Searching patterns in painting images with computer vision techniques (January 2013)

Xavi Escriche Galindo

***Abstract*— this paper presents a pattern recognition method focused on paintings images. The purpose is construct a system able to recognize authors or art styles based on common elements of his work (here called patterns). The method is based on comparing images that contain the same or similar patterns. It uses different computer vision techniques, like SIFT and SURF, to describe the patterns in descriptors, K-Means to classify and simplify these descriptors, and RANSAC to determine and detect good results. The method are good to find patterns of known images but not so good if they are not**

***Index Terms*— Art paintings, computer vision, feature description, image processing, local features, object recognition, pattern recognition.**

I. Introduction

COMPUTER vision is a branch of science and technology dedicated to extract information from digital images, in order to artificially simulate the human visual system on a machine. The purpose of the computer vision is to program computers to "understand" the features of a scene or an image. There are different kinds of studies inside computer vision like image processing, detection, pattern recognition, evaluation of results, statistical learning. In this paper, I focused on pattern recognition. Pattern recognition is the science that deals with the processes of engineering, computer science and math-related physical on abstract objects, in order to extract information for making joint properties between those objects.

I've just studied a Master's Degree in Open Source and, to finish it, I needed to make a project related with an open source technology. There was the option of working for a company or to do a research project. I chose second because I've always found interesting the scientific studies and I'd always wanted to participate on one.

Among the available research proposals, there was one over computer vision techniques in museum environments. I found it very attractive because I love art and especially painting. There were different lines of research but I wanted to focus mine on search and classification of common elements on paintings able to determine a computer the image's style, the historical period or the specific author.

There are some interesting proposals on the problem of automatic classification on artwork nowadays, but the capacity of the machine is still far from human capabilities. When we (as humans) look at a painting, we have the ability to know if this is a renaissance or impressionist painting, if it is a Van Gogh, a Dalí or a Monet. We have cognitive abilities that allow it, either by color, the elements of the work, style, or altogether. A machine, a computer is unable to do this. The question is how could this computer be able to acquire these skills? There are many possible approaches to achieve this objective. I've wanted to focus my study on "pattern recognition", understand pattern as an element within the work that distinguish and identify the authors. Some examples of patterns could be:

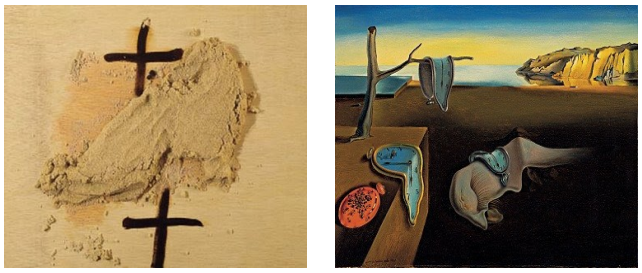

Fig. 1. On left side the pattern could be the crosses. On right side the pattern could be the clocks.

These patterns will be used by the system to classify and compare these in other images and so, be able to classify the artwork by author and style.

This paper is structured in seven points: In I), I introduce the subject of the research. In II), I expose the actual state-of-art. In III), I describe the methods and technologies that I've used. In the IV), I explain the pipeline and how the method is. In V), I explain in detail all the experiments and their results. In VI), I expose the conclusions. The last, VII) I analyze possible improvements, future works and things that it have been pending.

II. Previous work

There is an extensive literature on computer vision nowadays. The ones related to this study are in terms like **interest point detection**: Harris corner detector [1], based on the eigenvalues of the second moment matrix, was one of the most used detector method. However, Harris corners are not scale-invariant. Lindeberg [2] introduced the concept of automatic scale selection which allows to detect interest points in an image, each with their own characteristic scale. Mikolajczyk and Schmid [3] refined this method, creating robust and scale-invariant feature detectors with high repeatability. Lowe [4] proposed to approximate the Laplacian of Gaussians (LoG) by a Difference of Gaussians (DoG) filter. Related on this studies, there were another focused on **interest**

[1] This work is my final project of the University Master's Degree Programme in Free Software (UOC University).

