Near duplicate image detection

A Degree's Project

Submitted to the Faculty of the

Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona

Universitat Politècnica de Catalunya

by

Joan Ribera Mas

In partial fulfilment

of the requirements for the degree in

(*Audiovisual Systems*) ENGINEERING

Advisor: Mikołaj Leszczuk

Barcelona & Krakow, January 2015

Abstract

The project presents different approaches about how to solve the near duplicate images problem. MPEG-7 descriptors and SIFT transform have been used in order to solve it.

The results have been exposed in two different ways. The first one has been focused in the IMCOP project and more precisely on improving the results of the previous paper done in the AGH University [EGL14].

The second one have a more academic approach and the best solution for a given set of images databases have been searched.

This problem doesn't have a general solution and it is necessary to know which database we are using in order to apply the best option.

Resum

Aquest projecte presenta diverses aproximacions de com resoldre el problema de la detecció de imatges semblants. Descriptors MPEG-7 i transformada SIFT han sigut utilitzats per tal de resoldre aquest problema.

Els resultats han sigut exposats de dues maneres. La primera ha estat centrada en el projecte anomenat IMCOP, més precisament, en la millora dels resultats del article anterior realitzat a la Universitat de la AGH [EGL14].

La segona té un enfocament més acadèmic i s'ha buscat la millor solució per un conjunt de bases de dades d'imatges.

Aquest problema no té una solució general i es necessari saber amb quina base de dades estem treballant per tal de poder aplicar la millor opció.

Resumen

Este proyecto presenta diversas formas de resolver el problema de la detección de imágenes parecidas. Descriptores MPEG-7 y la transformada SIFT han sido utilizados para poder resolver este problema.

Los resultados han sido expuestos de dos maneras distintas. La primera se ha centrado en el proyecto IMCOP, más precisamente, en la mejora de los resultados del anterior artículo realizado en la Universidad de la AGH [EGL14].

La segunda manera tiene un enfoque más académico y se ha buscado la mejor solución para un conjunto de bases de datos de imágenes.

Este problema no tiene una solución general y se necesita saber con qué base de datos se está trabajando para aplicar la mejor opción.

Acknowledgements

I would like to thank my supervisor Mikołaj Leszczuk all the help that he has given to me during the project.

And I want to especially thank the help of my parents encouraging me constantly.

Revision history and approval record

Revision	Date	Purpose
0	20/12/2014	Document creation
1	25/01/2015	Document revision

DOCUMENT DISTRIBUTION LIST

Name	e-mail
Joan Ribera Mas	joan_ribera_mas@hotmail.com
Mikołaj Leszczuk	leszczuk@agh.edu.pl
Jorge Mata	jorge.mata@entel.upc.edu

Written by:		Reviewed and approved by:	
Date	20/12/2014	Date	25/01/2015
Name	Joan Ribera Mas	Name	Mikołaj Leszczuk/ Jorge Mata
Position	Project Author	Position	Project Supervisor

Table of contents

List of Figures

List of Tables

1. Introduction

Near duplicate image detection is the task of finding different versions of the same image.

For example images that are not exactly duplicates in binary form, but can be visually identified as the same image having undergone various editing steps such as color mapping, scaling, format changing, etc.

Near-duplicate image detection and sub-image retrieval is an important problem with several applications. Our system is motivated by one practical scenario: to get a lighter image database.

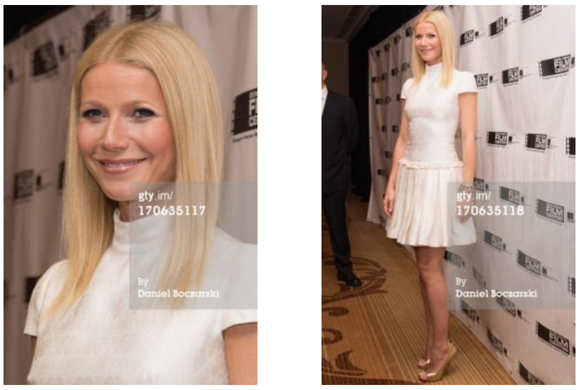


Figure 1.1: Example of near duplicate images

1.1. Statement of purpose

In this project I will discuss different ways to detect near duplicate images, a comparison between them, which one is the best and some improvements in order to have a better performance.

1.2. Requirements and specifications

The main goal of this project is to have different approaches of the near duplicate image detection and find the best implementation that improves the results previously done in the paper [EGL14] related with the IMCOP project and also find the best solution for the near duplicate image problem given different image databases.

In this work I have used MPEG-7 libraries provided by the AGH professors and also some toolboxes find it in the internet.

This project is a small piece of the IMCOP project carried out by the AGH University and the Orca Company.

IMCOP [DEP13] it's a system which improves current approaches in the field of digital preservation of IPTV and Internet Web-based content. It provides a comprehensive and extensive system for analysing, documenting and presenting dynamic Web-based content and complex, multimedia objects.

1.3. Work plan

1.3.1. Work structure

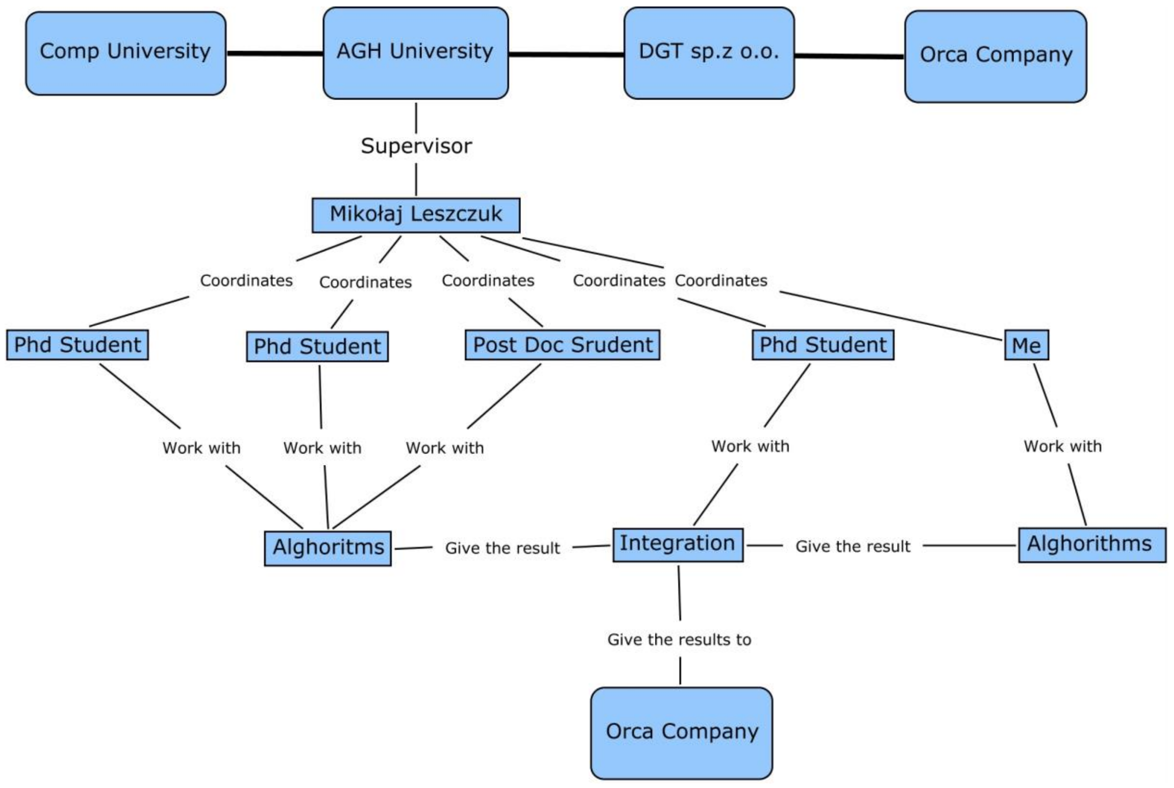


Figure 1.2: Work structure

1.3.2. Work packages

Project: Documentation about the project	WP ref: WP1
Major constituent: Investigation	Sheet 13 of 51
Short description: First contact with the project, investigation about the goals.	Planned start date: 25/09/2014 Planned end date: 08/10/2014
	Start event: 25/09/2014

	End event: 08/10/2014	
Internal task T1: Reading papers related to the IMCOP topic. Internal task T2: Searching for a useful image database in order to have a good set of testing images.	Deliverables: T1,T2	Dates: 08/10/2014

Project: Investigation about different compression algorithms	WP ref: WP2	
Major constituent: Investigation	Sheet 14 of 51	
Short description: Compare different compression techniques.(JPEG, mini JPEG, JPEG2000)	Planned start date: 09/10/2014 Planned end date: 17/10/2014	
	Start event: 09/10/2014 End event: 17/10/2014	
Internal task T1: Summary of the main differences about the different compression techniques	Deliverables: T1	Dates: 17/10/2014

Project: Investigation about MPEG-7 descriptors	WP ref: WP3	
Major constituent: Investigation	Sheet 14 of 51	
Short description: Investigate about MPEG-7 and CBIR (content-based image retrieval) techniques. And read papers about ND detection images.	Planned start date: 18/10/2014 Planned end date: 22/10/2014	
	Start event: 18/10/2014 End event: 22/10/2014	
Internal task T1: Summary	Deliverables: T1	Dates: 22/10/2014

Project: Research of matlab toolboxes	WP ref: WP4	
Major constituent: Software	Sheet 15 of 51	
Short description: Search for some matlab toolboxes used in ND detection images.	Planned start date: 23/10/2014 Planned end date: 29/10/2014	
	Start event: 23/10/2014 End event: 29/10/2014	
Internal task T1: Matlab toolboxes	Deliverables: T1	Dates: 29/10/2014

