Identification of Paintings from Camera-Phone Images

Raül Llorach Sánchez
Master en Programari lliure, Universitat Oberta de Catalunya
raulllorach@gmail.com

Abstract

This paper presents the algorithm developed, for final master degree project, to identify paintings from images taken with a camera-phone from a set of prepared image collection. The image processing algorithm first needs the database to be created by extracting the painting of interest, in the center of the photo. For doing that first of all the image is resized keeping the original aspect. Secondly it is converted to gray color. Now some work to remove noise is needed and it is done with an algorithms that performs down-sampling step of Gaussian pyramid decomposition. And then an algorithms that performs up-sampling step of Gaussian pyramid decomposition. After that Canny algorithm is applied to get all edges. Then is applied a probabilistic hough transform to find the line segments and from them, with a little processing, it is selected just four which presumably will be the ones that forms the frame of the painting. Calculating the intersection points of these lines we get four points and with them a perspective transform is applied to get just the painting image. Finally feature identification is performed on this image using SURF algorithm. All these images are used as a training images for the learning machine algorithm k nearest neighbors. On the other hand, different images are taken with a phone camera from the previous treated. This time, the captured image is not treated in the same way, instead, these images are just resized, converted to gray color and finally compute their features with SURF[1] *method. Then with k nearest neighbors algorithm it can be identified which is image that represents and with that possible information can be given to the user.*

1. Introduction

Recognition of objects in images has practical applications across many disciplines. One potential “Augmented Reality” application is the recognition of paintings in an art exhibit. Gallery visitors could take a picture of a painting with a cell phone, and then receive an automated phone call or text message containing information about the painting such as title, artist, historical context, and critical review. This service requires rapid identification of the painting in the image, without strong guarantees on lighting, viewing angle, and size of the painting in the image. Compression artifacts and low light effects, such as blur and noise, corrupt such images to varying degrees. The painting recognition task can be automated using techniques from the fields of digital image processing and computer vision.

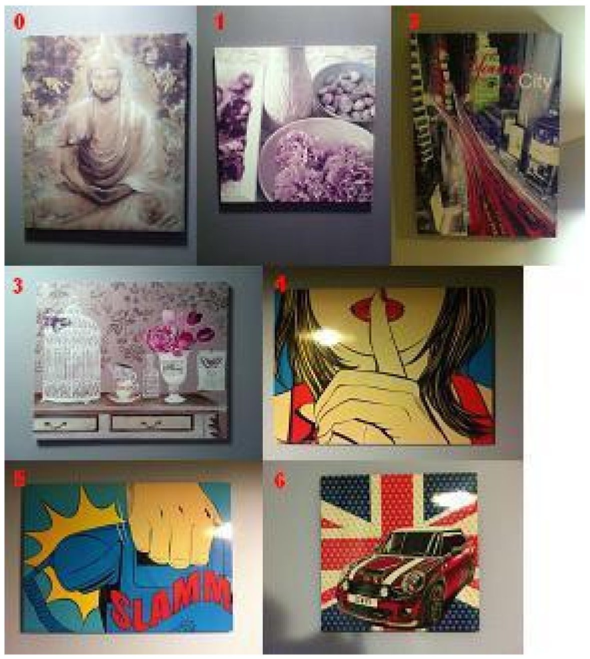

Figure 1: The 7 images selected in this study.

After consideration and prototyping of several candidate techniques, it was found that this problem has some unique characteristics. First, in some scenes it may be quite difficult to detect and extract the frame of the painting. In fact, in that sense it is needed more work that what it has been done here. In our experiments some assumptions has been assumed.

This paper presents the algorithm developed for the master degree final project, to identify paintings in the centers of query images taken with a camera-phone.

The overall image processing algorithm is divided in two parts. The first part consists on generating a database with all feature matrix of each image selected in this study. To do that, each image is processed in order to extract the foreground and get just the painting.