point and feature description: SIFT [5] have been shown to outperform the others. SIFT computes a histogram of local oriented gradients around the interest point and stores the bins in a 128- dimensional vector (8 orientation bins for each of 4 × 4 location bins). There were another good methods but there was one who increases the results in a faster way: SURF [6], [7] that was inspired by SIFT descriptor. SURF is based on sums of 2D Haar wavelet responses and makes an efficient use of integral images using an integer approximation to the determinant of Hessian blob detector,

In **clustering terms**, there are so many different algorithms but I focused on K-Means. K-Means is 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. This was first used by J. MacQueen [8], inspired on the idea Hugo Steinhaus [9] but the standard algorithm was first proposed by Stuart Lloyd in 1957, but it wasn't published until 1982 [10]. In 1965, E. W. Forgy [11] published essentially the same method, which is why it is sometimes referred to as Lloyd-Forgy, too. A more efficient version was proposed and published by Hartigan and Wong [12], [13]. There are another good cluster learning techniques like Nistér and Stewénius [14] Vocabulary Trees. This is similar to the k-means but is more efficient on a large vocabularies.

I used another technique called **homography**. This is a mathematical concept based on projective transformations that determines a correspondence between two flat geometric shapes, so each point and each line of a figure, corresponding respectively to a point and a line to the other. RANSAC [15] algorithm is used to remove data from atypical set of correspondences. I read some articles [16], [17] where they use RANSAC to find homographies between the images, similar to my solution.

Finally, there are other studies in the state-of-art related to my study that although I haven't used it, I think it's good to name here: HOG [18] are also a feature descriptors technique of object detection, based on counting occurrences of gradient orientation in localized portions of an image. FERNS [19], [20] is a new methodology for recognizing images based on a simple, efficient and robust algorithm that eliminates unnecessary preprocessing using a non hierarchical structures called ferns.

III. Methodologies

To realize my project, I had to use open source tools that they could be able to process images and also, make some kind of functions on them. For this purpose I selected OpenCV that permits to make it. *OpenCV* (***O**pen **S**ource **C**omputer **V**ision*) is an open source library of programming functions for real time computer vision. It is released under a *BSD license*[1]. It is free for both academic and commercial use. Inside OpenCV are so many libraries and functions that permit me to use the next methodologies:

A. SIFT

In 2004, *David G. Lowe* published an article [1] where presents a method called *Scale-invariant feature transform* (SIFT) used to detect and describe *local features* on images. These features are invariant to rotation and scale effects between images and provide a rather large coincidence when there are substantial changes in terms of distortion, noise, or change the lighting in the picture.

This algorithm is used for many applications, including: object recognition, mapping, robotic navigation, image stitching, 3D modeling, gesture recognition, video tracking and motion tracking.

This method consists on comparing the individual features of the image with a database of features known objects using a fast nearest neighbor algorithm, followed by a Hough transform to identify groups belonging to a single object, and finally perform verification through least-squares solution.

A SIFT *feature* is a selected image region (also called *keypoint*) with an associated *descriptor*.

A SIFT *keypoint* is a circular image region with an orientation. It is described by a geometric frame of four parameters: the keypoint center coordinates *x* and y, its scale (the radius of the region), and its orientation (an angle expressed in radians).

A SIFT *descriptor* is a 3-D spatial histogram of the image gradients in characterizing the appearance of a keypoint. The gradient at each pixel is regarded as a sample of a three-dimensional elementary feature vector, formed by the pixel location and the gradient orientation. Samples are weighed by the gradient norm and accumulated in a 3-D histogram *h*, which (up to normalization and clamping) forms the SIFT descriptor of the region. An additional Gaussian weighting function is applied to give less importance to gradients farther away from the keypoint center.