Project: MPEG-7 color descriptors implementation	WP ref: WP5	
Major constituent: Software	Sheet 15 of 51	
Short description: Implementation of the different color descriptors of MPEG-7, testing it with the databases and compare the results.	Planned start date: 29/10/2014 Planned end date: 07/12/2014	
	Start event: 29/10/2014 End event: 07/12/2014	
Internal task T1: short summary with the different descriptors and their performance and code.	Deliverables: T1	Dates: 07/12/2014

Project: Verification of the results	WP ref: WP6	
Major constituent: Software	Sheet 16 of 51	
Short description: Verification of the previous results.	Planned start date: 08/12/2014 Planned end date: 15/12/2014	
	Start event: 08/12/2014 End event: 15/12/2014	
Internal task T1: Short summary with the problematic images	Deliverables: T1	Dates: 15/12/2014

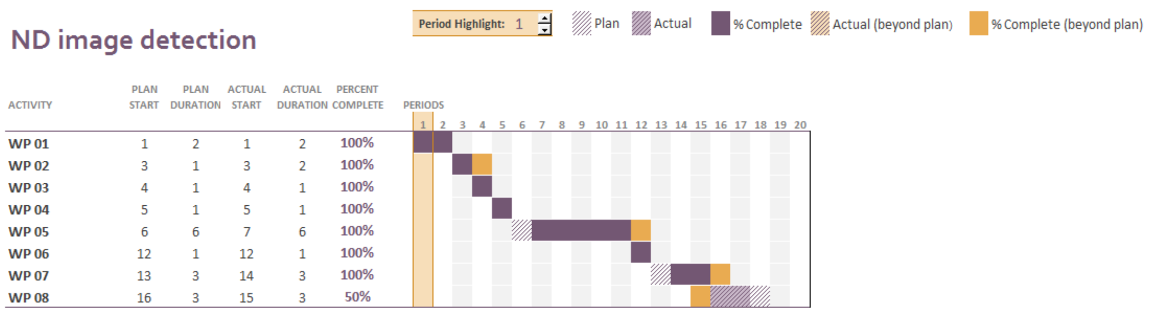
Project: Improvements of the descriptors performance	WP ref: WP7	
Major constituent: Software	Sheet 16 of 51	
Short description: Improvements of the performance of the MPEG-7 descriptors	Planned start date: 16/12/2014 Planned end date: 07/01/2015	
	Start event: 04/12/2014 End event: 07/01/2015	
Internal task T1: Code and examples.	Deliverables: T1	Dates: 07/01/2015

Project: Documentation	WP ref: WP8	
Major constituent: Documentation	Sheet 16 of 51	
Short description: Final documentation about the whole project.	Planned start date: 07/01/2015 Planned end date: 31/01/2015	
	Start event: 07/01/2015 End event: 31/01/2015	
Internal task T1: Documents	Deliverables: T1	Dates: 31/01/2015

1.3.3. Milestone

WP#	Task#	Short title	Milestone / deliverable	Date (week)
1	1	Reading papers about IMCOP	Presentation with the supervisor	08/10/2014
	2	Searching for the image database	Presentation with the supervisor	
2	1	Investigation about compression techniques	Summary	17/10/2014
3	1	MPEG-7 descriptors investigation	Summary	22/10/2014
4	1	Matlab toolboxes	Toolboxes	29/10/2014
5	1	MPEG-7 color descriptors implementation.	Summary and code. Critical review	07/12/2014
6	1	Verification of the previous results	Short summary	15/12/2014
7	1	Improvements.	Final review	07/01/2014
8	1	Documentation	Final Report	31/01/2015

1.3.4. Gantt diagram



ND image detection

Period Highlight: 1 ▨ Plan ▨ Actual ■ % Complete ▨ Actual (beyond plan) ■ % Complete (beyond plan)

ACTIVITY	PLAN START	PLAN DURATION	ACTUAL START	ACTUAL DURATION	PERCENT COMPLETE
WP 01	1	2	1	2	100%
WP 02	3	1	3	2	100%
WP 03	4	1	4	1	100%
WP 04	5	1	5	1	100%
WP 05	6	6	7	6	100%
WP 06	12	1	12	1	100%
WP 07	13	3	14	3	100%
WP 08	16	3	15	3	50%

Figure 1.3: Gantt diagram

1.3.5. Incidences

My first assigned task in the project framework was about trying to implement the optimal compressor for each image.

While I was working on that, I found that the technology that JPEGmini offers is almost the same that the one that our project needs.

JPEGmini [JPM11] is an image optimization technology that reduces the size of JPEG's by up to 5 or 6 times their original size, while maintaining resolution and perceived quality.

The main idea of the JPEGmini is that it imitates the qualities of the Human Visual System (HVS) very closely. This enables the system to apply the optimal amount of compression to each photo.

We needed to verify that all the information was true and the performance of the JPEGmini would suit up our project so I decided to test it in my own photos.

Figure 1.4: Image before compressor: 1,16 MB

Figure 1.5: Image after compression: 834 KB

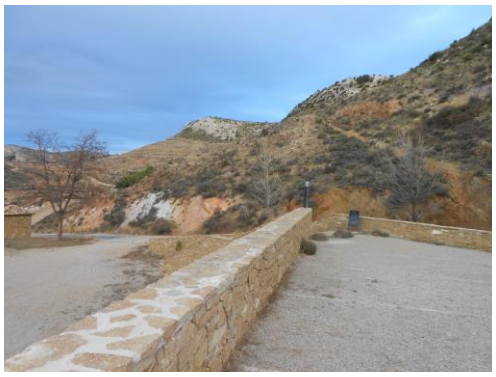


Figure 1.6: Image before compressor: 3,34 MB

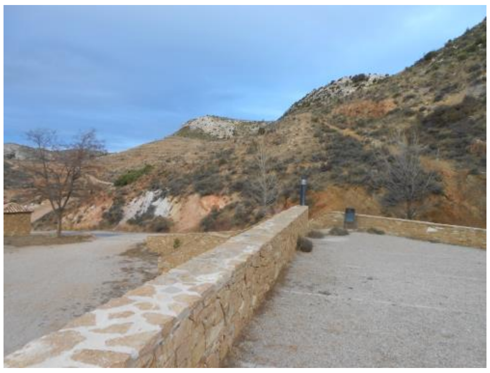


Figure 1.7: Image after compression: 1,69 MB

As we can see at the pictures any differences are notice between the two images and the rate of compression is quite big for the image size. If we use images of 5 or 6 MB we will have more compression.

Observing those results we decide to use JPEGmini [JP15] as our project compression technique.

So we decided to change the topic of my research to the near duplicate detection images problem.

2. State of the art

With the fast development of the Internet, and the availability of image capturing devices such as digital cameras, image scanners, the size of digital image collections is increasing rapidly. Efficient image retrieving, browsing and retrieval tools are required by users from various domains, including remote sensing, fashion, crime prevention, publishing, medicine, architecture and so on.

For this reason, near duplicate image detection had become one of the most important problems in image retrieval.

As more images are published on the Web, and as image manipulation software becomes more powerful and user-friendly, pirating images is becoming increasingly easy. Although digital watermarking techniques exist, these schemes are very difficult to design and there is an inherent trade-off between the robustness of the watermark and the amount of degradation induced in the image. To circumvent digital watermarking, the pirated images are often altered slightly — for instance, by cropping and rescaling.

A background, comprehensive review is required. This is known as the review of literature and should include relevant, recent research that has been done on the subject matter.

2.1. Review of Literature

These are brief summaries of previous work done in this area:

- **MyFinder** [YZC09]**:** MyFinder is a user interface application for near duplicate image detection.
  For each image they first detect a set of interest points using DOG and then they extract the LDP feature for each detected image.
  They quantize each LDP feature and construct the database with LSH hashing algorithms.
  Once they have the database constructed for each query they apply the same process of the database and then they use L2 distance and RANSAC geometric verification in order to minimize false matches.

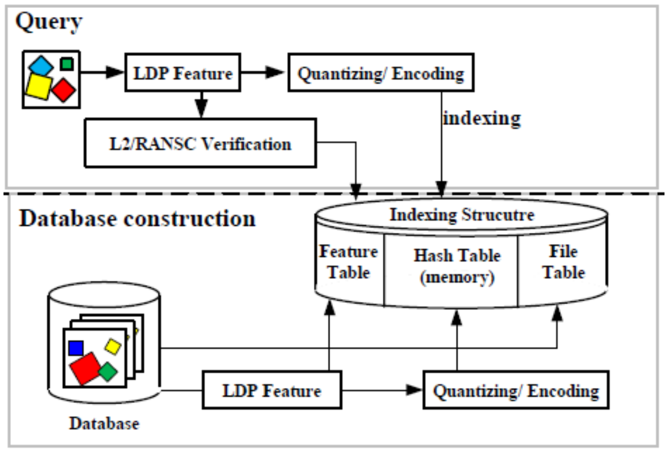


Figure 2.1: MyFinder algorithm scheme

- **Near Duplicate Image Detecting based on Bag of Visual Word Model** [LF13]**:** In this work instead of using LDP feature extraction they use SIFT transform using the visual word model to represent the feature descriptor for each image.

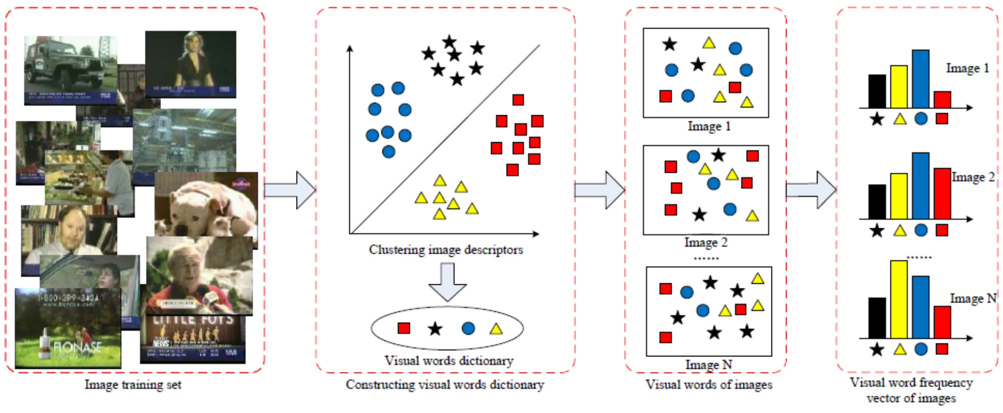


Figure 2.2: Visual word model

They construct the database with the LSH hash tables and then they compute the distance between two histograms.