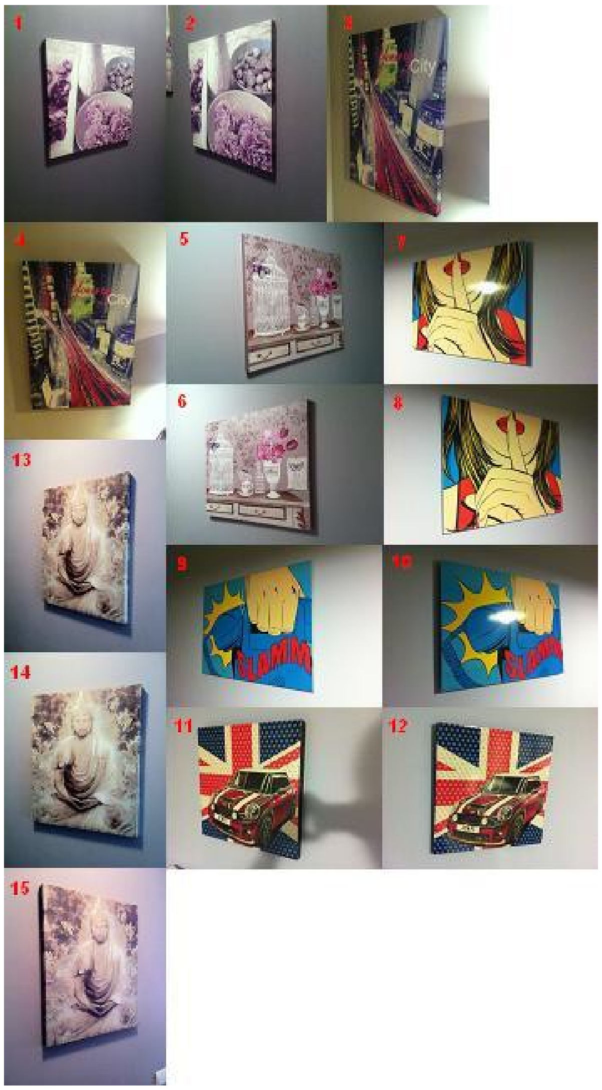

Figure 2: The 15 query images tested.

The resulting painting is computed the features that will identify it uniquely and then stored in a database. The second part consists on given a taken photo from a painting, resize it to be in the same proportions than the ones in database and without any rectification compute directly their features with SURF method. Finally these features are passed to the k nearest neighbors algorithm that detect the most similar feature matrix from the database that will probably will be the same image than the photo queried. All this program has been done with emguCV framework which is basically an open source wrapper of the openCV library. The overall algorithm has detected correctly all testing images.

2. Background

There are two major challenges involved in the art recognition application described above. First, there must be a way to extract the region of interest from the image. Second, it is necessary to describe this region in a distinctive way, one that is repeatable and can be cross referenced to a database of information that uniquely identifies the painting in question.

In order to extract a section of an image, it is necessary to have some information about the region in question. For this application, it is assumed that the painting to be added in the database must be photographed alone and roughly centered in the image frame. More specifically, the center pixel in the image is assumed to be part of the painting of interest.

The image processing field employs many techniques for separate different regions of an image. Image decomposition are one way, for example, to remove some noise from the content of a digital image. Many times, this can be useful to separate different regions of the image or get rid of some invaluable information of the image. Examples of such decompositions include the down-sampling step of Gaussian pyramid decomposition and the up-sampling step of Gaussian pyramid decomposition.

Another field from the image processing is the one aimed to classify and identify images[2]. There are many types of features that can be extracted from an image, ranging from simple things like corners and edges to more complicated things like SIFT[3] vectors or SURF vectors. Corners and edges are popular because they are easily recognizable and easily verified as being correctly found in the image. However, by themselves, these features are not especially good for repeatably and uniquely classifying an image.

It is desirable to have features which have some invariance to common image differences, such as changes in scale and illumination or even affine transformation. The Scale Invariant Feature Transform, or SIFT, method is a very popular algorithm for generating such features. The descriptor vector is created by histogramming gradient values in a scaled window around a key-point. This method has proved to be quite robust to moderate perspective transformation. A large volume of recent work has been based on SIFT features and much support for the use of SIFT exists in the literature. A more recent addition to the field is the Speeded Up Robust Features, or SURF, descriptor. The authors of SURF claim that this method outperforms previously proposed schemes with respect to repeatability, distinctiveness, and robustness, yet can be computed and compared much faster.