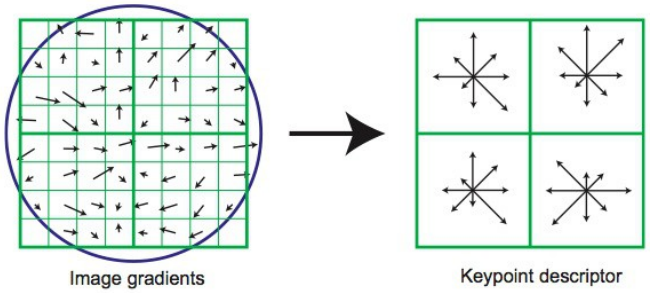

Fig. 2. shows how a keypoint descriptor is created. First (as show on left side), by computing the gradient magnitude and orientation at each image sample point in a region around the keypoint location. These are weighted by a Gaussian window, indicated by the circle. Then (as shown on the right), these samples are accumulated into orientation histograms summarizing the contents over 4x4 sub-regions with the length of each arrow corresponding to the sum of the gradient magnitudes near that direction within the region. This figure shows a 2x2 descriptor array computed from an 8x8 set of samples.

[1] http://creativecommons.org/licenses/BSD

B. SURF

In 2006, *Herbert Bay* presents an article [2] improved and revised two years later, in 2008 [3]. These articles presents another detector and robust image descriptor algorithm called *SURF: Speeded Up Robust Features*. This algorithm is used for computer vision systems such as object recognition. SURF was inspired by SIFT descriptor, although it is several times faster and according to the authors. SURF is based on sums of 2D Haar wavelet responses and makes an efficient use of integral images. As basic image features it uses a Haar wavelet approximation of the determinant of Hessian blob detector.

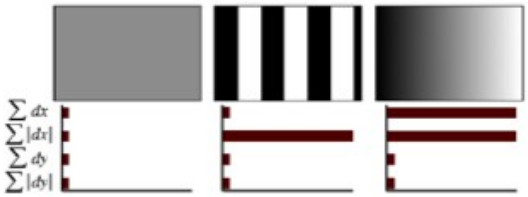

Fig. 3. shows the properties of the descriptor for three distinctively different image intensity patterns within a sub-region. The descriptor entries of a sub-region represent the nature of the underlying intensity pattern. (On left side) A case of a homogeneous region, all values are relatively low. (On left middle) In presence of frequencies in x direction, the value of ∑|dx| is high, but all others remain low. If the intensity is gradually increasing in x direction, both values ∑dx and ∑ |dx| are high.

C. K-Means

Cluster analysis or Clustering is the task of grouping a set of objects in such a way that objects in the same group (called cluster) are more similar (in some sense or another) to each other than to those in other groups.

In 1967, *James MacQueen* presents *K-means Clustering* [4]. This algorithm is a simple and easy clustering method used to minimize the mean squared Euclidean distance between objects in a given data set. This objects are classified through a certain number of clusters (assume k clusters) fixed a priori. The main idea is to define k centroids, one for each cluster. These k centroids represent the average value of the objects near every cluster.

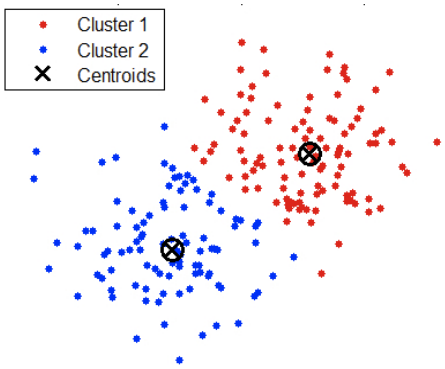

Fig. 4. shows a set of points in two colors, one for each cluster. In this case the k=2. The Centroids represents the mean value of each cluster.

D. RANSAC

In 1981, *Martin A. Fischler* and *Robert C. Bolles* presented an algorithm called *RANSAC ("RANdom SAmple Consensus")* [5]. This is an iterative robust method to estimate parameters of a model from a set of observed data which contains *outliers*. The idea is that the data consists of *inlier* (data whose distribution can be explained by some set of model parameters), and *outlier* (data that do not fit the model). Data can be subject to noise. The *outliers* can come, for example, from extreme values of the noise or from erroneous measurements or incorrect hypotheses about the interpretation of data.

It is a non-deterministic algorithm because it produces a reasonable result only with a certain probability, with this probability increasing as more iteration is allowed.

