MASTER THESIS

TITLE: Focus Detection and Sharpness Evaluation in Keyframes containing Faces

MASTER DEGREE: Master in Science in Telecommunication Engineering & Management

AUTHOR: Rodrigo Sampedro Casis

ADVISOR: Francesc Tarres Ruiz

DATE: June, 28th 2017
Title: Focus Detection and Sharpness Evaluation in Keyframes containing Faces

Author: Rodrigo Sampedro Caiss

Advisor: Francesc Tarres Ruiz

Date: June, 28th 2017

Abstract

In this work, a software tool to blur detection was implemented in order to select the best image-frame in a set of key frames. The main objective is to allow the detection and measurement of the blurring level of an image without human intervention, i.e. by artificial intelligence trained to detect blur. During the implementation of this Master Thesis, it was necessary to understand the concept of the blur, the causes and the different algorithms to detect local blur.

The first chapter summarizes the main causes of blur oriented to understand the difficulty of the problem and a brief introduction of state of the art in blur detection. The second chapter shows the first steps and problems encountered to understand the final solution adopted. Also during the second chapter, each partial-solution and their integration were explained in detail. Finally, in the third chapter, the results and conclusions of the work are explained and compared with other current research.

This work uses multiple methods to detect local blur, analysing neighbour’s results with different types of filters. Therefore, the solution is a local blur detector at pixel level that generates two images as output, one mask of blurred/sharped pixel areas, and a grey-image with the different levels of blur per pixel. However, the blurring detection is applied in 1D (one output per single pixel) losing its 2D position in the image but using the neighbouring pixels’ information to convert this method in a 1.5D.

On the other hand, the decision thresholds to classify the image as blurred or sharp were created by machine learning algorithm based on using Naïve Bayes techniques and Neural Networks solutions, to get a similar result to human blur compression. Thus, a large database with different types of blur and sharpness levels has been used with a ground-truth labelling by hand.

Finally, to get a more precise result, a multiscale analysis of each image was implemented to solve false positives over textures or far-away objects, hence cleaning the noise the result. To conclude the result is a stable software able to accomplish the set goals, with an efficiency similar to that of a human person classification.
CONTENTS

INTRODUCTION ........................................................................................................... 1

1. INTRODUCTION TO BLUR CONCEPT ................................................................. 2
   1.1. Objectives and Requirements ........................................................................... 2
   1.2. Software Deployment ....................................................................................... 2

1.3. DEFINITIONS AND IDEAS ................................................................................ 3
   1.3.1. Causes and types .......................................................................................... 4
   1.3.2. Textures, flat areas and blur regions ............................................................ 7

1.4. State of Art .......................................................................................................... 8
   1.4.1. Types of Detectors ....................................................................................... 8
   1.4.2. Techniques & ideas of blur detection .......................................................... 9
   1.4.3. Most Used Non Reference detectors .......................................................... 9

2. BLUR FEATURES .................................................................................................... 11
   2.1. First approach to the problem ......................................................................... 11
       2.1.1. Single Feature .......................................................................................... 13
       2.1.2. Problems of Single variance model ......................................................... 15
       2.1.3. Feature Patch Concept .............................................................................. 16
       2.1.4. Improve results based on features or scale ............................................. 18

   2.2. Features ........................................................................................................... 18
       2.2.1. Feature 1: Variance Estimator ................................................................. 19
       2.2.2. Feature 2: Variance rate ......................................................................... 20
       2.2.3. Feature 3: Log Kurtosis .......................................................................... 21
       2.2.4. Feature 4 Power Spectrum trends ........................................................... 24
       2.2.5. Feature 5, 6, 7: Local Linear Filters ....................................................... 25
       2.2.6. Feature 8 and 9: Colour Saturation ....................................................... 27

   2.3. Importance of features .................................................................................... 28

   2.4. Ambiguous threshold ...................................................................................... 29

   2.5. Data Base ......................................................................................................... 30

   2.6. Scale join prediction ....................................................................................... 31

   2.7. Analysis of results ........................................................................................... 33

   2.8. Decision algorithms ....................................................................................... 34
       2.8.1. Naive Bayes ............................................................................................. 35
       2.8.2. Naive Bayes Implementation ................................................................. 35
       2.8.3. Neural Network ....................................................................................... 38
       2.8.4. Neural Network Implementation ............................................................ 40
       2.8.5. Combinational Result .............................................................................. 44

   2.9. Face analysis approach .................................................................................... 44
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.</td>
<td>RESULT AND COMPARATIVE</td>
<td>46</td>
</tr>
<tr>
<td>3.1.</td>
<td>Multi-scale Results of my method</td>
<td>46</td>
</tr>
<tr>
<td>3.2.</td>
<td>Comparatives</td>
<td>47</td>
</tr>
<tr>
<td>4.</td>
<td>CONCLUSIONS</td>
<td>51</td>
</tr>
<tr>
<td>4.1.</td>
<td>Algorithm conclusions</td>
<td>51</td>
</tr>
<tr>
<td>4.2.</td>
<td>Knowledge and personal conclusion</td>
<td>51</td>
</tr>
<tr>
<td>ACRONYMS</td>
<td></td>
<td>52</td>
</tr>
<tr>
<td>REFERENCES</td>
<td></td>
<td>53</td>
</tr>
<tr>
<td>ANNEX A</td>
<td></td>
<td>55</td>
</tr>
<tr>
<td>1.1.</td>
<td>Schedule of Thesis</td>
<td>55</td>
</tr>
<tr>
<td>1.2.</td>
<td>Code</td>
<td>56</td>
</tr>
<tr>
<td>1.2.1.</td>
<td>Previous scripts</td>
<td>56</td>
</tr>
<tr>
<td>1.2.2.</td>
<td>Final Code</td>
<td>56</td>
</tr>
<tr>
<td>1.3.</td>
<td>Hardware and simulation</td>
<td>58</td>
</tr>
<tr>
<td>1.4.</td>
<td>Data Results</td>
<td>59</td>
</tr>
<tr>
<td>1.5.</td>
<td>Data Base</td>
<td>59</td>
</tr>
<tr>
<td>ANNEX B</td>
<td></td>
<td>61</td>
</tr>
<tr>
<td>1.</td>
<td>DECISION ALGORITHMS</td>
<td>61</td>
</tr>
<tr>
<td>1.1.</td>
<td>Naïve Bayes Classifier Algorithm</td>
<td>61</td>
</tr>
<tr>
<td>1.2.</td>
<td>K Means Clustering Algorithm</td>
<td>61</td>
</tr>
<tr>
<td>1.3.</td>
<td>Support Vector Machine Algorithm</td>
<td>61</td>
</tr>
<tr>
<td>1.4.</td>
<td>Linear Regression and Logistic Regression</td>
<td>61</td>
</tr>
<tr>
<td>1.5.</td>
<td>Nearest Neighbours</td>
<td>61</td>
</tr>
<tr>
<td>1.6.</td>
<td>Artificial Neural Networks</td>
<td>62</td>
</tr>
<tr>
<td>1.7.</td>
<td>Random Forests and Decision Trees</td>
<td>62</td>
</tr>
<tr>
<td>1.8.</td>
<td>Summary decisions</td>
<td>62</td>
</tr>
<tr>
<td>1.1.</td>
<td>Visual comparative of learning methods</td>
<td>62</td>
</tr>
<tr>
<td>2.</td>
<td>NAÏVE BAYES IMPLEMENTATION</td>
<td>63</td>
</tr>
</tbody>
</table>
2.1. Creation of Big files ........................................................................................................... 63
2.2. Implementation of all database .......................................................................................... 64
2.3. Detection of different types of blur .................................................................................... 64
2.4. Results by scale ................................................................................................................ 65