- **Near Duplicate Image Detection: min-Hash and tf-idf Weighting** [CPZ08]**:**

  They propose a method that uses a visual vocabulary of vector quantized local feature descriptors (SIFT) and for retrieval they exploit enhanced min-Hash techniques.

  They use standard min-Hash as a similarity measure.

2.2. Our work

In our work we use MPEG-7 descriptors and SIFT transform in order to extract the features of our images. We compute different distances depending on which method we are using.

3. Color Descriptors

For each descriptor, the AGH work team implemented a library for the MPEG-7 standard with the methodology explained at [MSS02].

3.1. Introduction

Color is an important visual attribute for both human vision and computer processing.

This chapter provides an overview of MPEG-7 color descriptors.

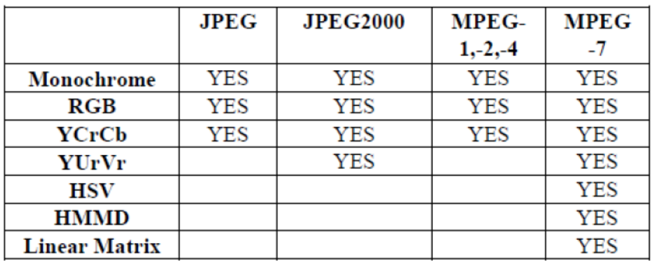
Various factors influenced the selection of these color descriptors. These include:

1. Their ability to characterize the perceptual color similarity, judged by performance of the descriptors in matching images.
2. Low complexity of the associated extraction and matching techniques, as MPEG-7 systems must be able to handle search and retrieval task over large multimedia databases or may be small, portable devices with limited computational power.
3. The size of the coded descriptions, which play an important role in indexing and in transmission of the descriptors over bandwidth-limited networks.
4. The scalability and interoperability of the descriptors.

3.2. Color spaces

The color space [RJ1] is used to specify the color space that a given descriptor refers to. It defines four color spaces: RGB, YCbCr, HVS and HMMD.

The HMMD is used only in the color structure descriptor. This provides an interoperability between various color descriptors. The table given below shows the color spaces supported by coding standards.



	JPEG	JPEG2000	MPEG-1,-2,-4	MPEG-7
Monochrome	YES	YES	YES	YES
RGB	YES	YES	YES	YES
YCrCb	YES	YES	YES	YES
YUrVr		YES		YES
HSV				YES
HMMD				YES
Linear Matrix				YES

Figure 3.1: Color spaces supported by standards

The Monochrome color space uses only the luma component (Y) of the YCbCr color space.

The explanation of each color space is in the appendices part.

3.3. Dominant Color Descriptor

The Dominant Color Descriptor (DCD) is defined as:

$$F = \{(c_{i,}p_i, v_i), s\}, \qquad (i = 1,2, \dots, N)$$

This descriptor gives a description of the representative colors of an image/image region. It consists of the number of dominant colors (N), and a vector of color components (c_i) for each dominant color as well as the percentage of pixels (p_i) in the image/image region in the cluster corresponding to c_i. A more precise characterization of the color distribution can be obtained with color variance v_i (describes the variance of color of the pixels in a cluster around the corresponding representative color) and the spatial coherence s (describes the spatial distribution of pixels associated with each representative color wherein high value would indicate that pixels of similar color are co-located). The main application of this descriptor is similarity retrieval in image databases and browsing of image databases based on single or several color values.

3.3.1. Extraction

The extraction procedure for the dominant color descriptor uses Generalized Lloyd Algorithm to cluster the pixel color values.

The distortion D_i in the ith cluster is given as:

$$D_i = \sum_n h(n) \, ||x(n) - c_i||^2, \quad x(n) \in C_i$$

Where c_i is the centroid of the cluster C_i, $x(n)$ is the color vector at pixel n and $h(n)$

is the perceptual weight for pixel n. The perceptual weights are calculated from local pixel statics to account for the fact that human visual perception is more sensitive to light changes in smooth regions than in texture regions. The update rule for the above distortion metric can be derived to be:

$$c_i = \frac{\sum h(n)x(n)}{\sum h(n)}, \qquad x(n) \in C_i$$

3.3.2. Similarity Matching

Considering two DCDs,

$$F_1 = \{(c_{1i}, p_{1i}, v_{1i}), s_{1i}\}, \quad (i = 1,2, \dots, N_1)$$

$$F_2 = \{(c_{2i}, p_{2i}, v_{2i}), s_{2i}\}, \quad (i = 1,2,\ldots,N_2)$$

Ignoring the optional variance parameter and the spatial coherence, the dissimilarity $D(F_1, F_2)$ between the two descriptors can be computed as:

$$D^2(F_1,F_2) = \sum_{i=1}^{N_1} p_{1i}^2 + \sum_{j=1}^{N_2} p_{2j}^2 - \sum_{i=1}^{N_1}\sum_{j=1}^{N_2} 2a_{1i,2j}\, p_{1i}p_{2j}$$

Where the subscripts 1 and 2 in all variables stands for descriptions F_1 and F_2, respectively, and $a_{k,l}$ is the similarity coefficient between two colors c_k and c_l (more information at [MSS02]).

One variation of the above distance is to use the spatial coherence filed. That is the distance that we use to compute the metrics between two descriptors. Is the following one:

$$D_s = w_1 abs(s_1 - s_2)D + w_2 D$$

Where s_1 and s_2 are the spatial coherencies of the query and target descriptors and w_1 and w_2 are fixed weights, with recommended settings to 0.3 and 0.7 respectively.

3.4. Scalable Color Descriptor

The Scalable Color Descriptor (SCD) is a histogram derived descriptor and can provide the global color features when measured over an entire image. It is encoded by the Haar transform and uses the HSV color space uniformly quantized to 255 bins. The figure below shows the respective color distributions in a color histogram. Based on the color distribution the two left images would be considered as more similar compared to the one on the right. In contrast to this the DCT comes with a much more compact representation but with the expense of lower performance in certain applications.

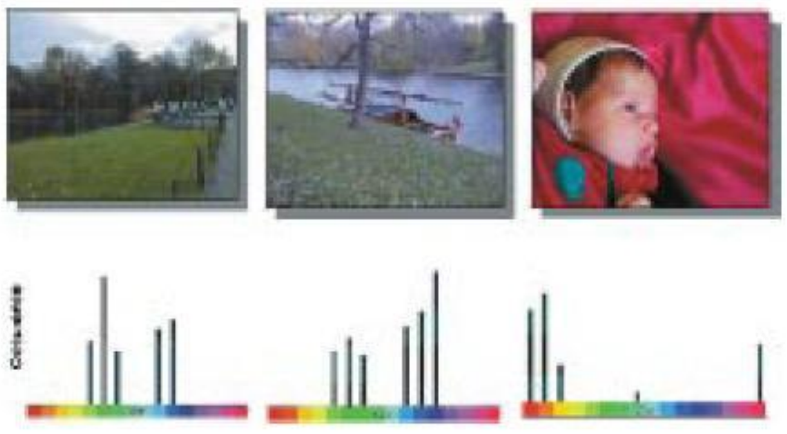


Figure 3.2: Color distribution in a color image

3.4.1. Extraction

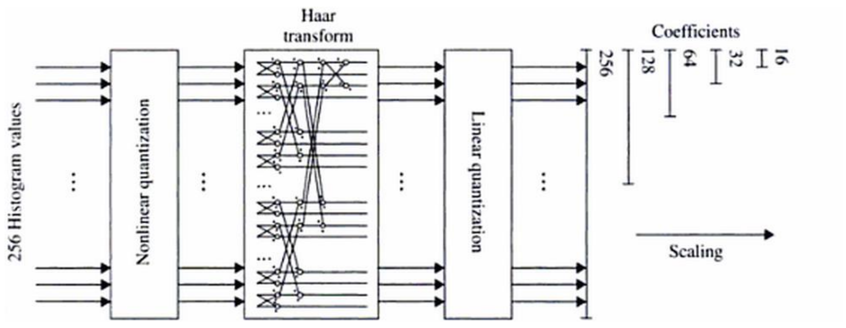


Figure 3.3: Schematic diagram of SCD

Figure 3.3 shows the block diagram of the SCD extraction process. The output representation is scalable in terms of number of bins, by varying the number of coefficients used. Interoperability between different resolution levels is retained because of the scaling property of the Haar transform. Thus matching based on the information from subsets of coefficients guarantees an approximation of the similarity in full resolution. Furthermore, the feature extraction operation can be scaled to lower levels(less bins in the source histogram).

3.4.2. Matching

l_1-norm based matching (sum of absolute differences) can be applied in the Haar transform domain; however, results are not identical with l_1-norm based matching in the histogram domain.

We use the second method in order to calculate distances. In this case in which only the sign bit is used the l_1-norm degenerates to a Hamming distance, allowing very low complexity in the distance calculation.

3.5. <u>Color Structure Descriptor</u>

The Color Structure Descriptor (CSD) is the generalization of the color histogram that captures some spatial characteristics of the color distribution in an image. It is defined in the HMMD color space using non uniform quantization, specific to Color Structure, between 32-256 colors.