RANSAC algorithm can be applied to get the *homography* of each image pair. Homography is a concept in the mathematical science of geometry. A homography is an invertible transformation from a projective space (for example, the real projective plane) to itself that maps straight lines to straight lines. In the field of computer vision, any two images of the same planar surface in space are related by a homography.

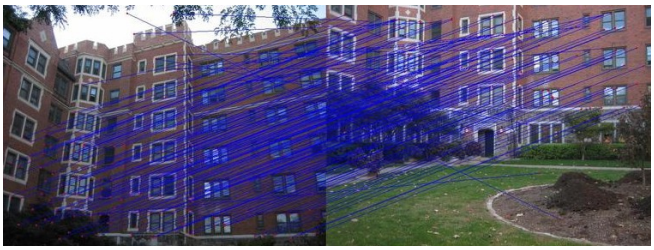

Fig. 5. shows a comparison with 181 matching pairs.

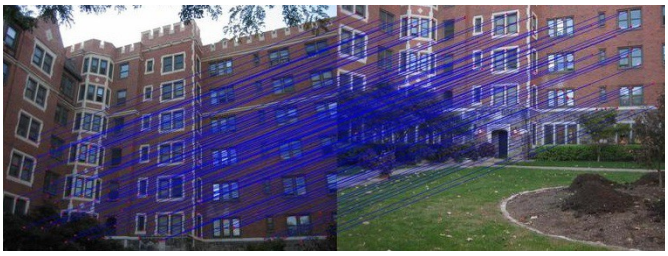

Fig. 6. shows the same image after applying RANSAC. This removes the "outliers". Now the number of matches are 139.

IV. Pipeline

Through the combined use of the methods in the previous section, I have built a system able to perform a series of experiments to help me get to recognize patterns in images. I separate the process in two phases:

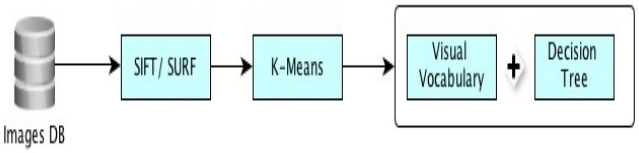

Fig. 7 shows he training phase scheme.

The Fig. 7. shows the first phase, the "training phase". In this phase the system learn from the images of the database and build a vocabulary with these images and store them in an index structure (a decision tree).

On the left, there's a database that consists in a set of images (I1, I2... In) representing the data required for the system to learn. Each image contains one or more patterns. For example:

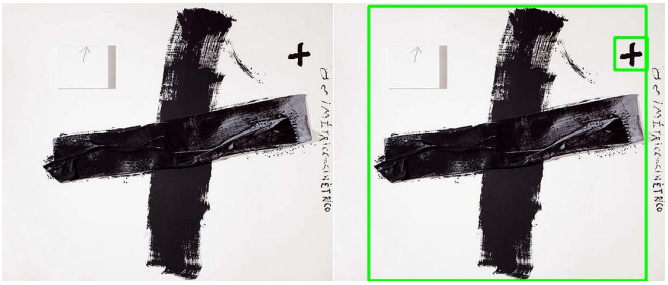

Fig. 8 On left side, we can see the image that contains two patterns (in this case, a cross). On right side, the green squares shows every pattern.

The first thing we have to do is select images that contain the pattern we want to find (in our case, the crosses). For each image, we will cut the pieces of the pattern and store those images in new files. For example:

Fig. 9. On left side, we have the original. On right side, we have the two images. The names of these will be the name of the original image -i.jpg, where "i" is an index.

Once we have selected, cut and saved all pictures, we will have our training dataset with all of our patterns. Now, we need a method to describe these patterns. There are different methods to do that but I have chosen SIFT or SURF because they are well known in the world of computer vision. These are used to detect and describe local features in images. These features will help us to describe and classify our image patterns.

First of all, the system creates the detector and descriptor objects. These can be of many types[2]. Although, in this study, I have focused only on SIFT, SURF and USURF. The last one is equivalent to SURF but it doesn't compute the orientation of each descriptor. Then, it read all the images and applies the detector to find their keypoints and the descriptor to extract their descriptors.