3. NEURAL NETWORK IMPLEMENTATION ...................................................................... 65
   3.1. Properties and tuning of Neural Nets .............................................................................. 65
   3.2. Visual representation of Neural structure ........................................................................ 66

4. RESULTS ANALYSIS ......................................................................................................... 68
   4.1. Final Scale results over Database include training ......................................................... 68

ANNEX C .................................................................................................................................. 73
   1. VISUAL COMPARATIVE ................................................................................................. 73
   2. SOME IMAGES RESULTS ............................................................................................... 75
INTRODUCTION

The main purpose of this project is to solve an automation problem in selecting key frames for online video platforms. Some companies like Ugiat Technologies [1] have algorithms to segment scenes and detect sets of representative key frames, these key frames are useful for online platforms but currently, the selection of the best key frame continues to be manual. One important parameter to select the most appropriate key frame is a low blur effect or low blur-motion in the faces as well as smiles and open eyes.

The main work of this project is to develop an algorithm for evaluating the sharp areas of the image and create metrics to decide the best image in a set, based on sharp image information.

This document explains the causes and analyses the different ways to detect blur with an algorithm using low-level features that detects and decides whether a pixel is sharp or blurry. On the other hand, the algorithm allows to assess blur/sharp information creating one blur-grey-image letting know if regions of interest like face, smile or eyes are un-blurred and the level of blur of each one.

During the realization of the project various methods, strategies and technologies of current interest in the automation of tasks, such as training and machine learning have been used to optimize and improve the results.

Finally, once the algorithm detects the blurred areas, a qualitative, quantitative and comparative measure of the results has been made to know if the result of the software fulfils the requirements and whether objectives have been achieved.
1. Introduction to Blur Concept

1.1. Objectives and Requirements

The main objectives of this work have been oriented to get an automatic blur detection, the first step is to understand blur/sharp conditions in order to create blur estimators to measure, the second step tries to get a generic algorithm and finally the third try to customize the algorithm by specific training. The algorithm to evaluate face blur level must be at the pixel level, i.e. one measured per each pixel generating one or more measured image features as result.

The feature or features should detect the level of blur of each pixel and classify as blurred or sharped, generating two different regions. The result should be enough robust with different types of blur, scale and avoid false detections. These results so demanding, require a complex solution based on several estimators at different scales. On the other hand, multiple complex measures need multidimensional threshold as decision makers, i.e. a large database of labelled images and machine learning techniques must be used.

During the research, development and testing process an advance knowledge of image processing, machine learning and blur properties has been acquired as a personal goal of the thesis. Moreover, the software must be easy to understand, modular and easy to update or implement improvements.

Finally, the short-terms goals as steps have been:

1. Understand blur concept and blur/sharp detection.
2. Understand Ugiat software to get key frames and faces, eyes and smile regions.
3. Test basic/simple blur estimator.
4. Improve basic detector reading papers and current blur estimator.
5. Add multiple blur/sharp estimator, multiple features as measure result.
6. Define threshold decisions based on result analysis.
7. Get or generate a test database.
8. Machine learning or similar methods for auto define thresholds by trainings.

A more and precise diagram of time can be found in Annex A. point 1.

1.2. Software Deployment

During the realization of this work, the software of Ugiat Tech has been used to extract key frames, which contain faces, smiles and open eyes. The main work is to develop a blur detection as a module to help Ugiat Tech software. The final
software development has been made in Matlab but some functions and external software are based on C/C++.

During the first part of the project an introduction to C++ under QT framework was made in order to use and modify Ugiat software. The first approach of basic ideas like import, resize and get edges was made in C++ using opencv libraries.

After the second test of variance as blur estimator, C++ was too hard than other script solutions for development. However Matlab software allows to use script solutions, with a similar syntax of opencv in C++ (fast translation) and implementing the major of opencv functions and its own extra functions. Also, Matlab has one easy environment for debugging, that allows to inspect data and plot results without external libraries. Finally, Matlab was defined as main develop language because is an easier form to develop the algorithm without the hard develop problems of C++.

One important functions of Matlab is the use of cuda or parallel processing without big change in the code (pseudo-automatic, easy to make). The worst part is the use of non-compiled language (scripting language) making this option a slower solution than C/C++ solutions for business implementation.

The main code is structure in three scripts, one script that load, processes images calling concrete modular functions and save results. When the database has been full analysed, a second script concatenates the database result to use as input for learning machine or support vector machine libraries that generate the decision makers. Finally the last script load and process test images and use the learned decision makers to analyse the blur levels.

Most parts of the codes are modular, using Matlab functions. Also, specific parts of the code are in C/C++ (functions) recycle from reference papers related with spectrum features.

Finally, all software except Ugiat external sources are published in [2] GitHub link with the trained decision makers. The database and the result are storage by Ugiat tech due to the big size of the information (more than 260 Gbytes).

1.3. Definitions and Ideas

To understand how the detection of blurred zones works, we must clarify its causes and the definition of the blur.

The human visual system, it is a perfect neural software that distinguishes a blur/sharp photo, this “simple fact” for people is based on our training during childhood and continues being a mysterious mechanism nowadays. Currently, this decision only can be approximated detecting the math bases of blur and using machine learning to decide.

A good definition of blur is one of the most common degradations in a photo, popularly it is defined as an area where a person cannot see clearly or edges are
no longer to distinct. Blur idea is related with low detail, poor visual information, in a more technical words: low quantity of edges or low power at high frequencies and also is related with average filters effect, digital zoom process and low entropy areas.

In this work, blur is defined as an area with low quality visual detail where normally the information is distributed or diluted in a difficult form to see. Accordingly, detect blur is based on measuring the quantity, quality and form of one object information in the image.

1.3.1. Causes and types

In photography, there are some types of blur, the most extended type is motion blur but there are more, like focus blur, zoom blur and optic/sensor blur.