3. Image processing algorithm

This section details the implemented algorithm for painting recognition. It is robust to changes in lighting and perspective. It is composed of two different parts. The first part pretends to create the database of all paintings on the study. Each painting is segmented out of the image and rectified to a straight-on view. Then a SURF is performed on the rectified painting and the resulting feature descriptors are stored in the database. The second part of the algorithm is when a photograph is taken from one of the paintings in the exposition and normalized to be similar to the ones in database. Then a SURF is performed to the photo to extract their feature descriptor[4] which in turn will be passed to the machine learning algorithm called k nearest neighbors that will compare with all the paintings of our study will return the painting with the most similar descriptor that presumably will be the painting from the photographed.

3.1. Painting rectification

First of all, the image is resized with the method Resize from emguCV that scale the image to specific size. This is done to reduce JPEG artifact effects. Next to this step the image is converted to gray color with the method Convert. This is necessary to apply two methods (PyrDown and PyrUp) that perform down-sampling and up-sampling step of Gaussian pyramid decomposition. This is done to remove some noise of the image and it will be useful for the next steps. Now it is applied Canny algorithm that will find edges[5] on the image and will mark them.

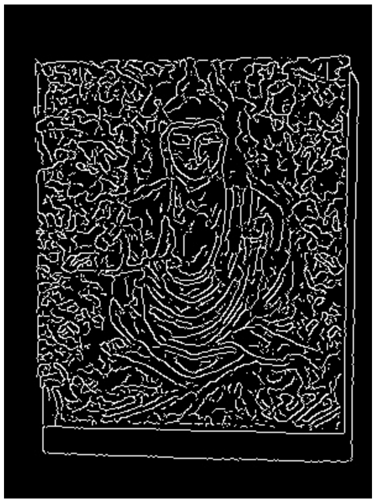
Figure 3: Result of Canny algorithm

From the edges detected with Canny algorithm it is applied a probabilistic hough transform to find line segments. This method can be parametrized to control its behavior like how many pixels to consider it a line, etc. In our case study we find the values that fits best for all the training images.

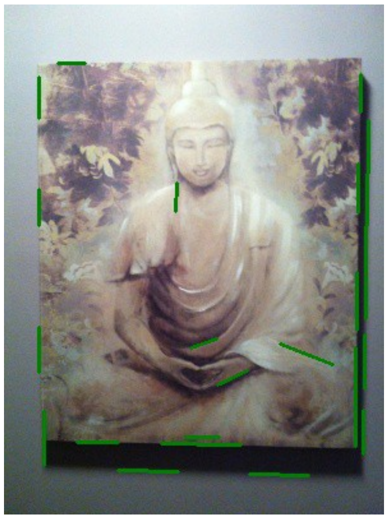
Figure 4: Result of probabilistic hough transform

Now, with some image processing it is possible to extract those external lines that are more probable to be the frame of the painting. With these lines selected it is possible to expand them and detect where they cross each other and with these points we can crop the image to obtain just the painting and get rid of all the rest of the image.

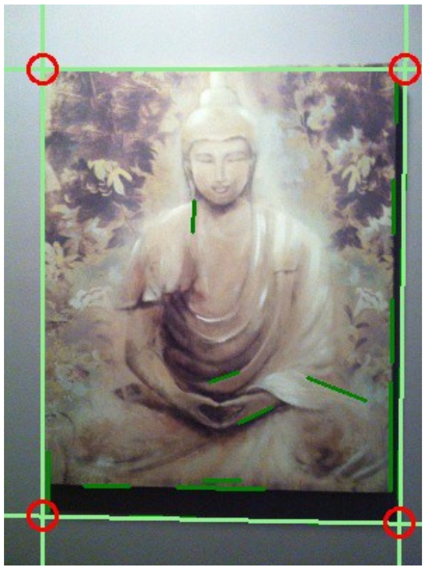
Figure 5: Detection of all intersection lines

Now it's time to crop the image and also to correct the perspective if the image has been taken with a little perspective. Nevertheless the algorithm requires that the image taken to be with the less perspective as possible. In order to get this perspective transformation it has been used the method GetPerspectiveTransform that returns the HomographyMatrix, then it is passed to the method WrapPerspective that will return the image transformed. Then each image object has the property ROI (region of interest) that is useful to define the region to be cropped, in this case will be the dimensions of the frame detected.