Once it has the descriptors of all images, it applies kmeans clustering in all the set of images descriptors. This method consist in find all the closest centers for each descriptor. This also constructs a vocabulary with all these center values. This method also extracts all the equivalences between every image descriptor and the nearest center to it. This is what we call "visual vocabulary" because describes all the images in the vocabulary. Finally, it constructs a structure (similar to a tree) with all the k centers has every image. For example:

Image Number	K-Centers
Image 1	(1, 4, 5, …, n)
Image 2	(2, 3, 5, …, m)

This structure, that we called "decision tree" will be used after to decide which images are candidate (as we saw on the next phase)

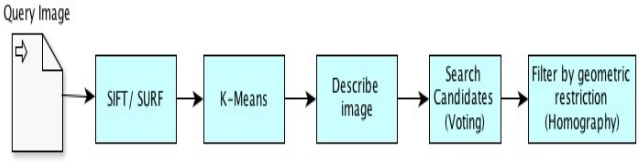

Fig. 10 shows the production phase scheme.

This consists in the "production phase" because we can do different searching test.

On the left side of Fig. 10, there's the query image. This represents the image that we want to compare with the images from their vocabulary to see if it contains one or more of the predefined patterns. The system doesn't know a priori if it contains.

To this image, the system also applies SIFT or SURF to detect his keypoints and extract his descriptors. The next that it has to do is apply a "manual kmeans". This is not as apply kmeans on the query image descriptors because we need to compare to the vocabulary descriptors. This is what we called "describe image". To do this, first, it applies a function that transforms all the query image descriptors to k-center descriptors keeping the nearest center for each descriptor. For example:

Descriptor number	Nearest K-Center number
Descriptor1	3
Descriptor2	120

The way to select the closest center is comparing the all the Euclidean distances (in each dimension and in absolute

[2] Feature Detectors, Descriptor Extractors

values) between every image descriptor and all the center descriptors. The once who has the less sum of all differences values, that is the selected center, the closest one.

Once it has this information, the next step to do is compare the query image with all the images of the vocabulary (trying to find the defined vocabulary patterns on it). To do this, it makes a voting process that consists in looking for every vocabulary image if it has the k-center descriptor of imageQuery. The result of this process is an array that contains the number of votes, or what is the same, the number of centers that has each image of the vocabulary. For example:

Image Number	Number of votes
Image 1	10 votes
Image 2	23 votes

The next step of the algorithm consists in compare every voted image discriminating those who have obtained few votes, because this means that they are not similar with the query image. The system only process those who has a number of votes >= to a defined parameter. If the image satisfies the condition, then it selects.

For every selected image, it gives his descriptors and transforms them into k-centers transforming (in the same way mentioned above for the queryImage).

Then it looks for his space points (X, Y) with the same k center in two images. Of this way, it obtains the correspondence between the points of the two images with the same k-center. It constructs two arrays of points, one for the queryImage and one for the imageSelected:

ImageSelectedX	ImageSelectedY	ImageQueryX'	ImageQueryX'
123	234	23	100
1	100	34	134

The last step is to look if these points are similar. To do that, it uses the RANSAC algorithm to find homographies between these correspondence points. The system performs some homographies and evaluates if they are good or bad. This will determine if the query image contains a pattern or not.

There are different ways to determine whether the homography is good or not. Once is calculating a *determinant* to the homography. If the value of *determinant* is closed to 1 (0.9... or 1.0...) it consider that is a good homography.

Also uses another method. This consist in obtain the four corner points, with a square form, from the selected image *(objCorners)*. Then, it performs a perspective transformation on these corners image applying the homography obtained before. The result to apply the perspective on this *objCorners* in the queryImage returns a result with another four point figure (*sceneCornersObject)*.

