a one dimension array. 'All descriptors' means all vectors of different dimensionality are stacked/flattened into one dimension array. As described in the Use all information section above, we took first the minimum length (min), the maximum length (max) and a random length between the min and max. It turned out that taking the maximum length performed better in all cases. That is the main reason we used all of the extracted information. 'Geometric mean' or 'Arithmetic mean' means we reduced an entire vector of Nth dimensionality to one single number and concatenate all the numbers to get at the end a one dimension array.

The results above are expected because of the nature of the used image dataset where most of the shapes are the same also hard to tell what the category is based on the color only. We played a bit with some other images and got much better accuracies in some cases and got worse in some others.

4. Conclusions

The main points we learned from this part of the project can be summarized as follows:

- The choice of any method or technique depends heavily on the image dataset and what we want to achieve.

- Color features have more discrimination power than geometry features in our case (accuracy of 70% for Hue features and less for all other geometric features) and this can be explained by the nature of our image dataset where most fruits have the same shape.

- Composing color information and geometric characteristic did not really help to better discriminate images compared to using only color features. The accuracy dropped from 70% to 55%.

- Composing color information and geometric characteristic helped to better discriminate images compared to using only geometric features. Compare SIFT and HUE-SIFT for geometric. The accuracy increased from 47% to 55%. Hence, color improves classification. In a different dataset where SIFT would be more informative but color less, this would make a difference.

- There is no 'the best' technique, method or features in the absolute meaning. It all depends on the dataset and the task to solve.

5. Learning new images and dataset automatic extension

We manually annotated our training dataset which we took as a representative for each category (fruit). First there is offline learning and then once the system is in use, it can also learn online. We wrote a code which guesses the category of a new fruit image and asks for confirmation. If the guess was correct, that is ok, else the system asks the user to teach it what the correct category is and adds it automatically to its dataset (with the correct label). There is a possibility to take a picture using a webcam

or even consider all the possible rotations (it is implemented as a parameter a user has to set).

The idea is taken from the following scenario:

Imagine a child (3- 4 years old) who has a lovely fluffy cat as a pet (which looks like figure 1). One day, the child went with his/her father for a walk. They met with an old person who has a fluffy little dog (which looks like figure 2), the child said: 'look daddy it is a cat!' The father corrected him/her: 'no dear, that's a dog'. Then, the child learned at least to distinguish between these two animals (both already seen).

Figure 1

Figure 2

6. FRUIT TOY

For fun, we implemented some sort of fruit toy to teach users what a given fruit is (returns the wikipedia description and an image). Here we have two groups of fruits: those who are included in the training dataset, i.e. the system can recognize them and those which are not yet included. For the last case, the system is able to return a description and one reference image.

Here are two example dialogues the system can handle:

1. The fruit is covered in the dataset:

S: Welcome to the FRUIT WORLD, What fruit are you looking for?
[Greeting]
U:
S: You did not enter anything, you need to specify which fruit you are looking for.
[Ask what object (fruit) to look for]
U: banana
S: I can show you an image or a description of banana. What do you want me to do?
[Display the menu-ask what task to perform]
U:
S: You need to tell me what you want me to do with banana. I can show you an image or a description.
[Display the menu-ask what task to perform]
U: description
S: The banana is an edible fruit, botanically a berry, produced by several kinds of large herbaceous flowering plants in the genus Musa. In some countries, bananas used for cooking may be called plantains. The fruit is variable in size, color and firmness, but is usually elongated and curved, with soft flesh rich in starch covered with a rind which may be green, yellow, red, purple, or brown when ripe.
[Show description-execute the chosen task]
S: Perhaps you want also to see an image of banana?
[Display the other possibility of the menu]
U: no, thanks
S: Talk to you soon, goodbye!
[Quit]

2. The fruit is not covered in the dataset:

S: Welcome to the FRUIT WORLD, What fruit are you looking for?
[Greeting]
U: cantaloupe
S: Unfortunately, for now, I can not recognize cantaloup but I can tell you what this fruit is. Do you want me to show you its description?
[Display the menu-ask]
U: yes
S: Cantaloupe (also cantelope, cantaloup, muskmelon (India and the United States), mushmelon, rockmelon, sweet melon, honeydew, Persian melon, or spanspek (South Africa)) refers to a variety of Cucumis melo, a species in the family

Cucurbitaceae. Cantaloupes range in weight from 0.5 to 5 kilograms (1.1 to 11.0 lb). Originally, cantaloupe referred only to the non-netted, orange-fleshed melons of Europe.