All types of blur are caused by the same problem; the information is not captured in the most appropriated form (perhaps by the photographer or by the limitations of the hardware). The mechanism in which the images are captured is the root of the causes of blurred images on digital cameras or blurred vision in humans, such as myopia, astigmatism or farsightedness.

The degradation in the image information is defined as Point Spread Function, (PSF) that it is a general optical system’s impulse response, in other words, a function that defines how one point information is spread over the neighbour pixels when the object of the pixel is not correctly focused or is in motion.

![Optics and sensing mechanism](image)

**Fig. 1 Optics and sensing mechanism.**

The creation of an image is based on optics and light sensing. The optic system allows to converge the light rays in a plane that recreate the original object inverted, this recreation image is capture by a lot of little sensors. The optic limitations or the sensor limitations are the causes of the blur.

- Motion blur: the real world object move in time, i.e. image is an integrated detection of a recreated object in the sensor plane in time. If the sensor needs a minimum time to detect the light (shot), motion is integrated in the sensor. Fast movement can mix the information during the capture in similar form as average filtering in the direction of movement (object in movement or the camera movement).
Focus Detection and Sharpness Evaluation

- Focus blur: The optics and the sensor position fix where the focus plane and the image plane are, that mean that only objects in a concrete depth of field are in focus and the rest of objects are blurred by the optics.

- Digital Zoom blur: When an image is digital zoomed, the resolution of the zoomed area is large, but the information is the same. The computer interpolate the new pixels based on the originals to avoid image pixilation. The result is a bigger area (zoomed), but the new pixel are similar to average filter result, i.e. similar to natural blur image.
Focus Detection and Sharpness Evaluation

Fig. 5 Digital zoom effect without average filter (centre) with average filter (right).

- **Optic / Sensor blur:** The real optic and the real sensor are limited, which make fixed the depth field of the camera, by the hardware. The cause of blur in very far/close objects are the optics limitations that cannot improve and sensor sensibility which fix the motion blur umbral.

Fig. 6 Depth of Field diagram explanation in function of aperture.

- **Natural blur:** Sometimes environment physics change how the light rays behaviour, creating a natural blur, this is the case of the atmosphere with Rayleigh scattering, high humidity/pollution conditions or fluid fluctuation for a big zoom like astronomic photos.

Fig. 7 Natural blur, scattering (left) and atmospheric fluctuations (right).
1.3.2. Textures, flat areas and blur regions

Textures and flat region are similar to blur regions because both have poor detail information, one example is a photo of the sky. The feature terms of low edge images are practically the same for a non-blurred or blurred shot, the sky is very similar in focus and blur images because is a background in the photo and doesn’t provide relevant information. The same problem happened with a forest or grass, near and far areas are similar in their features terms making difficult define when the image change from sharp textures to blurred textures.

![Fig. 8 Textures examples of background.](image)

How to distinguish a smooth white wall, from one rough but blurred? Depend on the rest of the image. The sky, the forest, the sea or other type of background textures, are not appropriated to evaluate without another parts of the image that are the important objects in the picture.

The problem to evaluate textures or flat regions is the context of the image. A face has eyes, mouth, and nose with a lot of edges and cheeks and forehead with flat regions with low edge. People do not detect blur in human flat face regions because the context of the edges regions conclude that the face is not blurred because the edge regions are fine, unblurred.

![Fig. 9 Low edge face part with 0 sigma (left), 0.8 (center), 1.6 sigma (right).](image)

In the Fig. 9 images with only a little part of it are edge region (nose) as result is difficult to estimate how blurred the images are, based solely on flat regions, but in an image with eyes and mouth or hair is easier.
1.4. State of Art

Studies on blur regions detection and characterization of un-sharp have become one of the important research branches since image processing started. In order to improve zoom images, de-blur image or compensate blur-motion some strategies of detection, classification and restoration had been created. Currently continues to improve driven by the digitization of content, the multimedia boom of the internet and the increase of capacities in consumer electronics such as smartphones.

There are different types of algorithms depending on complexity requirements and time response and others that need references or training images. Mainly the most widespread algorithms are based on grey analysis but there are a few algorithms based on colours analyses.

The common use of the digital camera at end of 90's and 2000's, popularized algorithms for focus detection (based on reference detection), just as smartphones have popularized image enhancement, compensation function and classification in 2007. The need to measure the introduction of blur in compression algorithms, the use of super-resolution techniques combining multiples sharp images and the human visual perception models in the current devices have exponentially increased the number of publications and new methods, parameters and comparative databases of blur detectors over last 5 years, understanding it as a hot topic with research and development.

1.4.1. Types of Detectors

Usually, the algorithms are classified by the complexity, which is related to their consumption of resources. In blur detection methods, this classification is summarized under the necessity or not of having references (multiple relative’s images) to create a metric value:

- FR (Full Reference metrics), original image is compared with previous or predefined image to know if the original is better/worse than reference. These metrics are easy to compute (less resources) and not require training for decision, very useful for videos or compression blur detection.

- NR (Non Reference metrics), does not need reference information, requires more complex algorithms and trained decision makers. These type of algorithms are more similar to human vision perception but they need good databases adapted to the specific problem for training.

- RR (Reduced Reference metrics), use some available features extracted from previous reference as edges to compare with actual image. It is a trade-off between FR and NR, allow to implement metric with high time requirements like camera focus detection/correction. Example safe statistics during some photograms, compare and decide based on previous data.
As project’s requirement, it is necessary to detect sharp/blur on generic images, based only on one key frame, this work uses **Non Reference (NR) metrics** not needing reference images.

### 1.4.2. Techniques & ideas of blur detection

As I mentioned before, the blur is very similar to an average filtering or Gaussian filtering, one of the first types of techniques for detecting blur is very similar to the method most used to fix it, blind image de-convolution. Gaussian/PSF de-convolution techniques were created in 70’s for astronomical images and improved in 90’s. These types of methods are based on recovery of the PSF (Point Spread Function, explained 1.5.1) to restore the image. Many of these methods [3] based on PSF function and de-convolution or Gaussian de-processing, need great resources, long computing times and often they need reference image to restore original. Before the complex process, the blur can be measured based sigma value of gaussian convolution for the specific PSF.

Low DoF (depth of field) methods, are based to detect photography technique, where usually one object is focused near the centre and surrounding pixels are out of focus. This technique uses frequency and spatial comparisons using wavelet transform, creating a DoF indicator based on the ratio of wavelet coefficient in high frequencies. This type of detectors [4] are not a general blur detector for blur detection on commons images.

Low directional high frequency energy methods [5], are methods used to measure the blur motion based on evaluate the high frequency energy and calculate the direction of the motion of an image. The idea is to define the energy as sum of squared derivative of image and the concept of high frequency energy decreased incomparably along the direction of the motion in blurred image.