With these two 4-points objects it knows also if it's a good or a bad homography. If the object obtained by applying the perspective (*sceneCornersObject)* looks like a similar square form of the *objCorners* (with a margin error defined in another parameter) it has a good homography. A good result has the next form (saved in a result image):

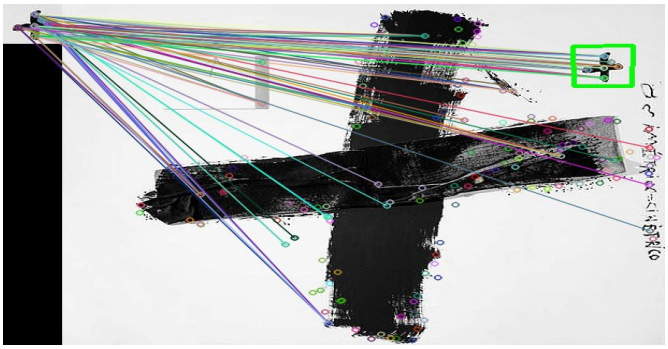

Fig. 11. Shows the matches (points correspondence) between 2 images. The green square shows how a good homography finds the cross pattern.

It's possible that the image query contains more than one pattern. That's why the system performs more homographies. Once it find a good homography, removes the points inside the green square (also called "inliers") and it remake the RANSAC algorithm with the rest of the points that has query image.

After doing the process for the entire voted images loop, the system saves another image result that it will contain all the patterns searched well:

Fig. 12. This image shows an image with two green squares. This means that the system have found the two possible patterns. This represents a 100% of success.

V. Results

I've made an application[3] in C++ language using OpenCV[4] library (both open source technologies) to develop the search pattern system. The application is made using different OpenCV modules (*core*, *highgui*, *imgproc*, *features2d and calib3d*).

This application will allow us to analyze the results qualitatively. After applying different experiments, we can quantize these results to display them in a numerical and graphical way. We obtain an error for each experiment.

The idea is to test the system with *N* images to obtain *N* errors (E1, E2... En). With these, we can calculate the mean of all the errors and obtain a final error, which will serve us to analyze and evaluate our experiments.

[3] https://github.com/kaneda75/SearchPatternsInArt
[4] http://opencv.willowgarage.com/wiki/

1) Instance identification (with the query image on dataset)

The experiment consists on compare all the images of a 10 image data set that contains one or more pattern (crosses). The query image is include on the data set. I made one test for every image, for every algorithm type (SIFT, SURF, USURF) and for a range of k centers (from 1 to the total number of descriptions in the data set). The system returns an error on every test, defined as:

Error type	Result
Found 0 patterns	0%
Found #P* patterns	1/#P (2P Image: 50% if found 1, 100% if found 2)
Found all patterns	100%

**#P is the number of patterns that has an image.*

I look if the images are correct and I write the result error for all of the tests for every image. So, I analyze these in a qualitative way but I write every value on a file. With these values, I calculate the mean of the error for all the images:

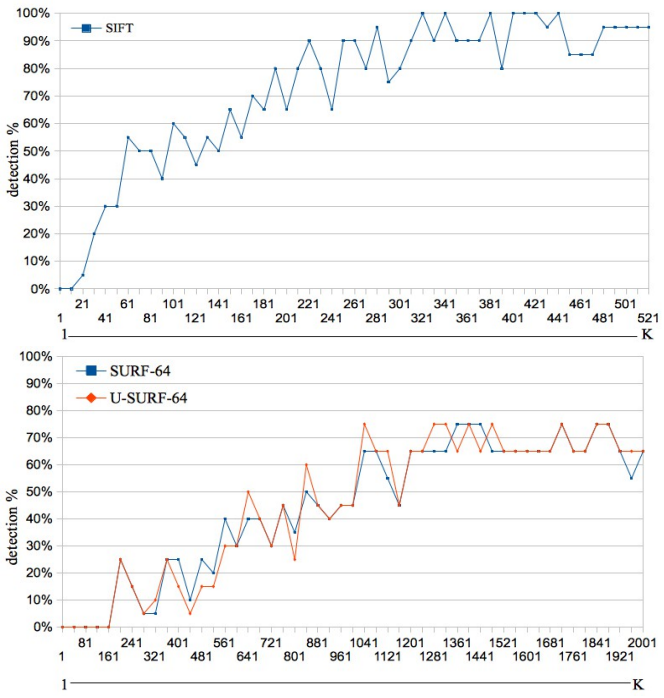

Fig. 13. Comparison of different descriptors. Top: SIFT. Bottom: SURF and U-SURF using 64 elements in each descriptor.