[Show description-execute the chosen task]

S: I can also show you a reference image of cantaloupe. Do you want me to do so?

[Display the other possibility of the menu]

U: yes, please

S: Here is how Cantaloupe should look like:

S: Talk to you soon, goodbye!

[Quit]

7. FUTURE WORK

In this project, we only explored some techniques and methods due to the time constraint and the limitation of our experience. Therefore, as a future work, we would like to explore more techniques and combination of features, namely spatial color histogram. Moreover, we would like to extend the current list of fruits to cover all fruits listed in Wikipedia. There is also another thing we would like to improve. For now, we do not really make difference for instance between an image of one banana and an image which contains bananas. The only exception is for apples where we annotated separately green-apple, red-apple and apples. Simply because it is time consuming to build such distinctive image dataset. Even then, there is no guarantee that our system will be able to make a clear distinction between singular and plural fruits because if an image contains bananas, it is more likely that the system classifies it as banana also because it contains actually a banana. We would like to explore further this issue. Also, it would be interesting if we collect all the implemented methods, techniques and feature extraction in one code able to run different combinations and return the one which performs the best.

8. GLOSS

Color histograms: Histograms that provide organised information about color distribution of either an image or a sub-image in a single line (Sergyan, 2008). They take into account features such as contrast, brightness and intensity distribution of colors in an image.

http://practicalphotographytips.com/Digital-Camera-Basics/color-histograms.html

Levenshtein distance: A string metric for measuring the difference between two sequences. Informally, the Levenshtein distance between two words is the minimum number of single-character edits (i.e. insertions, deletions or substitutions) required to change one word into the other.

http://people.cs.pitt.edu/~kirk/cs1501/Pruhs/Fall2006/Assignments/editdistance/Levenshtein%20Distance.htm

OpenCV: A free open source library for computer vision and image processing**.** For more information: https://opencv-python-tutroals.readthedocs.org/en/latest/index.html

SimpleCV: An open source framework for building computer vision applications with access to several high-powered computer vision libraries such as OpenCV – without having to first learn about bit depths, file formats, color spaces, buffer management, eigenvalues, or matrix versus bitmap storage. http://simplecv.org

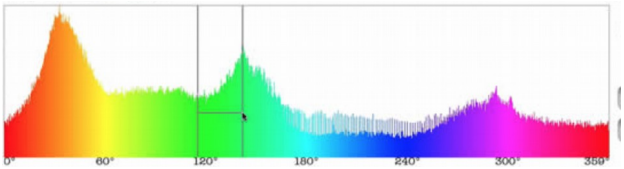
Scale-invariant feature transform (SIFT): An algorithm in computer vision to detect and describe local features in images. They are rotation-invariant as well as scale-invariant, which means, even if the image is rotated, we can find the same corners.

http://opencvpythontutroals.readthedocs.org/en/latest/py_tutorials/py_feature2d/py_sift_intro/py_sift_intro.html

Speeded Up Robust Features (SURF): As the name suggests, it is a speeded-up version of SIFT.

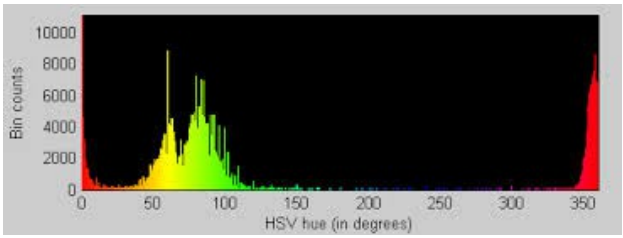
http://opencv-python tutroals.readthedocs.org/en/latest/py_tutorials/py_feature2d/py_surf_intro/py_surf_intro.html?highlight=surf