A *structuring element* is defined and moved across the entire image one or more pixels at a time. At each position the color of each pixel covered by the structuring element is determined. For each color the histogram bin containing its count is incremented. The actual descriptor is obtained by normalization and non linear quantization of the final histogram.

The main application envisaged for it is image/video retrieval in multimedia databases, particularly when high accuracy is required.

The extra spatial information makes the descriptor sensitive to certain image features to which an ordinary color histogram is blind. For example, the results of applying this descriptor for the images in figure 3.4 will be different while the results obtained by the color histogram would be the same.

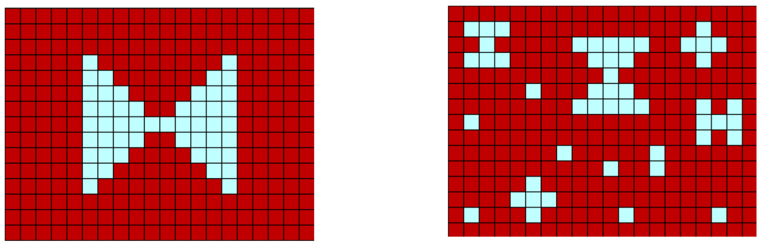


Figure 3.4: Two images with different local spatial structure of the color but with the same color histogram

3.5.1. Extraction

The CSD is best understood in terms of the Color Structure Histogram, h_s, upon which $\bar{h}_s$ is based. Extraction of a CSD is a three-step process:

1. A 256-bin CS Histogram is extracted from an image represented in the 256 cell-quantized HMMD color space. If the image is in another color space, then it must be converted to HMMD and requantized prior to extraction.
2. If $N < 256$ is desires, then bins are unified to obtain N-bin CS Histogram.
3. The values (amplitudes) of each N bins are nonlinearly quantized in accordance with the statics of the color occurrence in typical consumer imagery.

3.5.2. Matching

The matching method is the same as the one used in the structure color descriptor.

We use l_1-norm based matching in order to compute the distance between histograms.

3.6. Color Layout Descriptor

The Color Layout Descriptor (CLD) is a very compact and resolution invariant representation of color for high speed image retrieval. It captures the spatial layout of the representative colors on a grid superimposed on a region or image based on the Discrete Cosine Transform (DCT). It is expressed in the YCbCr color space. The size of the array is fixed to 8x8 elements to ensure scale invariance.

It can be used for fast searching of databases as well as filtering in broadcasting applications.

3.6.1. Extraction

The extraction process of the descriptor from an image consists of four stages: image partitioning, representative color detection, DCT transformation, and nonlinear quantization of the zigzag-scanned coefficients (see Figure 3.5). In the first stage, the image is divided into 64 blocks (8 blocks x 8 blocks). Since the sizes of the input image are not necessarily multiple of 8, it is assumed that the blocks can differ in their size, although the pixels are distributed in the most uniform way. In the following stage, a single representative color is selected from each block. Consequently, a tiny image representation of size 8x8 is obtained. Any method to compute each representative color can be applied, but the average of the pixel colors in a block is sufficient in general. In the third stage, each of the three color components is transformed by a 8x8 DCT, so three sets of 64 DCT coefficients are obtained. It is recommended by the standard to use the YUV color space. In the last stage, each set of coefficients is zigzag-scanned and a few low-frequency coefficients are nonlinearly quantized. In the MPEG-7 standard, it is recommended to use a total of 12 coefficients, 6 for luminance and 3 for each chrominance.

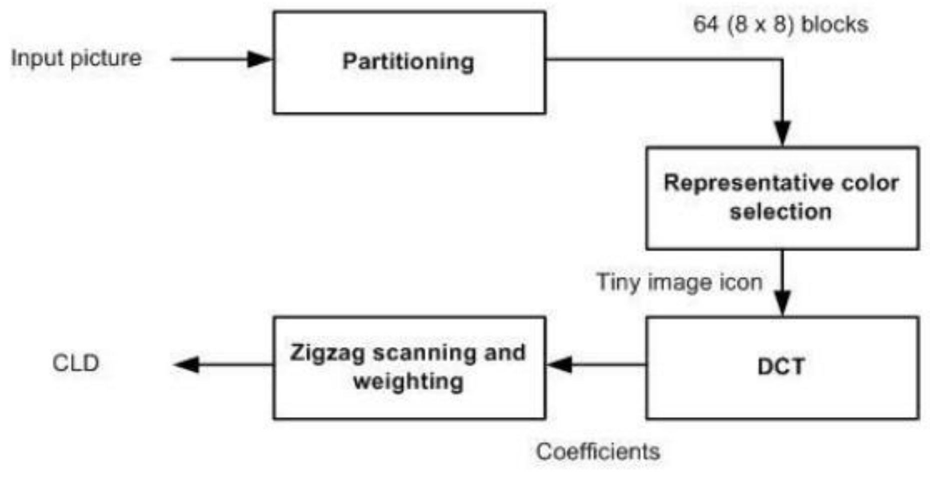


Figure 3.5: Extraction process of Color Layout Descriptor

3.6.2. Matching

For matching two CLD's, $\{DY, DC_{r,} DC_b\}$ and $\{DY', DC_r{}', DC_b{}'\}$

$$D = \sqrt{\sum_i w_{yi}(DY_i - DY_i{}')^2} + \sqrt{\sum_i w_{bi}(DCb_i - DCb_i{}')^2} + \sqrt{\sum_i w_{ri}(DCr_i - DCr_i{}')^2}$$

Here, the subscript i represents the zigzag-scanning order of the coefficients. The perceptual characteristic of the human vision system could be included for similarity calculation since the feature description is in frequency domain. The distances should be weighted appropriately, with larger weights given the lower frequency components, to match the characteristic. Since the complexity of the similarity matching process shown above is low, super high-speed image matching can be achieved.

4. Sift Transform

Scale Invariant Feature Transform (SIFT) is an image descriptor for image-based matching and recognition.

4.1. Introduction

The SIFT features are invariant to image scaling and rotation, and partially invariant to change in illumination and 3D camera viewpoint. They are well localized in both the spatial and frequency domains, reducing the probability of disruption by occlusion, clutter, or noise. Large numbers of features can be extracted from typical images with efficient algorithms.

Applications include object recognition, robotic mapping and navigation, image stitching, 3D modelling, gesture recognition, video tracking, individual identification of wildlife and match moving.

4.2. Extraction

The cost of extracting these features is minimized by taking a cascade filtering approach, in which the more expensive operations are applied only at locations that pass an initial test [L04].

Following are the major stages of computation used to generate the set of image features:

1. **Scale-space extrema detection:** The first stage of computation searches over all scales and image locations. It is implemented efficiently by using a difference-of-Gaussian function to identify potential interest points that are invariant to scale and orientation.
2. **Keypoint localization:** At each candidate location, a detailed model is fit to determine location and scale. Keypoints are selected based on measures of their stability.
3. **Orientation assignment:** One or more orientations are assigned to each keypoint location based on local image gradient directions. All future operations are performed on image data that has been transformed relative to the assigned orientation, scale, and location for each feature, thereby providing invariance to these transformations.
4. **Keypoint descriptor:** The local image gradients are measured at the selected scale in the region around each keypoint. These are transformed into a representation that allows for significant levels of local shape distortion and change in illumination.

An important aspect of SIFT extraction is that it generates a large numbers of features that densely cover the image over the full range of scales and locations. A typical image of size 500x500 pixels will give rise to about 2000 stable features.

4.3. Matching

For image matching and recognition, SIFT features are first extracted from a set of reference images and stored in a database. A new image is matched by individually comparing each feature from the new image to this previous database and finding candidate matching features based on Euclidean distance of their feature vectors.

The keypoint descriptors are highly distinctive, which allows a single feature to find its correct match with good probability in a large database of features. However, in a cluttered image, many features from the background will not have any correct match in the database, giving rise to many false matches in addition to the correct ones. The correct matches can be filtered from the full set of matches by identifying subsets of keypoints that agree on the object and its location, scale, and orientation in the new image. The probability that several features will agree on these parameters by chance is much lower than the probability that any individual feature match will be in error. The determination of these consistent clusters can be performed rapidly by using an efficient hash table implementation of the generalized Hough transform.

Each cluster of 3 or more features that agree on an object and its pose is then subject to further detailed verification. First, a least-squared estimate is made for an affine approximation to the object pose. Any other image features consistent with this pose are identified, and outliers are discarded. Finally, a detailed computation is made of the probability that a particular set of features indicates the presence of an object, given the accuracy of fit and number of probable false matches. Object matches that pass all these tests can be identified as correct with high confidence.

5. Results

In order to compute the results with each descriptor we use tree different databases (see appendix databases), one for training and two for testing. The training database is based on the Red Carpet photos and is the same used in the paper [EGL14]. The database contains sets of near duplicate images in order to evaluate the performance of our descriptors (see the images in the appendix).

The first test set is also Red Carpet topic and is a random pick up of different photos about models and actors in the Red Carpet. This set also contains near duplicate images.

The second test set is a database that I used in previous projects in the University and is from the University of Kentucky [NS06]. The image database contain 2000 images but I only pick up 44. The database is divided in 11 sets of 4 near duplicate images per set. All the image can be found in the database appendices.

5.1. Results with CSD

Using the distance previous described we decide to put a threshold. If the distance between two images is lower than that value we choose them as near duplicate images.

This are the results of the training and the test sets (we also test the training set once we had the threshold set it):

Database	Method	Set	TP	FP	TN	FN	Precision	Sensitivity	Specificity	F-measure
Red carpet	CSD	Training	90	16	1330	8	0.85	0.91	0.99	0.88
Red carpet	CSD	Test set 1	70	10	1681	6	0.87	0.94	0.99	0.9
Kentucky	CSD	Test set 2	106	0	1758	70	1	0.6	0.99	0.75

Table 5.1: Results with CSD

The algorithm performance for the red carpet databases are really good, we achieve good precision and sensitivity.