On Fig.13. we can see the obtained results and we can define some conclusions: SIFT, uses less descriptors (523) than SURF (2012) in this set. This thing makes SIFT faster and better to recognize patterns. The optimal k is near the number of total descriptors. This means that these algorithms function well when the images are more similar. The method is valid to find patterns of images inside the dataset.

2) Instance identification (Applying effects)

This experiment consist on apply the same methodology of experiment 1, but applying different effects on the query image. First applying a *Gaussian blur*[5] that smooth the image. The results are good in some cases if the Gaussian kernel size is little.

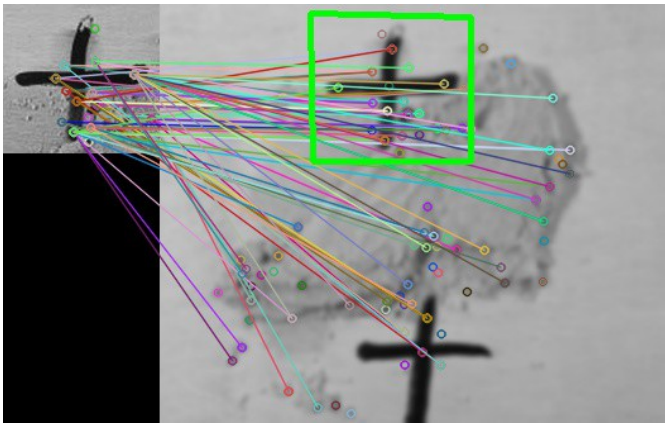

Fig. 14. Shows the match in a smooth image with gaussian kernel size =11

If the effect applied is higher (kernel size=21) the method can't match any image.

A second effect I tested is to *resize* the image query. The method doesn't work well if it resize the image to his double size (size * 2) or if it resizes to the middle (size / 2). If we apply two resizes together, first multiplying the image size * 2 and the divide the size / 2, the method works perfect, as good as experiment 1.

3) Instance identification (leave-one-out)

This experiment consist on apply the same methodology of experiment 1, but in this case, leaving out the data set, the patterns of the query image. I've tested the same proofs of experiment 1, but in this case all the results are negative. The method cannot find any pattern.

VI. Conclusion

Through these experiments, we've seen how this method works very well recognizing patterns in images when they form part of our set (our database). In a large database with many patterns classified of images of all the works of different artists, this system would be valid to identify the works and artists.

Furthermore, SIFT and SURF algorithms work very well comparing the same images, but not so well when the patterns are similar, but not equal. This is because the way that they describe images are very detailed.

[5] http://en.wikipedia.org/wiki/Gaussian_blur

VII. FUTURE WORK

In order to proceed with the study of searching pattern matching, it could be good to do different things: make other tests changing the system parameters that we tried, trying to get more and not so accurate matches. Maybe, modifying the way the patterns are described, using other methods which were not SIFT and SURF. Furthermore, tests could be made with databases that have many more patterns. In this way the system would have more information and would be easier to it to recognize patterns. Another option would be remake the way that we consider found the pattern (with the homography). Definitely researching and testing more.

ACKNOWLEDGMENT

I would particularly like to thank Xavier Baró Solé who has taught me on most of the methodologies and has suggested numerous improvements on the content and presentation of this paper. Also, I would like to thank Àgata Lapedriza Garcia who guided me at the beginning of my study.

Also, I want to thank my parents and my girlfriend Betty for all the patience and encouragement they have given me.