Hue/Hue histogram: Hue refers to a gradation or variety of a color or tint. It is the property of light by which the color of an object is classified as red, blue, green, or yellow in reference to the spectrum. Hence, it is one of the main properties (called color appearance parameters) of a color, defined technically as "the degree to which a stimulus can be described as similar to or different from stimuli that are described as red, green, blue, and yellow" (the unique hues). Orange and violet (purple) are the other hues, for a total of six, as in the rainbow: red, orange, yellow, green, blue, violet. The other color appearance parameters are colorfulness, chroma, saturation, lightness and brightness. A hue histogram looks as the following:



https://en.wikipedia.org/wiki/Hue

HSV/HSV histograms: HSV stands for hue, saturation, and value, and is also often called HSB (B for brightness). It is one of the most common cylindrical-coordinate representations of points in an RGB color model. It rearranges the geometry of RGB in an attempt to be more intuitive and perceptually relevant than the cartesian (cube) representation. An HSV histogram looks as follows:



Spatial color histograms: Histograms used to preserve some local information in the image. Instead of doing one histogram for the whole picture, the image is sliced into n x n smaller pieces, like 24×24, and make a histogram for that piece only. Collecting all the small pieces in one part forms what is called spatial histogram. For more details, see:
http://www.ee.oulu.fi/mvg/files/pdf/pdf_730.pdf

NumPy: A package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting...etc.
http://docs.scipy.org/doc/numpy1.10.1/user/index.html

SciPy: A set of open source scientific and numerical tools for Python. It currently supports special functions, integration, ordinary differential equation solvers, gradient optimization, parallel programming tools...etc. http://scipy.org

Arithmetic mean: A mathematical representation of the typical value of a series of numbers, computed as the sum of all the numbers in the series divided by the count of all numbers in the series. It is commonly referred to as "average" or simply as "mean".
http://www.investopedia.com/terms/a/arithmeticmean.asp

Geometric mean: The average of a set of products, the calculation of which is commonly used to determine the performance results of an investment or portfolio. Technically defined as "the 'n'th root product of 'n' numbers". For n numbers: multiply them all together and then take the n'th root (written $\sqrt[n]{}$). More formally, the geometric mean of n numbers a_1 to a_n is:

$$\sqrt[n]{(a_1 \times a_2 \times \ldots \times a_n)}$$

https://www.mathsisfun.com/numbers/geometric-mean.html

K-means: One of the simplest unsupervised learning algorithms that solve the well known clustering problem. The procedure

follows a simple and easy way to classify a given data set through a certain number of clusters (assume k clusters) fixed a priori. The main idea is to define k centroids, one for each cluster.

http://home.deib.polimi.it/matteucc/Clustering/tutorial_html/kmeans.html

Sklearn: Scikit-learn (formerly scikits.learn) is an open source machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, *k*-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

Cosine similarity matrix: A measure that calculates the cosine of the angle between two vectors (or two documents on the Vector Space). This metric is a measurement of orientation and not magnitude, it can be seen as a comparison between documents on a normalized space.

http://blog.christianperone.com/2013/09/machine-learning-cosine-similarity-for-vector-space-models-part-iii/

Edges: Edge Histogram Descriptor (EHD) describes edge distribution with a histogram based on local edge distribution in an image.http://citeseerx.ist.psu.edu/viewdoc/downloaddoi=10.1.1.182.5856&rep=rep1&type=pdf

Contours: Curves joining all the continuous points (along the boundary) of an object, having same color or intensity.

http://docs.opencv.org/2.4/doc/tutorials/imgproc/shapedescriptors/bounding_rects_circles/bounding_rects_circles.html?highlight=contours#theory

Skeleton: Skeletonization/Medial Axis Transform is a process for reducing foreground regions in a binary image to a skeletal remnant that largely preserves the extent and connectivity of the original region while throwing away most of the original foreground pixels.