The problem here is the performance of the Kentucky database. The good results with the precision is due to the nature of the database because the differences between each set are quite big so the algorithm doesn't have problem to detect it.

The interesting thing about this dataset is that each group of near duplicate images have a lot of rotation changes between them and we can see the lack of robustness that our algorithm presents to this feature. Also the problem that we can notice here is that the threshold set in the previous sets is not enough to detect images of the same set so we obtain a lot of false negative detections.

Perhaps we could improve our performance if we would train our algorithm with a dataset with the same characteristics as the Kentucky database.

5.2. Results with the SCD

The same as the previous descriptor we set a threshold and we decide if an image is near duplicate or not.

This are the results:

Database	Method	Set	TP	FP	TN	FN	Precision	Sensitivity	Specificity	F-measure
Red carpet	SCD	Training	68	26	1324	26	0.72	0.72	0.98	0.72
Red carpet	SCD	Test set 1	51	35	1579	16	0.59	0.76	0.98	0.66
Kentucky	SCD	Test set 2	84	9	1751	92	0.92	0.48	0.95	0.62

Table 5.2: Results with SCD

As we can see the results with the scalable descriptor are worse than the ones with the CSD. This is due to the way that the descriptor analyse each image. In the SCD the image histograms at the HSV color space are taking into account so in a dataset like red carpet that we have a common background is very easy to obtain false positives between images of different sets.

Also in the Kentucky dataset the results are not satisfactory and we obtain a lot of false negative samples.

5.3. Results with the CLD

As the other descriptors, we compute the distance as it is described before.

In this case, the distance need some weights so we give to the Y (luminance) the weight of 0.15 and 0.5 for the two chrominance.

This are the results:

Database	Method	Set	TP	FP	TN	FN	Precision	Sensitivity	Specificity	F-measure
Red carpet	CLD	Training	64	4	1342	34	0.94	0.65	0.97	0.77
Red carpet	CLD	Test set 1	49	14	1604	24	0.77	0.67	0.98	0.72
Kentucky	CLD	Test set 2	92	12	1748	84	0.88	0.52	0.95	0.65

Table 5.3: Results with CLD

The results with the CLD are slightly better that the ones that we obtained with the SCD but still worse than the ones with the CSD.

The main problem with the CLD is setting the threshold in the training set because the distances between images of different sets are quite close so it is difficult to find the best threshold for the optimal model.

Also the problem with the CLD is really similar at the one that we had with the SCD because it detects a lot of images as false positive and it doesn’t model well the differences between the images in the same set with images with other sets.

5.4. Results with DCD

The results with the DCD are the following ones:

Database	Method	Set	TP	FP	TN	FN	Precision	Sensitivity	Specificity	F-measure
Red carpet	DCD	Training	61	14	1333	36	0.81	0.63	0.97	0.7
Red carpet	DCD	Test set 1	53	19	1588	21	0.73	0.71	0.98	0.72
Kentucky	DCD	Test set 2	75	28	1559	102	0.72	0.42	0.93	0.53

Table 5.4: Results with DCD

As the results with the SCD or the CLD the DCD descriptor have a lot of problems with images that have a lot of one color in common.

The problem that we have when we are working with the DCD and our databases is that numerous images of our databases have a similar background color so when we extract the descriptor from different images but similar background the system detect them as near duplicate.

5.5. Results with SIFT

Using the MATLAB toolbox described here [VLF07] we were able to create an algorithm that first of all index all the images in a database (SIFT transform of each image) and then matches one image with all the other images in the dataset by the L2 norm between each keypoint or descriptor of the image.

Also with this toolbox we were able to set a threshold for the matching process which could tell us the uniqueness of one point.

We set the threshold to the default number and we assumed that 89 or more scores between 2 images were enough to say that those images are near duplicate.

Here are the results:

Database	Method	Set	TP	FP	TN	FN	Precision	Sensitivity	Specificity	F-measure
Red carpet	SIFT	Training	92	2	1344	6	0.97	0.93	0.99	0.94
Red carpet	SIFT	Test set 1	65	64	1542	10	0.5	0.86	0.99	0.63
Kentucky	SIFT	Test set 2	152	6	1754	24	0.96	0.86	0.98	0.91

Table 5.5: Results with SIFT

As we can observe the result with the SIFT transform are really good in the Kentucky database. The SIFT transform is able to identify the images of each set and we don't have a lot of error.

On the other hand, SIFT has a lot of problems with the Red Carpet database. This is due to the lack of robustness at modelling similar backgrounds. Different images have a lot of similar keypoints so the algorithm detect them as near duplicate. So the SIFT descriptor is a good technique but with our goals is not the best algorithm.

6. Results with improvements

Once we obtained the results with all the descriptors we decided to try to improve the results for the descriptor with the best results, in our case the CSD descriptor.

6.1. Improvements

We want to improve the performance for both testing sets.

First of all, observing the second testing set, the Kentucky database, we have applied the following improvements taking into account the database characteristics:

- Comparison the image with the original and the image rotated 90, 180 and 270 degrees. We have maintained the image size [OC14].

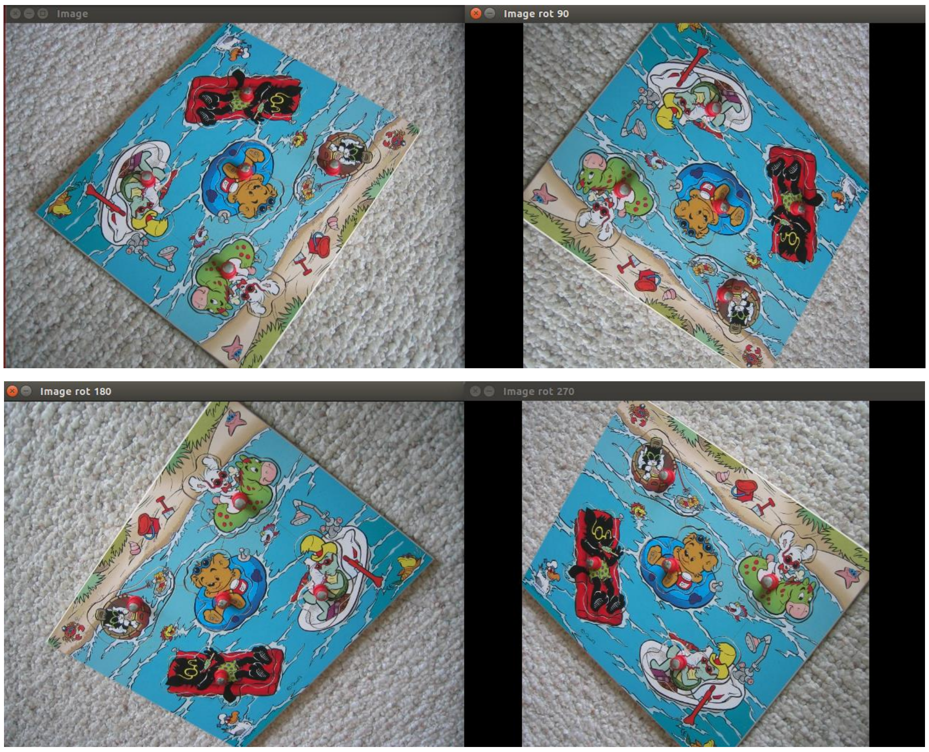


Figure 6.1: Illustration of the image rotation for an image. Image sizes: 640x480 pixels

- Comparison the image with the original, and the image rotated 90, 180 and 270 degrees applying a replacing pixels technique in the image borders. We have maintained the image size [SE10].

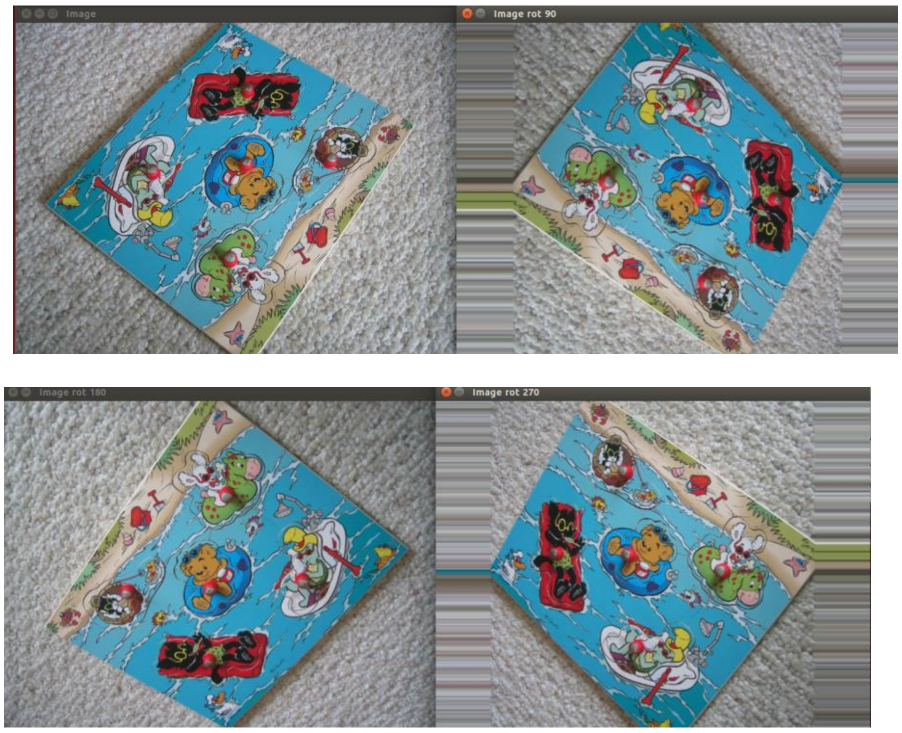


Figure 6.2: Illustration of image rotation with replacing pixels for an image. Image sizes: 640x480 pixels

- Comparison the image with the original and the image transpose and flip 90, 180 and 270 degrees. We haven't maintained the image size. This method is the same as applying a manual rotation at the image visor.