IMAGE CREDITS

Fig.1 "Sabata" *Antoni Tàpies* (1995) (also on Fig. 14), "The Persistence of Memory" *Salvador Dalí* (1931).
Fig.2. [5] "Distinctive image features from scale-invariant keypoints".
Fig.3. [7] "SURF: Speeded Up Robust Features".
Fig.4. http://www.mathworks.es/es/help/stats/kmeans.html.
Fig.5-6 www.cs.cmu.edu/afs/andrew/scs/cs/15-463/f07/proj4/www/mdougla1.
Fig.8,9,11,12. "Asimétric" *Antoni Tàpies* (2007).

REFERENCES

[1] C. Harris and M. Stephens. "A combined corner and edge detector". *Proceedings of the Alvey Vision Conference,* pp. 147-151, 1988.

[2] T. Lindeberg. "Feature detection with automatic scale selection". *IJCV*, *Vol. 30, No. 2*, pp. 79-116, 1998.

[3] K. Mikolajczyk and C. Schmid. "Indexing based on scale invariant interest points". *ICCV, Vol. 1*, pp. 525-531, 2001.

[4] D. G. Lowe. "Object recognition from local scale-invariant features". *ICCV,* 1999.

[5] D. G. Lowe. "Distinctive image features from scale-invariant keypoints". *International Journal of Computer Vision*, *Vol. 60, No. 2,* pp. 91-110, 2004.

[6] H. Bay, T. Tuytelaars, and L.Van Gool. "Surf: Speeded up robust features". *European Conference on Computer Vision*, *Vol. 1*, pp. 404-417, 2006.

[7] H. Bay, A. Ess, T. Tuytelaars and L. Van Gool, "SURF: Speeded Up Robust Features", *Computer Vision and Image Understanding (CVIU)*, *Vol. 110, No. 3*, pp. 346-359, 2008.

[8] J. B. MacQueen. "Some Methods for classification and Analysis of Multivariate Observations", *Proceedings of 5-Th Berkeley Symposium on Mathematical Statistics and Probability*, *Berkeley, University of California Press, Vol. 1*, pp. 281-297, 1967.

[9] H. Steinhaus. "Sur la division des corps matériels en parties", *Bulletin de l'Académie Polonaise des Sciences, Classe III, Vol. 4, No. 12*, pp. 801-804, 1956.

[10] S. P. Lloyd, (1957). "Least square quantization in PCM". *Bell Telephone Laboratories Paper*. *IEEE Transactions on Information Theory, Vol. 28, No. 2*, pp. 129-137. Published in journal much later: 1982.

[11] E.W. Forgy. "Cluster analysis of multivariate data: efficiency versus interpretability of classifications". *Biometrics, Vol. 21,* pp. 768-769, 1965.

[12] J.A. Hartigan. "Clustering algorithms". *John Wiley & Sons, Inc*, 1975.

[13] J.A. Hartigan and M. A. Wong. "Algorithm AS 136: A K-Means Clustering Algorithm". *Journal of the Royal Statistical Society, Series C (Applied Statistics), Vol. 28, No. 1,* pp. 100-108, 1979.

[14] D. Nistér and H. Stewénius. "Scalable recognition with a vocabulary tree". *CVPR, Vol. 2*, pp. 2161-2168, 2006.

[15] M. A. Fischler and R. C. Bolles. "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography*". Communications of the ACM*, *Vol. 24, No. 6,* pp. 381-395, 1981.

[16] L. Moisan, P. Moulon and P. Monasse. "Automatic Homographic Registration of a Pair of Images, with A Contrario Elimination of Outliers", *Image Processing On Line (ISSN 2105-1232)*, 2012.

[17] E. Vincent, and R. Laganire. "Detecting Planar Homographies in an Image Pair", *IEEE Symp. Image and Signal Processing and Analysis*, pp. 182-187, 2001.

[18] N. Dalal and B. Triggs. "Histograms of Oriented Gradients for Human Detection", *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2005.

[19] M. Ozuysal, P. Fua, and V. Lepetit. "Fast Keypoint Recognition in Ten Lines of Code". *Proc. CVPR,* 2007.

[20] M. Ozuysal, M. Calonder, V. Lepetit and P. Fua. "Fast Keypoint Recognition Using Random Ferns". *IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.* 32, pp. 448-461, 2010.