Edge sharpness analysis methods [6] are techniques based on getting the Edge of the image (related with high frequency) and create parameters or coefficients based on statistics or measures over the edges. This is the main type of technique that is used in this document, because allows to use different types of methods over the same input (edges image), usually have low processing time requirements.

Finally, there are extra techniques based on frequency phase [7], different types of transforms [8] and combination of the previous types which currently are not the hot spot for development but some of they are used in this software. A good comparison of most common methods can be found in this publication [9].

### 1.4.3. Most Used Non Reference detectors

Within the framework of non-reference and mainly in Edge Sharpness analysis, there are different math-statistics models for detecting blur:
Variance metrics: flat and blurred regions have low variance between pixel information, these type of metric measure the variance to get a flatness-coefficient where best focussed images get greater intensity variations than blurred images. This models [10] measure the variance of the histogram of edge pixels images (measure wide of normal histogram).

Autocorrelation metrics: autocorrelation function (ACF) is the inverse Fourier transform of the power spectrum, ACF will contain sharpness information and smooth regions, result in a low broad ACF peak at blurred images [11].

Edge Histogram entropy: anisotropy can be measure as the entropy evaluated at different directions. Anisotropy decrease as more degradations are added to the images (more blurred per example) [12].

Histogram frequency or spectral trends: blurred images have low power at high frequencies, the discrete cosine transform (DCT) coefficients and the trends of power spectrum derivative allows you to differentiate unfocused images.

Edge Width analysis: a sharp image have narrow edges, a blur image have wide edges, measuring the edge width is an estimate of how fuzzy an image can be [13].

Local edge kurtosis: similar to variance metric, allow to measure how peak is the histogram of edge pixels [14]. The relation between wide and high of histogram.

Derivative magnitude statistics: use Gradient magnitude and Laplacian magnitude as high pass filter. Accordingly, the distributions of the log gradient magnitude for blurred regions should have shorter tails than that for other regions.

These detectors outputs are numeric parameters like statistics (mean, variance, maximum) or visual parameters of the relationship with its neighbours. Normally it is necessary to use functions that allow normalizing or dimensionless the parameter and adjust the results or adapting to the human perception model.

Finally, almost the most used methods or estimators have been studied for the realization of this project, however only a few have been finally implemented, due to lack of resource, effectiveness or complexity problems. Moreover most part of the implementations, show the result parameter normalized (0-255) to show in a 2D image, a visual interpretation of the measure.
2. Blur Features

2.1. **First approach to the problem**

This point is an explanation of the first strategy performed, tests and results obtained as the introduction for 2.2 the final procedure solution.

As a first step in the realization of the project, a blur image detector based on derivative filters was performed. The first stages are responsible for importing the image, rescaling (if it exceeds the maximum size) and extracting the grey image.

The software analyses image of 600K of the pixel (around 780x780 standard picture), if the size is greater, the image is scaled with the same proportion to 600K pixels of resolution. The software uses a bicubic interpolation in 2D with the nearest 4-by-4 neighbourhood that is the best approximation without artificial effects.

![Fig. 10 Most extended interpolation methods 1D and 2D.](image)

The conversion from colour image to grey image is based on luminance computed as:

\[
Luminace = 0.2989 \times Red + 0.5870 \times Green + 0.1140 \times Blue \quad (1)
\]

![Fig. 11 Original (left) and grey-scale image result.](image)

The second step, extract the image’s edges, reduce noise with different filters to know which is the most appropriated for our case. Only the magnitude of the edge is useful for our case.

Then gradient or laplacian filters are applied to extract the edge of the image, selecting a kernel size of 3x3 or 5x5 to reduce image’s noise and remove fiction
edges. When and image is sharp the variance of edge’s pixels values get high results, when image is blurred get low results.

As third step, an easy-dummy detector based on variance measure the level of blur, other parameters are collected to normalize the measure in step fourth. These extra parameters are related with area, brightness, level of decision etc.

Variance (5) is a common an easy statistic parameter, but has problems with high-noisy images because noise’s power is related with variance, distorting the measurements.

Thus, gradient filter is in general better than laplacian filter, because the second is more sensible to noise interference and only in low noise and high texture images laplacian filter get better results than gradient filter.

The fourth main step is based on normalize and compare with the results of other images. As summary the process to get the edge and measures of blur based on variance of one image is:

1. Pre-noise filter, average filter 3x3, mode symmetric.
2. Select kernel size, Sobel technique with kernel of 3x3, 5x5, 7x7 and 9x9 based on image size (Normally 3x3 o 5x5).
3. Get derivative image in X axis and Y axis.
4. Sum both images as vector to get the module, only magnitude is interesting $edge_{mag} = \sqrt{derivative_x^2 + derivative_y^2}$ (2).
5. Calculate grey-threshold based on Otsu’s method for mask edge area.
6. Calculate sum of pixel as pseudo energy $E_{img} = \sum pixel_{value}$ (3) and edge’s area as result of edge’s image binarization and dilate based on Otsu results.
7. The variance pixel value of edge’s image is normalized with the area of edges and pseudo energy. $Var_{norm} = variance \times \frac{E_{img}}{A_{edge}}$ (4), in order to compare image with different quantity of edges and edges intensity.
This model based on variance allows to compare similar images or consecutive images in a video (reference method). The use of extra parameters related with size, edges’ quantity or edges’ area, compensate if the image has a low or high quantity of edges allowing to compare different types of images with similar noise and textures conditions.

A second approach was based on edge’s width measures, blurred images have wider edges than sharp images, the laplacian filter is better to get a defined edge width. Some papers like [15] based on JNB (Just Noticeable Blur) technique was implemented during this project, but its complexity compared to statistical methods like the variance or kurtosis, forced to discard width methods.

2.1.1. Single Feature

The first software test, made on C++ QT and opencv libraries show that after 15 test, the right approach to the problem is based only on gradient edge image variance. Therefore this method allows to compare images but not define a threshold that separates the different scales of blur of each image.

In order to get a non-reference method, the original image has been blurred to compare it with itself. The idea is fansy, “blur” areas are similar after Gaussian filter, but focussed areas change, measuring the rate of changes on sharpness areas, the original image can be catalogued. The idea was proposed on “A No-reference Image Blur Metric Based on Two pass Edge Analysis” [16] based on width edge analysis, however, in my work, it is done with the variance rate change.

After analysing the behaviour of 150 photos, it has been detected the existence of a range between 1-3 sigma as parameter of Gaussian filter that has different trends on blurred and sharped images.

The procedure is based on the variance changes between original and digital blurred images. Comparing the variance changes between original and different levels of blur manage by gaussian filter’s sigma, the rate of change can be measured.
The trends show that blurred images have a lower rate of change that sharpened images. In flat or blurred areas do not significantly change the values of the regions and only slightly modifies their variance.