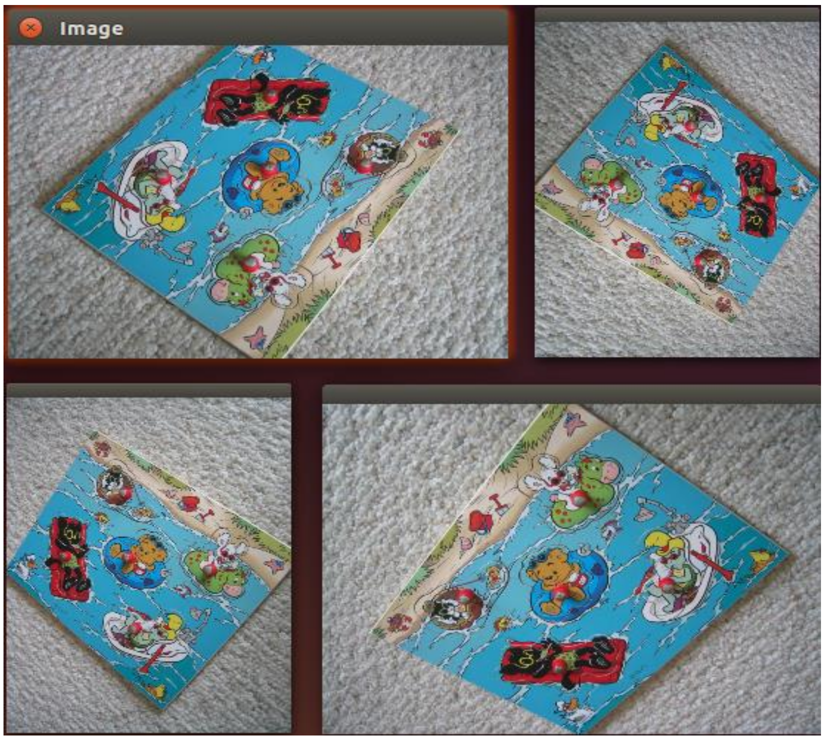


Figure 6.3: Illustration of image flip + transpose

Image size: 640x480(original and 180 degrees), 480x640(90 and 270 degrees)

- Comparison the image with the original and the image flipped vertically, horizontally and both ways. We have maintained the image size.

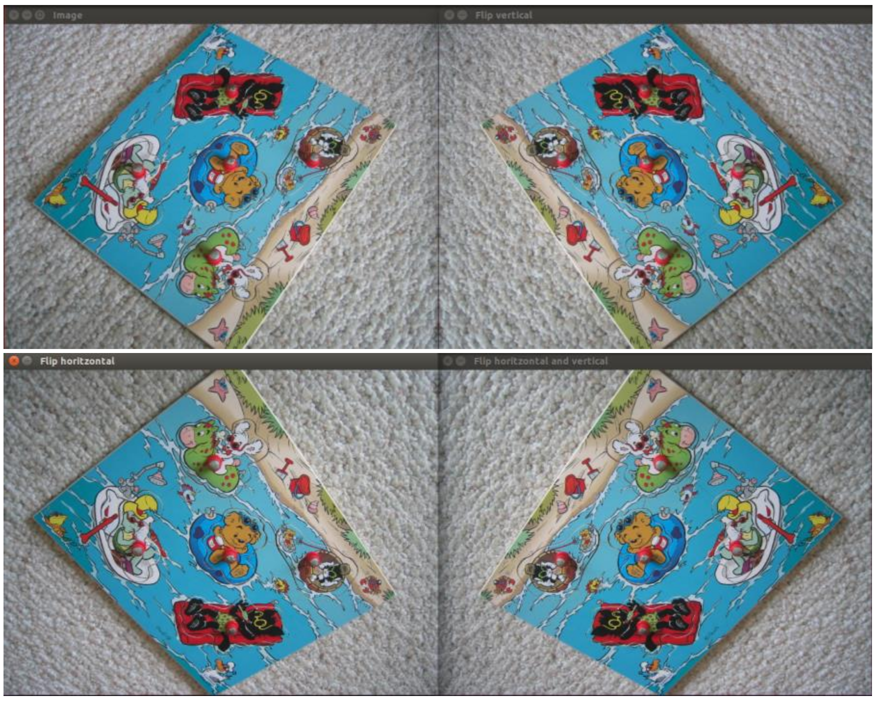


Figure 6.4: Illustration of image flipping. Image size: 640x480

Observing the Red Carpet database we figure it out that we could use image cropping in order to detect images from the same set that we previously didn't detect. We apply the following cropping [CH14]:

- Method one: Comparison the image with the original and the image cropped the upper part and the image cropped from the bottom part. The cropping is done in a way that we divide the image by two parts overlapping the middle part in order to have more relevant information.

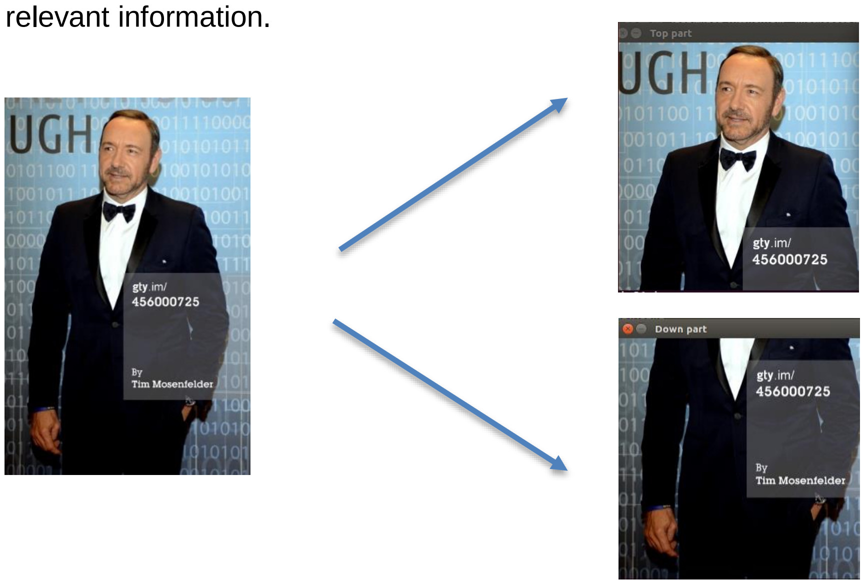


Figure 6.5: Illustration of image cropping top and down part

- Method two: Comparison the image with the original and the image with the four quarters with relevant information in each quarter. The cropping is done this way because if we divide the image exactly in four quarters the results are worse than if we overlap them with the relevant information of each quarter.

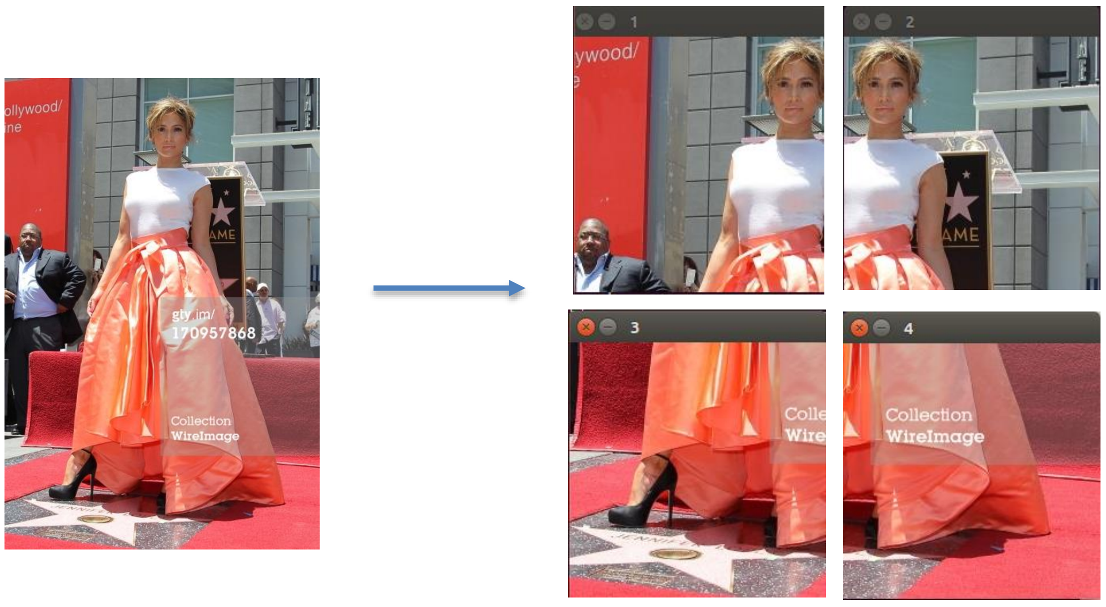


Figure 6.6: Illustration of image cropping in quarters

The last improvement technique that we apply is to combine the previous described methods.