The final step before store it in database is to compute their descriptor that is done with SURF algorithm. This is done with the method ComputeDescriptorsRaw from the object SURFDetector. This method will return a matrix of floats and will be used as a training example for the machine learning algorithm k nearest neighbors. These are the steps that will be applied in the same way for all training images.

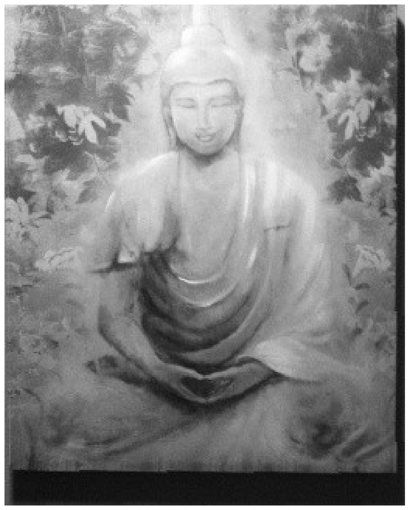
Figure 6: Image transformed and cropped

3.2. Painting identification

3.2.1 K Nearest Neighbors

The emguCV framework has some libraries related to machine learning field[6]. Here we have used the KNearest class to classify/identify a given descriptor. To prepare this algorithm it is needed two data structures. The first one is a matrix of floats that corresponds to all descriptors from the initial paintings captured and prepared, this matrix is known as training data. The second structure is a matrix of floats that will associate each zone of the first matrix with a specific float value. The first zone of one descriptor painting will be fill with 0 and so on.

3.2.2 Painting identification with K Nearest

To identify a given painting we compute their descriptor directly from the image taken from the phone camera. This descriptor is passed to the KNearest object which will compute a punctuation among all the training examples. The most important aspect here is that the descriptors from the image queried will probably be more similar with the original image normalized at the first stage. That's why the nearest descriptor found should be the original image.

To achieve it, we have developed a very simple punctuation method. The matching is done row by row from the descriptor matrix. For each row it is found the most nearest image whose descriptor is most similar. The image returned gets an additional point and so on for the rest of rows. At the end, the one with the highest punctuation will be the painting in the photograph.

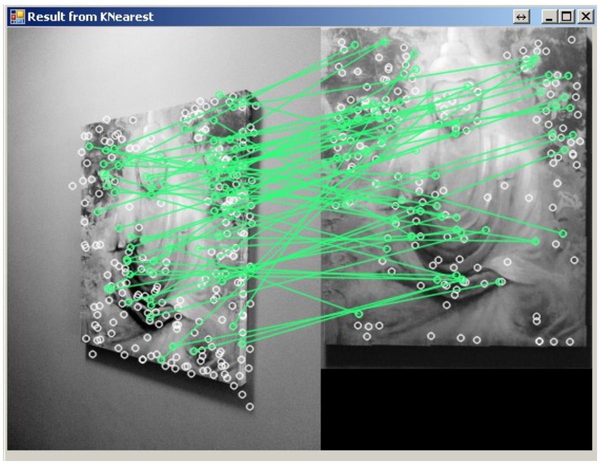

Figure 7: Example of surf feature matching

At the end of the algorithm in order to show the resulting image we have used a method in the emguCV framework that shows the features detected between two images and also it tries to represent with green lines the exact match of one feature between both images. As we can observe in the figure 7, some features are not matched in green. This is done by the method *VoteForUniqueness*. This method can select among all features just the ones which are more unique.