![Fig. 13 Process to get variance rate of change.](image)

On base of results, a classification threshold can be defined. In figure 14, 100 images analysed show original focus images red, 0.8 sigma yellow, 1.2 sigma green, 1.6 sigma blue and 2.4 sigma purple. For values higher than 1.2 sigma human vision detect the image like blurred.

![Fig. 14 Rate of change Variance at different blur level, when image is blurred.](image)

The conclusion is that this feature apparently is quite useful to evaluate between focused or totally unfocused photos, it can also be used to analyse which photo is better, but it does not end up being completely generic.

When photos have too many textures get high results and flat images get low results due to the extra parameters that normalized variance are not enough to compensate the different textures environments. On the other hand the noise of the image must be low, because the model process can measure this noise like sharp edges, in conclusion, this algorithm only can used for binary classification sharpened/blurred images and are not enough for project’s requirements.
2.1.2. Problems of Single variance model

There are some problems related with textures an noise when the variance is used as blur detector but exits more conceptual problems related with the way to analyse the images.

Variance is a 1-D value, the position of pixels has been lost; the variance of global image evaluate all area of the image, an often some parts are sharpened and others are blurred. The main problem of the model is how to save 2-D information and how to analyse the different regions of the image.

As the first solution is to analyse only the part of the image related to the edge mask, avoiding flat and blurred regions, apparently works except when the main object is out of focus and the background is focused. In figure 15, the background focused (right) get a better result than plants focused and background blurred (left).

These type of problems do not meet the requirements of images with faces, where the face must be appropriately focused. The solution must detect the state of blur in hot areas, giving more importance to the purpose of the photo than to the context or background.

One solution for both 2D and regions analysis is to mesh the image in small pictures and analyse each one separately and later calculate the global value based on the parts. The mesh strategy increases complexity and make difficult evaluate the blur level of the image, but can be implemented and used for face analyse putting one block per face.
In figure 16 can be observed the trend of decreasing the sigma of the pixels (root of the variance), on top of image the same block with different level of blur is analysed (left-right with more blur), on the bottom of the histogram of each block (behind-front) show that blur block are less wide have a lower variance when the block in analysis is increasing its blur.

Finally variance is an easy/simple parameter for detect blur within its limitations, however, there are other parameters that allow us to discern some more specific types of blur without having the defects of the variance. As a conclusion, the model of multiple estimators should be implemented to discern false positives and improve detection.

### 2.1.3. Feature Patch Concept

In general, the problem of previous solutions is that is difficult to make something generic with a specific tool. In order to solve this problem in a generic form and getting the specific result, a modular software with different implementations is needed.

The first modules should prepare the image for the measurements, the second perform and combine multiples blur measurements, the third analyse the results and decide the value of blur obtaining a 2-D image with results per pixel and finally a higher specific modules can analyse the face areas of the image based on blur/sharp results image. This solution (Fig.17) allows one form to make multiples measures by pixel, keeping 2-D information and evaluating each area of the image at pixel level. The first and second modules are already implemented in the former solutions, while the third modules are a more generic improvement on 2.1.1 and 2.1.2 solutions.

The main problem of this solution is the need of greater amount of resources (multiple estimators) together with the requirement to use machine learning to obtain a reliable decision maker to analyse the multi-feature input pixel.
Patch concept is where an image of NxM pixels is transformed to A*A x N*M pixels, where A*A is the number of neighbours pixels (patch around one specific pixel). The patch pixels are in a vertical vector, one vector for each original pixel. These vectors are all concatenated forming a matrix of N*M columns, where each column is the original pixel and the neighbours pixels of the patch.

This form allows to get all neighbours pixels in a vertical vector, another advantage is the fact of realise the operation measurement as vector operation. As all pixels have their own column, it allows to use GPU parallel processing like cuda to solve quickly the mathematical measure calculation.

Fig. 17 Modular software structure.

Fig. 18 Patch and features process transformation.
Finally, can be performed a multiple measures over the same matrix, getting one result per pixel and estimator. Thus the final result is a row vector with the measurement for each pixel that can be grouped forming a matrix result.

If the blur estimator does not need the 2-D neighbour’s information, the processed image can directly transform image results to vector. When all estimators-features results are finished in a row vector, they are concatenated obtaining a matrix with one feature per column and features pixel results per row (FxN*M dimensions).

All features results are vertical vectors facilitating the training algorithm is made by pixel level. The result of this complex form to process the data is that the size of the image is independent, does not affect the process of analysing. Only its size should be limited by RAM or GPU resources limitations to be processed in parallel. To chop and process the image in parts it is a simple solution for resource limitation and easy to implement in cloud current systems.

### 2.1.4. Improve results based on features or scale

The efficacy of detect blur depends on resource spend as independent features, the size of the database for training and the light and noise image’s circumstances.

It has been taken as a basis that the illumination and the amount of noise of the images to analyse, is within an acceptable range for the algorithm. Therefore, the robustness and efficiency of software are strongly related to the training data set, however, beyond a certain training limit, the algorithm cannot progress if it does not have enough estimators or have a correct ground truth.

On the other hand, certain areas views appear to be diffuse in small scale but are sharp on larger scales. To avoid this kind of problems it is necessary to make an analysis at different scales of the image and combine the results.

The conclusion is that as long as resources permit, the maximum of features should be implemented to detect the blur, as well as to perform the training at multiple scales to avoid errors by large or small resolutions.

### 2.2. Features

This section explains the different features obtained from different blur estimators as input to train and decide the level of blur in each pixel of the image. All features presented in this chapter have been implemented in the final solution.

In order to explain with a visual comprehension, some images have been processed per each estimator, showing how the blur is detected and quantified by the estimator. These images appear in raw data, normalised to 0-255 values and combined with the original grey image to highlight the results.
Figure 19 shows two test images, “dragon-fly” that shows a clear sharpened object over blur background and “row of beers” which shows an increasing blur degradation by depth of field concatenated with the optic used (focused).

The gradient filter, extract edge information for edge-statistics estimators:

2.2.1. Feature 1: Variance Estimator

As the previous solution shows, the variance of edge is a measure of the blur, using the patch concept with AxA kernel size.

\[
(I_{edge})_{NxM} \xrightarrow{Patch\ transf} (I_{Patch})_{A+AxN+M}
\]
For each column of $I_{patch}$ variance calculated as:

$$Var(X) = E[(X - \mu)^2] = \sum \frac{(x - \bar{x})^2}{A^2}$$  \hspace{1cm} (5)

The areas with more blur have low values, i.e. dark pixels, sharped areas have high values, i.e. near white and grey areas are flat, textures or blurred areas.