6.2. Results

Database	Method	Set	TP	FP	TN	FN	Precision	Sensitivity	Specificity	F-measure
Kentucky	Rot (black margins)	Test set 2	120	0	1758	56	1	0.68	0.99	0.81
Kentucky	Rot (stretched edges)	Test set 2	122	0	1758	54	1	0.69	0.99	0.82
Kentucky	Rot (without maintaining size)	Test set 2	112	0	1758	64	1	0.63	0.99	0.77
Kentucky	Flipping	Test set 2	112	0	1758	64	1	0.63	0.99	0.77
Kentucky	Cropping method 1	Test set 2	110	0	1758	66	1	0.62	0.99	0.77
Kentucky	Cropping method 2	Test set 2	106	0	1758	70	1	0.6	0.99	0.75

Kentucky	Rot + Flip	Test set 2	124	0	1758	52	1	0.7	0.99	0.82
Kentucky	Rot + Crop	Test set 2	114	0	1758	62	1	0.64	0.99	0.79
Kentucky	Crop + Flip	Test set 2	114	0	1758	62	1	0.64	0.99	0.79

Table 6.1: Results with the Kentucky database

Database	**Method**	**Set**	**TP**	**FP**	**TN**	**FN**	**Precision**	**Sensitivity**	**Specificity**	**F-measure**
Red Carpet	Rot(black margins)	Test set 1	70	20	1681	6	0.77	0.92	0.99	0.85
Red Carpet	Rot(stretched edges)	Test set 1	70	18	1681	6	0.8	0.92	0.99	0.86
Red Carpet	Rot(without maintaining size)	Test set 1	70	10	1681	6	0.87	0.92	0.99	0.9
Red Carpet	Flipping	Test set 1	70	10	1681	6	0.87	0.92	0.99	0.9
Red Carpet	Cropping method 1	Test set 1	74	16	1681	2	0.82	0.97	0.99	0.89
Red Carpet	Cropping method 2	Test set 1	70	22	1681	6	0.76	0.92	0.99	0.83
Red Carpet	Rot + Flip	Test set 1	70	18	1681	6	0.8	0.92	0.99	0.86
Red Carpet	Rot + Crop	Test set 1	74	24	1681	2	0.75	0.97	0.99	0.85
Red Carpet	Crop + Flip	Test set 1	74	34	1681	2	0.68	0.97	0.99	0.8

Table 6.2: Results with the Red Carpet database

When we use combined methods the rotation and the cropping that we are applying is the one that gave us better results in the previous steps.

7. Budget

Software	Units	Price/Unit	Total Price	Amortization Period	Amortization
Matlab Licence	1	2586€	2586€	10	233€
Ubuntu + Eclipse	1	0€	0€	10	0€
		Total	2586€		233€

Salaries	Days	Hours/Day	Price/Hour	Total
Technician	140	5	8€	5600€

Total Project cost: 8186€

8. Conclusions and future development

In this project, different ways to detect near duplicate images have been presented.

Using three different databases we could see the different performance of the color descriptors and the SIFT transform at this approach. As we can observe in the results this problem is not easy to solve so once we obtain the results for each descriptor we decide to improve the best one.

When we obtain the final results we can extract two different conclusions.

On the one hand, if we focus our work in the IMCOP project and the best performance in the red carpet database have to be obtained we could say that using the CSD is the best way to solve it. Also if they focus the research on detecting all near duplicate images in the database, the image cropping method one is the best approach because we obtain almost a perfect sensitivity only missing two images. The cost of this method is that increasing the sensitivity we are decreasing the precision of the system so we are detecting more images as they were near duplicate.

Without improvements:

Database	**Method**	**Set**	**TP**	**FP**	**TN**	**FN**	**Precision**	**Sensitivity**	**Specificity**	**F-measure**
Red carpet	CSD	Training	90	16	1330	8	0.85	0.91	0.99	0.88
Red carpet	CSD	Test set 1	70	10	1681	6	0.87	0.94	0.99	0.9
Kentucky	CSD	Test set 2	106	0	1758	70	1	0.6	0.99	0.75

Table 8.1: CSD results without improvements

Testing sets with improvements:

Database	**Method**	**Set**	**TP**	**FP**	**TN**	**FN**	**Precision**	**Sensitivity**	**Specificity**	**F-measure**
Red Carpet	Cropping method 1	Test set 1	74	16	1681	2	0.82	0.97	0.99	0.89
Kentucky	Cropping method 1	Test set 2	110	0	1758	66	1	0.62	0.99	0.77

Table 8.2: CSD results with image cropping method 1 improvements

On the other hand, if we focus our work in an academic way and the best performance at both databases have to be obtained we could say that also CSD is the best choice but instead of using image cropping as improvement we would use the combining methods of image rotation and image flipping. With this method we decrease a little bit the performance of the red carpet database but the general results of both databases are better than the ones obtained without improvements.

Without improvements:

Database	**Method**	**Set**	**TP**	**FP**	**TN**	**FN**	**Precision**	**Sensitivity**	**Specificity**	**F-measure**
Red carpet	CSD	Training	90	16	1330	8	0.85	0.91	0.99	0.88
Red carpet	CSD	Test set 1	70	10	1681	6	0.87	0.94	0.99	0.9
Kentucky	CSD	Test set 2	106	0	1758	70	1	0.6	0.99	0.75

Table 8.3: CSD results without improvements

Testing sets with improvements:

Database	**Method**	**Set**	**TP**	**FP**	**TN**	**FN**	**Precision**	**Sensitivity**	**Specificity**	**F-measure**
Red Carpet	Rot + Flip	Test set 1	70	18	1681	6	0.8	0.92	0.99	0.86
Kentucky	Rot + Flip	Test set 2	124	0	1758	52	1	0.7	0.99	0.82

Table 8.4: CSD results with rotation and flipping improvements

As a final conclusion, it is important to know that depending on which database we are working on, different descriptors will be necessary. For instance, if the training database would be similar to the Kentucky one we could use the SIFT transform and obtain good results while if the database will be similar to the red carpet we will have to choose the CSD. That is one of the reasons that the near duplicate image problem doesn't have a general solution. We will have to know with which database we are dealing and which solution will best fit.

As a future work, we could use face detection algorithm is order to detect better the faces in the red carpet database and apply a more efficient image cropping.

Bibliography

[CH14] Ashwin. "How to work with ROI in OpenCV" [Online] Available: http://choorucode.com/2014/05/09/how-to-work-with-roi-in-opencv/.[Posted on May 9, 2014]

[CPZ08] O. Chum, J. Philbin, A. Zisserman. *Near Duplicate Image Detection: min-Hash and tf-idf Weighting*, British Machine Vision Conference, CMO, Czech Technical University in Prague ,VGG, University of Oxford, 2008

[DB07] D. Briggs. "The Dimensions of Colour". [Online] Available: http://www.huevaluechroma.com/083.php. [Posted on 2007]

[DEP13] IMCOP project information (2013-2015). [Online] Available: http://www.kt.agh.edu.pl/en/projekt/622

[EGL14] A. Eshkol, M. Grega, M. Leszczuk, O. Weintraub. *Practical Application of Near Duplicate Detection for Image Database,* Orca Interactive, Israel, AGH University of Science and Technology, Krakow, Poland, 2014.

[FA11] F. Abecassis. "OpenCV – Rotation". [Online] Available: http://felix.abecassis.me/2011/10/opencv-rotation-deskewing/. [Posted on October 4, 2011].

[JP15] JPEGmini. [Online] Avaiable: http://www.jpegmini.com/

[JPM11] "JPEGmini: More Efficient Image Compression." [Online] Available: http://www.websiteoptimization.com/speed/tweak/jpegmini/. [Posted on August 29, 2011]

[NS06] D. Nistér and H. Stewénius. *Scalable recognition with a vocabulary tree*. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 2161-2168, June 2006. [Online] Available: http://www.vis.uky.edu/~stewe/ukbench/

[LF13] Z. Li, X. Feng. *Near Duplicate Image Detecting Algorithm based on Bag of Visual Word Mode*l, Journal of Multimedia, Vol 8, no.5, Henan Xinxiang, China, October 2013, p. 557.

[L04] D. G. Lowe. *Distinctive Image Features from Scale-Invariant Keypoints*, International Journal of Computer Vision, University of British Columbia, Vancouver, B.C., Canada, 2004.

[MSS02] B. S. Manjunath, P. Salembier, and T. Sikora, Introduction to MPEG-7, *Multimedia Content Description Interface*, John Wiley and Sons, Ltd., Jun 2002.

[OC14] OpenCV 2.4.9.0 documentation. “Remapping”. [Online] Available: http://docs.opencv.org/doc/tutorials/imgproc/imgtrans/remap/remap.html. [Update on April 21, 2014]

[OC14] OpenCV 3.0.0 documentation. “Geometric Transformations of Images”. [Online] http://docs.opencv.org/trunk/doc/py_tutorials/py_imgproc/py_geometric_transformations/py_geometric_transformations.html. [Update on Dec 30, 2014]

[RJ1] R. Johri. “Color Descriptors from Compressed Images”. [Online] Available: http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/AV0910/johri.pdf.

[SE10] S. Emami. “Rotating or Resizing an Image in OpenCV” [Online] Available: http://www.shervinemami.info/imageTransforms.html. [Posted on 9th June, 2010].

[VR10] C. Ventura. “Image-Based Query by Example Using MPEG-7 Visual Descriptors”. D. Thesis, Image and Video Processing Group, Faculty of Telecommunications Engineering, Universitat Politècnica de Catalunya, Barcelona, March 2010.

[VLF07] “SIFT detector and descriptor” [Online] Available: http://www.vlfeat.org/overview/sift.html. [Posted on 2007]

[YZC09] X. Yang, Q. Zhu, K-T. Cheng*. MyFinder: Near-Duplicate Detection for Large Image Collections*, MM'09, Beijing, China, 19–24 October 2009, p. 1013.

Appendixes

A. Color Spaces

The color space [RJ1] is used to specify the color space that a given descriptor refers to. It defines four color spaces: RGB, YCbCr, HVS and HMMD.

A.1. RGB Color Space

The RGB color space is one of the most popular models. This space is defined as the unit cube in the Cartesian coordinate system which has Red (R), Green (G) and Blue (B) additive primaries as a basis. These colors are added together in various ways to reproduce a broad array of colors. Each of the three primaries is called a component of that color, and each of them can have an arbitrary intensity, from fully off to fully on, in the mixture. Zero intensity for each component gives the darkest color, and full intensity of each gives a white [VR10].

When the intensities for all the components are the same, the result is a shade of gray, darker or lighter depending on the intensity. When the intensities are different, the result is a colorized hue, more or less saturated depending on the difference of the strongest and weakest of the intensities of the primary colors employed.