http://homepages.inf.ed.ac.uk/rbf/HIPR2/skeleton.htm

Correlation: In statistics and in probability theory, distance correlation is a measure of statistical dependence between two random variables or two random vectors of arbitrary, not necessarily equal dimension. An important property is that this measure of dependence is zero if and only if the random variables are statistically independent. This measure is derived from a number of other quantities that are used in its specification, specifically: distance variance, distance standard deviation and distance covariance.

https://en.wikipedia.org/wiki/Distance_correlation

Chi-Squared distance: A distance measure d(x, y) is a method used to compute the distance between two histograms x=[x_1,..,x_n] and y=[y_1,...,y_n] having n bins both.

d(x,y) = sum((xi-yi)^2 / (xi+yi)) / 2 . It is

often used in computer vision to compute distances between some bag-of-visual-word representations of images. The name of the distance is derived from Pearson's chi squared test statistic $X^2(x, y) = sum((xi-yi)^2 / xi)$ for comparing discrete probability distributions (i.e histograms).

http://www.umass.edu/landeco/teaching/multivariate/readings/McCune.and.Grace.2002.chapter6.pdf

Intersection distance: A similarity measure for images used to calculate intersection between histograms. It is even proved to be an effective kernel for machine learning

http://docs.opencv.org/2.4/modules/imgproc/doc/histograms.html

Euclidean: The straight line distance between two points. In a plane with p_1 at (x_1, y_1) and p_2 at (x_2, y_2), it is $\sqrt{((x_1 - x_2)^2 + (y_1 - y_2)^2)}$.

http://www.cut-the-knot.org/pythagoras/DistanceFormula.sstml

Manhattan: The distance between two points measured along axes at right angles. In a plane with p_1 at (x_1, y_1) and p_2 at (x_2, y_2), it is $|x_1 - x_2| + |y_1 - y_2|$.

https://xlinux.nist.gov/dads//HTML/manhattanDistance.html

Chebysev distance: Called also Tchebychev distance, maximum metric, or L_∞ metric in mathematics. It is a metric defined on a vector space where the distance between two vectors is the greatest of their differences along any coordinate dimension. It is named after Pafnuty Chebyshev.

https://en.wikipedia.org/wiki/Chebyshev_distance

Canny Edge: The Canny edge detector is an edge detection operator that uses a multi-stage algorithm to detect a wide range of edges in images. It was developed by John F. Canny in 1986. Canny also produced a *computational theory of edge detection* explaining why the technique works.

http://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/canny_detector/canny_detector.html

Bhattacharyya: In probability and statistics, the Bhattacharyya distance as this was originally introduced by Anil Kumar Bhattacharya) and also called Hellinger distance is used to quantify the similarity between two probability distributions.

https://en.wikipedia.org/wiki/Hellinger_distance

Laplacian features: In the mathematical field of graph theory, the Laplacian matrix, sometimes called admittance matrix, Kirchhoff matrix or discrete Laplacian, is a matrix representation of a graph. Together with Kirchhoff's theorem, it can be used to calculate the number of spanning trees for a given graph. The Laplacian matrix can be used to find many other properties of the graph.http://www.owlnet.rice.edu/~elec539/Projects97/morphjrks/laplacian.html

9. References

[1] OpenCV, python tutorial is the main information source used in this project.

https://opencv-python-tutroals.readthedocs.org/en/latest/index.html

[2] SimpleCV

http://simplecv.readthedocs.org/en/1.0/

[3] Indoor-outdoor image classification, M Szummer, RW Picard – Content-Based Access of Image and Video Database,1998.

[4] Timo Ahonen, Abdenour Hadid, and Matti Pietikäinen, "Face Description with Local Binary Patterns: Application to Face Recognition", 2006 (draft version) http://www.ee.oulu.fi/mvg/files/pdf/pdf_730.pdf

[5] Carlos López-Franco, Luis Villavicencio, Nancy Arana-Daniel, and Alma Y. Alanis, "Image Classification Using PSO-SVM and an RGB-D Sensor", Hindawi Publishing Corporation Mathematical Problems in Engineering Volume 2014, Article ID 695910, 17 pages http://dx.doi.org/10.1155/2014/695910

[6] Image Processing guide: http://homepages.inf.ed.ac.uk/rbf/HIPR2/wksheets.htm