Fig. 22 Variance (left), Normalised Variance (centre), grey-variance combination (right).

The variance image is the numeric value measured, the normalised image is scaled to 0-255 image grey level and the combinational image show 60% feature 40% original image in order see the most sharpened areas detected in the image.

2.2.2. Feature 2: Variance rate

Similar to Point 2.1.1 and using patch concept the variance change rate is measure over two different Gaussian-blur level, sigma equal 1.2 and 2.1.

Fig. 23 Variance changes blurring image.

This type of feature detects the sharped areas in black, opposite to previous features, in order to get the rate change of edges’ variance. The variance of each
patch of blurred images is compared with original edges’ variance. Finally the pixel’s feature value is the square root of quotient between rates of changes of blurred images:

\[
RVR = \sqrt{\frac{\varphi_{2.1}}{\varphi_{1.2}}} \quad (6)
\]

RVR (Root Variance Rate), where \( \varphi \) is the variance rate of blurred Gaussian images explained in 2.1 and 2.2 obtained empirically. As shows Figure 24 the dark areas are sharped regions, bright regions are blurred or unchanged areas.

Fig. 24 RVR feature (left), normalised (centre), combinations RVR-original image.

### 2.2.3. Feature 3: Log Kurtosis

When edges are sharpened, drastic changes between pixels values are detected as higher variance values, however, the variance of a texture, noise or specific patron can increase variance’s value without sharpened edges.

In statistics, there are different parameters that model the function’s histogram behaviour, the most used are the mean and standard deviation. The mean only show the average value without knowledge about how the values are distributed, the standard deviation (\( \sigma \)) only indicates how scattered or distant are those values in average, but there other related with skew, peak, multiples maxims etc.

![Different variance and skew forms histograms.](image)

Fig. 25 Different variance and skew forms histograms.

The mean of a probability distribution is also known as the expectation or first raw moment \( \mu = E[X] \) (7), the central moment related how far are the values of the
centre $\mu_n = E[(X - \mu)^n]$ (8), $n=2$, is the variance. The standard deviation of power $k$ is defined as $\sigma^k = \left(\sqrt{E[(X - \mu)^2]}\right)^k$ (9), for $K=1$ is the command dispersion of set values, for $K=2$ is the variance. One form to dimensionless the values and only analyse the form is “the normalised central moment” or “standardized moment” is defined as $\bar{\mu}_n = \frac{\mu_n}{\sigma^n} = \frac{E[(X - \mu)^n]}{E[(X - \mu)^2]^{n/2}}$ (10).

The first standardized moment ($\bar{\mu}_1$) is zero ($\mu - \mu$), the second standardized moment ($\bar{\mu}_2$) is one ($\frac{\sigma^2}{\sigma^2}$), the third standardized moment ($\bar{\mu}_3 = skew$) is a measure of skewness of set values and the fourth standardized moment ($\bar{\mu}_4 = Kurth$) is a measure of peakedness.

**Fig. 26** Same variance different form, Kurtosis estimator (left), Kurtosis measure (right).

Different form of set values can have the same variance and different kurtosis. The sharpened areas have peaked responses histogram, this fact can help detecting false results and improving variance estimator. Traditionally kurtosis is used as $Kth = \bar{\mu}_4 - 3$ getting values centred on zero, for our case no negative numbers are desired, using directly fourth standardized moment.

**Fig. 27** Kurtosis result (top), Log-Kurtosis (bottom), x axis (left), y axis (right), Row of Beer test image.

\[
(I_{edge})_{N\times M} \xrightarrow{\text{Patch trans}} (I_{Patch})_{A\times AxN\times M}
\]
For each column of $I_{\text{Patch}}$ kurtosis is calculated as:

$$Kurtosis = \mu_4 = \frac{\mu_4}{\sigma^4} = \frac{E[(X - \mu)^4]}{\text{var}^2} = \frac{\text{mean}(\text{NormSquared}^2)}{\text{mean}(\text{NormSquared})^2}$$ (11)

$$\text{NormSquared} = (X - \mu)^2$$ (12)

Kurtosis is evaluated for each axis using X and Y Sobel’s derivatives filters results. The most blurred areas get smaller values of kurtosis, dark areas, well sharped areas are near white value.

![Image](image1.png)

Fig. 28 Kurtosis result (top), Log-Kurtosis (bottom), x axis (left), y axis (right), Dragon-Fly image test

On other hand, in order to have a better measure of sharpness, the minimum X and Y measurements are put on a logarithmic scale to use as feature that helps variance measures.

![Image](image2.png)

Fig. 29 Kurtosis (left), Log-Kurtosis (centre), grey-log-kurtosis combination (right).
Variance estimator is strongly related with edges areas, but kurtosis only appears where edges’ areas do not form part of textures (view Fig 29).

2.2.4. Feature 4 Power Spectrum trends

Natural images have a similar frequency slope when power spectrum is analysed in a log scale. Due to the low-pass filtering characteristic of a blurred region, some high frequency components are lost and the amplitude spectrum slope of a blurred region trends to be steeper. Blur attenuates high frequency components and therefore makes the power spectra fall off much faster than its sharp counterpart.

In David J.Field’s publication [17], explain the bases of the behaviour of natural images in power spectrum context. Computing the power spectrum of an image \( I \) with size \( A \times A \) by taking the squared magnitude after Discrete Fourier transform (DFT), where \( I(u, v) \) denotes the Fourier transformed image. Can be expressed the two-dimensional frequency in polar coordinates, \( u = f \cos \theta \) and \( v = f \sin \theta \) and construct \( S(f, \theta) \).

\[
S(u, v) = \frac{1}{A^2} |I(u, v)| \rightarrow S(f, \theta) \quad (13)
\]

\[
S(f) = \sum_{\theta} S(f, \theta) \approx \frac{K}{f^{-\alpha}} \quad (14)
\]

Where \( K \) is an amplitude scaling factor for each orientation and \( \alpha \) is the frequency exponent denominated slope of power spectrum. For natural images \( \alpha \approx 2 \), in blurred images has large values between 2 and 10, i.e. lower amplitude in power spectrum.

In [18] a similar technique based on average power spectrum was develop by Liu Xu in radians where average term \( n \), is the number of different values of \( \theta \), the code implemented in c was found and incorporated in to the project.

\[
J(\omega) = \frac{1}{n} \sum_{\omega \theta} S(\omega, \theta) \approx \frac{K}{\omega^{-\alpha}} \quad (15)
\]

Fig. 30 Liu Xu, average power spectrum results [18], blue clear image, red blurred image.
In figure 30, can be observed the trend of blurred images (red) with a higher negative slope than clear images blue. This new feature analyse the patch image of each pixel getting a value of average power spectrum in log scale of neighbours pixels.

\[
(\text{I}_{\text{original}})_{N \times M} \xrightarrow{\text{Patch transf}} (\text{I}_{\text{Patch}})_{A \times N \times M}
\]

For each column of \(I_{\text{Patch}}\) average spectrum power is calculated as:

\[
\text{AveSpecPower} = \sum_{\omega} \log(J(\omega))
\]  

(16)

Fig. 31 Log Average Spectrum Power (left), grey-spectrum feature combination (right).