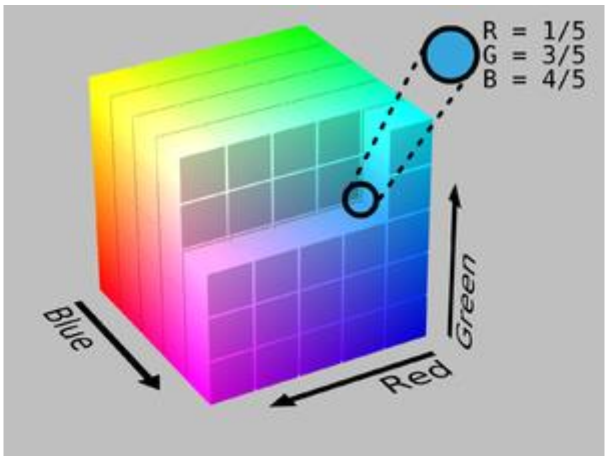


Figure A.1: RGB cube

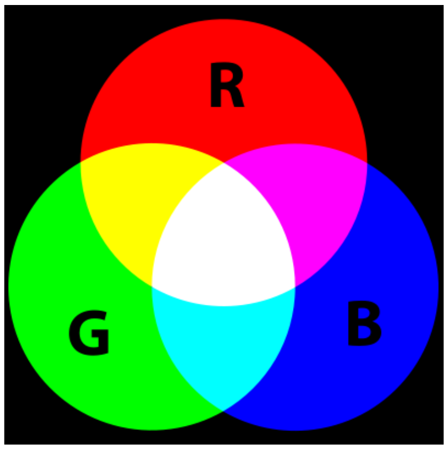


Figure A.2: RGB model

A.2. YCbCr Space

YCbCr is another color space where the component Y represents the luma, i.e. the brightness, and Cb and Cr are the blue difference (B-Y) and the red difference (R-Y) components, respectively. One advantage of the YUV color format is based on the characteristics of the human visual perception: Since the human eye is much more sensitive for brightness information compared to color information, we can give less importance to the chrominance information in some dissimilarity that are computed in this color space, such as Color Layout Descriptor.

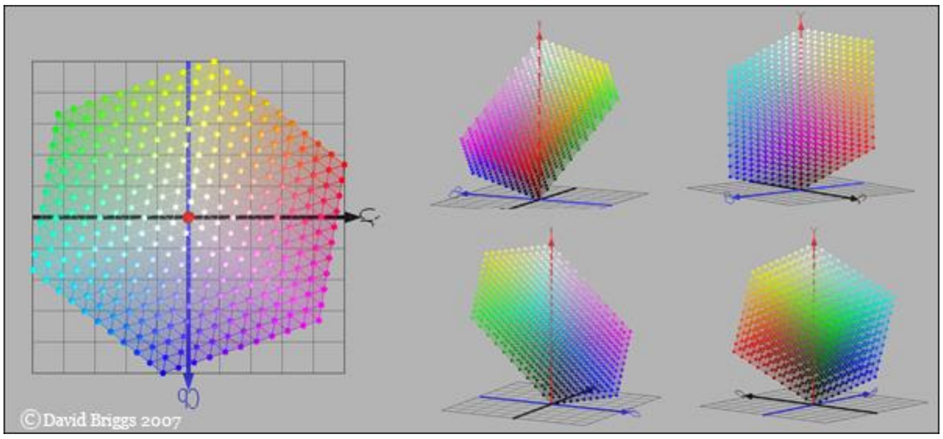


Figure A.3: RGB colors arranged in YCbCr color space, using the program RGB Cube

A.3. HSV Color Space

The HSV color space is developed to provide an intuitive representation of color and to approximate the way in which humans perceive and manipulate color.

Hue (H) specifies one color family from another, as red from yellow, green, blue or purple. Saturation (S) specifies how pure a color is and Value (V) specifies how bright or dark a color is. The HSV color model can be seen as a double cone.

The angle around the central axis represents the hue, the distance to the axis represents the saturation and position along the central axis represents the value or luminance.

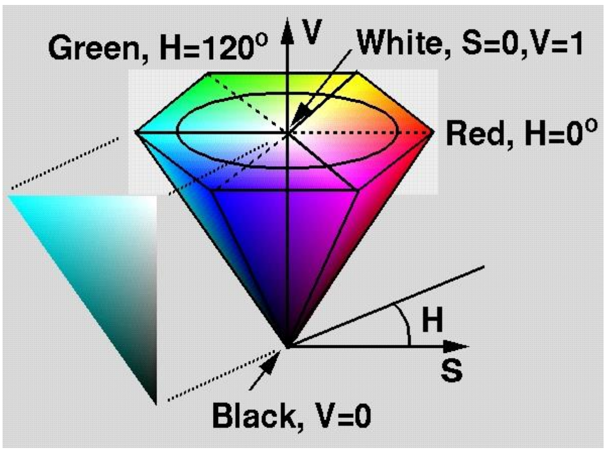


Figure A.4: HSV color space

A.4. HMMD Color Space

The HMMD (Hue-Max-Min-Diff) color space is closer to a perceptually uniform color space. The Hue has the same meaning as in the HSV color space. Max and Min components are the maximum and minimum among the R, G, B values, respectively. The Diff component is defined as the difference between max and min. Even though the four components are identified in the name of the color space, one more component, Sum, can be defined as the average of Min and Max components. Only three of the five components are sufficient to describe the HMMD color space.

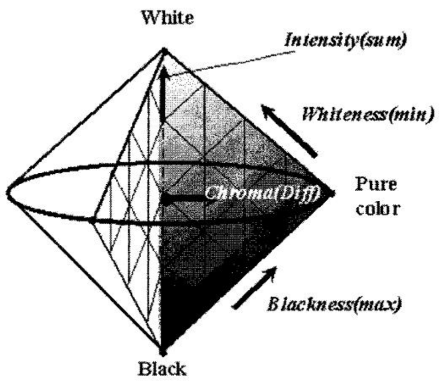


Figure A.5 HMMD color space

B. Databases

We use three different databases in order to obtain the performance of each descriptor.

B.1. Training set

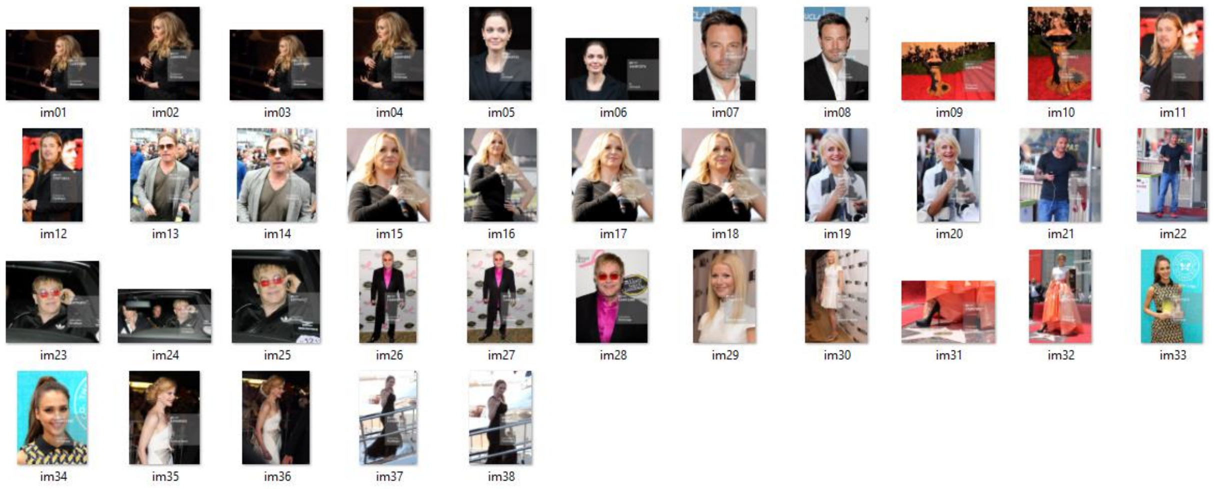


Figure B.1: Training set

B.2. Test set 1

Figure B.2: Test set 1

B.3. Test set 2

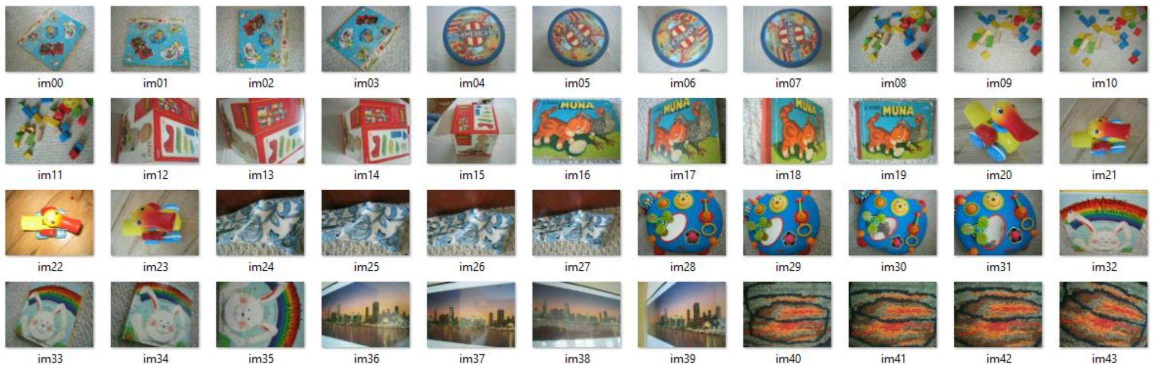


Figure B.3: Test set 2

Glossary

DOG:	Difference of Gaussian
LDP:	Local Directional Pattern
LSH:	Local Sensitive Hashing
RANSAC:	Random Sample Consensus