3.2.3 Normal Bayes Classifier

A naive Bayes classifier is a term in Bayesian statistics dealing with a simple probabilistic classifier based on applying Bayes theorem with strong (naive) independence assumptions. A more descriptive term for the underlying probability model would be "independent feature model".

In simple terms, a naive Bayes classifier assumes that the presence (or absence) of a particular feature of a class is unrelated to the presence (or absence) of any other feature. For example, a fruit may be considered to be an apple if it is red, round, and about 4" in diameter. Even though these features depend on the existence of the other features, a naive Bayes classifier considers all of these properties to independently contribute to the probability that this fruit is an apple.

In spite of their naive design and apparently over-simplified assumptions, naive Bayes classifiers often work much better in many complex real-world situations than one might expect. Recently, careful analysis of the Bayesian classification problem has shown that there are some theoretical reasons for the apparently unreasonable efficacy of naive Bayes classifiers. An advantage of the naive Bayes classifier is that it requires a small amount of training data to estimate the parameters (means and variances of the variables) necessary for classification. Because independent variables are assumed, only the variances of the variables for each class need to be determined and not the entire covariance matrix.

3.2.4 Painting identification with Normal Bayes

The emguCV framework has a class named NormalBayesClassifier that lets you classify an specific item according to the training examples passed to the classifier. We have applied this algorithm with our scenery, the paintings. First of all we use the image descriptors gathered on the first stage as training samples for the algorithm. Basically for each image descriptor is associated a type or class. Then, a painting is computed the SURF algorithm and extracts its descriptor. It is passed to the classifier and for each row it is applied the NormalBayes algorithm that returns the training image that may correspond to.

As the previous classifier, a punctuation system has been developed and consists on adding points on each row result. At the end the training image with more points is the winner. In our little scenery has showed that this algorithm is less effective that the previous one which has identified all the query images correctly. This one, instead, has fail on one of the query images.

Another experiment has been done with SIFT method to extract all the image descriptors. In that case while with the KNearest algorithm has identified all query images correctly, with the Normal Bayes classifier has just identified correctly two query images.

4. Conclusion

With the idea of implementing an application of “Augmented Reality”, an algorithm has been developed to identify paintings from a sample scenario. In the algorithm, firstly the paintings are photographed in order to extract their unique descriptors and store it in a database. It is done by detecting their frames and then applying a perspective transformation. Then visitors take photos from the sample paintings and they are identified through a machine learning algorithm like k-nearest neighbor and normal Bayes.

The developed algorithm is capable of identifying correctly all training images using either SURF or SIFT descriptor extractor and k nearest neighbor as the classifier algorithm. With SURF descriptor extractor but with normal Bayes classifier has identified all paintings except one. The worst results are with SIFT and normal Bayes classifier. It could explain to us that SURF is able to get more unique features than SIFT. On the other hand, k nearest neighbor is the classifier with the best results. The algorithm is robust to changes in viewing angle and in lighting.

5. References

[1] Bay, Herbert; Tuytelaars, Tinne; Gool, Van, Luc, *SURF: Speeded Up Robust Features*, European Conference on Computer Vision, 2006.

[2] Galindo, Escriche, Xavi, *Searching patterns in painting images with computer vision techniques*, Universitat Oberta de Catalunya, 2013.

[3] Juan, Luo; Gwun, Oubong, *A Comparison of SIFT, PCA-SIFT and SURF*, CSC Journals, Kuala Lumpur, Malaysia, 2009.

[4] Tuytelaars, Tinne; Mikolajczuk, Krystian, *Local invariant feature detectors: a survey*, Journal Foundations and Trends in Computer Graphics and Vision, 2008.

[5] Temmermans, Frederik; Jansen, Bart; Deklerck, Rudi; Schelkens, Peter; Cornelis, Jan, *The mobile museum guide: artwork recognition with eigenpaintings and SURF*, WIAMIS, 12th International Workshop on Image Analysis for Multimedia Interactive Services, 2011.

[6] Blessing, Alexander; Wen, Kai, *Using machine learning for identification of art paintings*, Stanford University, 2010.