As show results, natural regions are clearer than the blurred (grey areas), or artificial regions (black areas).

2.2.5. Feature 5, 6, 7: Local Linear Filters

Another form to detect blur is using spatial filters such as Gabor [19] and Laplacian. They capture local band-pass or high-pass information that supplements frequency and gradient domain features in order to detect specific patterns. Gabor functions are a biological mechanism found in the visual cortex of mammalians, especially in primates to detect or discern particularly textures.

Gabor’s filters are linear filters directly related with the same mechanism of human visual system (a large explanation in [20]). Based on a dataset images and ground-truth labels, blurred and unblurred patches can be analysed obtaining linearly independent filters to detect unblurred patches.
The Image & Visual computing lab of University of Hong Kong [21] allows to download in some projects, the Gabor’s linear filters results after long learning process over one million of blurred and unblurred images. These filters are denoted as the generalized eigenvectors of eigenvalues which characterizes the clear areas. The linear filters result are provided in 11, 15 and 21 kernel’s size.

![Image of linear filters](image)

**Fig. 32** First 11 linear filters (top) [21], matrix of 46 eigenvectors (columns) compose by 121 coefficients (rows) in real value (left, bottom) and in normalized 0-255 (right, bottom).

Per each scale, the results are 46 or more eigenvectors (where each one is a linear filter). These filters are non-intuitive and get edge-spatial bright results over sharped areas and dark pixels over blurred areas as figure 32 show.

![Image of test image and linear filters](image)

**Fig. 33** Test image and first 29 linear filter’s results.

The filters (eigenvectors) are given in eigenvalues order and the results’ source explains that the first five filters are most independent with gradient-based statistical measures like the previous features. As conclusion first, second and fourth eigenvector results are selected as feature 5, 6 and 7.

\[ f_{5,6,7} = I_{\text{original}} \ast F_{\text{gabor}} \]
2.2.6. Feature 8 and 9: Colour Saturation

In [22] and [23] blur has been related to low colour saturation, thus in order to use this relation as blur feature, the minimum value of RGB (Red, Green, Blue values) normalised by colour’s saturation should be calculated for each pixel.

\[
S_{\text{pixel}} = 1 - \frac{3}{R + G + B} \cdot \min(R, G, B) \quad (18)
\]

Due to the different types of lighting, saturation intensity must be normalized using the highest value in the image.

\[
F_8 = \frac{S_{\text{pixel}} - S_{\text{max}}}{S_{\text{max}}} \quad (19)
\]

For feature 8 saturation intensity image is enough but for feature 9 patch concept is used in order to get a mask of sharpened areas and blur levels:

\[
(I_{\text{saturation}})_{N \times M} \xrightarrow{\text{Patch trans}} (I_{\text{Patch}})_{A \times AXN \times M}
\]
For each column of $I_{patch}$ saturation intensity patch is calculated as:

$$F_9 = \frac{\max [S_{pixel}]_{patch} - S_{max}}{S_{max}}$$

(20)

2.3. Importance of features

Although many features have been implemented (9 features), all of them can be catalogued in 4 main types. The first statistical methods variance and kurtosis that detect the clear areas or textures, the second local learned linear filter that detects main unblurred patrons, third the spectrum features, that in a different form detect different levels of the blur as statistic features and finally extra-helper features.

![Fig. 36 Dragon-fly test image (left), multi features combination image (right) based on first three important types of features.](image)

These extra-helped features are not key features but help given extra information to improve the results. An example is to use more eigenvectors results as linear filters creating new features where the most part of the information is overlapping with the second type but other part it is new information. Other form to get extra features is the use of colour intensity features, the rate of change of variance and autocorrelations or entropy features.

Most parts of extra-features like entropy or autocorrelation have not been implemented due to the possible scarce new information and the high computational resources consumption. The RAM and time required per each image analysis is a compromise that does not allow to add more features than the ones indicated.
As shown in figures 36 and 37, focus, depth of field blur, motion blur, flat or texture areas appear with dark values while sharpened areas are bright areas.

Finally, the original test images with the 9 features results of features filters are shown in Fig. 38 as each feature gives some context to blur/sharp areas.

### 2.4. Ambiguous threshold

Once the features images have been calculated, the blurred and sharpened areas must be defined.
How to decide if the area is or is not diffused and the degree of it? On base to Fig. 37 and Fig. 38 a grey-image with the sum of different features can be made as blur level image result (Fig. 39). On the other hand, this image doesn’t define with notoriety the different regions and the value of them. The process of binarizing and dilate can be one option to get more defined areas but it is only a gross approach.

Finally, a difficult multi-threshold based on multi-features should be implemented to decide the blurred or unblurred regions. The complex threshold is a difficult task to perform and practically impossible to optimize by a person. The solution is the use of machine learning where a set of data and the ground-truth with a classifier algorithm that defines the threshold to minimize the error on the training data set. If the database is sufficiently large and varied, the decision algorithm should be generic and robust.

Naïve Bayes classifier is a simple probabilistic classifier based on Bayes’s theorem [24] commonly used for text categorization or medical diagnosis, implemented on most math software. On the other hand, neural networks [25] allows to classifier data on base a large data set with the same mechanism than our brain and also is implemented in most software.

2.5. Data Base

In order to satisfy a large database, some blurred images from different public database have been included mainly from Hong Kong university (1000 labelled images) [21], UPC GTAV face database [26] (unlabelled images) and atresmedia web base images (atresplayer [27]).
To simplify and generalize the process, the images have been quickly and manually catalogued, with errors of 15-25% in order to speed up and give robustness to the database. Also, flats region can be catalogued as sharp that decrease the analytic performance of accuracy but generalise the learning.

Finally, a dataset of 1250 images have been used, 1000 for training, 200 for the test (mainly related with faces) and second test based on focus blur of 50 images. Our ground-truth labelled image is based on label sharped areas with 255 (white) value. The Hong Kong University’s database label blur areas. Therefore, it is necessary to obtain the negative image to adapt it to our system.

![Database subsets distribution for train and test.](image)

**Fig. 41** Hong Kong University’s blur database images and Gtav[26] faces and Atresmedia images[27] and web Hollywood images.

Our dataset is centred on faces or tv-programs in order to learn and test our case of study measuring the real performance on key frames.

![Database subsets distribution for train and test.](image)

**Fig. 42.** Database subsets distribution for train and test.

### 2.6. Scale join prediction

In the introduction of the bases of blur (1.3.2), the problem of textures and low details resolution was described to know the problems to identify a region too far in the picture or too near with textures. Almost the image noise that interfered
with the results creating false detection can be reduced using higher scales averaging the original noise.

In order to solve the know problems and improve the level details of the algorithm, all features have been processed with 3 different path sizes of 11x11, 15x15, 21x21 pixels and both algorithm Naïve Bayes and Neural Network were trained on the different scales.

Two different strategies have been implemented one easy combination of each scale named multi-scale zero that give a good grey sharp level to measure and other multi-scale based on iterative interference approximation to get a better binary result.

\[
Multiscale_{zero} = \frac{R_{11} + R_{15} + R_{21}}{3} \tag{21}
\]

The multi-scale feature function found with the linear filter (point 2.2.5) is the best appropriated function to join the 3 results as iterative interference join. The model based on scale ambiguity of studies [32] [33] that use the Loopy belief propagation for approximate inference [34] to fuse information from different level to get a better solution.

Reference [21] explains: “Specifically, a blur response \( b_i^s \) is calculated on the patch centred at pixel \( i \) at a particular scale \( s \). The model connects the blur score of each pixel with those of the surrounding pixels. Inter-scale correlation is also built among patches centred at the same corresponding pixel in different levels.

![Fig. 43 Multi-scale graphical model [21]](image)

Given local blur response \( \{\tilde{b}_i^s\} \) in each scale \( s \) and for each pixel \( i \), the total energy on the graphical model is expressed as:

\[
E(b) = \sum_{s=1}^{3} \sum_i \left| \tilde{b}_i^s - \tilde{b}_i^{s+1} \right| + \alpha \sum_{s=1}^{3} \sum_i \sum_{j \in \mathcal{N}_i} \left| b_i^s - b_j^s \right| + \beta \sum_{s=1}^{2} \sum_i \left| b_i^s - b_i^{s+1} \right| \tag{22}
\]

Where \( b_i^s \) is the score that we need to infer for each pixel. The first data term is unary to preserve the overall feature structure in image space. The second term is the spatial affinity, where \( \mathcal{N}_i^s \) is the four-neighbor set for pixel ‘i’ in scale ‘s’.
The last term is the inter-scale affinity, which bridges feature responses in different levels $b_i^s$ and $b_i^{s+1}$ have the same center pixel in two scales $\alpha$ and $\beta$ are weights. All the terms in Eq. (22) use the $l_1$ norm distance for robust inference.

Eq. (22) can be optimized via loopy belief propagation [34]. It starts from an initial set of propagation messages, and then iterates through each node by applying message passing until convergence. The final blur response map in the top layer is our result. 

\[ \text{Multiscale iterative} = MSfunction(R_{11}, R_{15}, R_{21}, \alpha) \] 

where has been use $\alpha = 0.5$ and $R_x$ result scale $x$.

Finally an easy adaptation Eq.(23) of the iterative function which allows to get the best multi-scale interference result has been implemented with our results getting the improvement mentioned in the publication [21].

2.7. Analysis of results

As the groundtruth is not perfectly defined and it is possible to find blur zones within sharp regions, but it is not so common to find sharp areas in the blur label region, the following estimators have been defined capable of measuring the efficiency of the algorithm for its comparative. The most extended use to compare binary classification is precision, recall and accuracy.

The pattern recognition precision is defined as the number of true positives over the number of true positives plus the number of false positives, i.e. is the fraction of retrieved true result that are relevant in the prediction. Also, is called positive predictive value (PPV), note that the meaning and usage of "precision" in the field of information retrieval differs from the definition of accuracy and precision within other branches of science and technology.

Recall is defined as the number of true positives over the number of true positives plus the number of false negatives, i.e. is the fraction of relevant true results that are successfully retrieved, in binary classification is also named as sensibility and it can be viewed as the probability that a relevant document is retrieved by the query or a measure of the proportion of positives that are correctly identified.

![Precision and Recall visual interpretation](image)

**Fig. 44** Precision and Recall visual interpretation.
The true negative rate or Specificity is the same definition of precision but the over the true negative results. It is a measure of the proportion of negatives that are correctly identified.

Accuracy is also used as a statistical measure of how well a binary classification test correctly identifies or excludes a condition. That is, the accuracy is the proportion of true results (both true positives and true negatives) among the total number of cases examined.

\[
\text{Precision} = \frac{\sum_i^n \{ p_i = t_i \land t_i = \text{true} \}}{\sum_i^n \{ t_i = \text{true} \}} = \frac{\sum \text{mask} \ast \text{truth}}{\sum \text{truth}} \tag{24}
\]

\[
\text{Recall} = \frac{\sum_i^n \{ p_i = t_i \land t_i = \text{true} \}}{\sum_i^n \{ p_i = \text{true} \}} = \frac{\sum \text{mask} \ast \text{truth}}{\sum \text{mask}} \tag{25}
\]

\[
\text{Specificity} = \frac{\sum_i^n \{ p_i = t_i \land t_i = \text{false} \}}{\sum_i^n \{ p_i = \text{false} \}} = \frac{\sum \text{invmask} \ast \text{truth}}{\sum \text{invmask}} \tag{26}
\]

\[
\text{Accuracy} = \frac{\sum_i^n \{ p_i = t_i \}}{N} = \frac{\sum \text{mask} \ast \text{truth} + \sum \text{invmask} \ast \text{truth}}{N} \tag{27}
\]

Where \( N \) are the pixels to be classified, \( p_i \) is the prediction for pixel \( i \) and \( t_i \) is the ground truth label for pixel \( i \), mask is the binarized image, truth the groundtruth image and invmask is the inverse image of mask all image in \([0,1]\) values.

Remember that our algorithm is focused on detecting sharp areas as white (1) and blur areas with dark value (0), most parts of other algorithm to compare are inverse, trying to detect blur areas and also some flat areas are labelled as sharp. The test subset has more blurred areas (background) than sharpened areas (faces), see more Annex A 1.5, these issues degrade the analytic performances getting values between 60-80 of accuracy.

The first analysis is based on visual comparison of results (subjective), the second analysis is based on performance and precision-recall curves. Finally, the third analysis is based on mean and variance behaviour precision, recall, accuracy and specificity of the algorithm over all dataset using an umbral of 0.5 in range 0-1 to binarize results. Also, the third analysis uses a pseudo precision, recall, specificity and accuracy based on the use of grey result instead of mask, i.e. without binarize the image getting a more qualitative measure of probabilities.

**2.8. Decision algorithms**

In order to get the decision thresholds, Matlab naïve Bayes and neural network implementation have been used as learning machine algorithms. Other methods like support vector machine (SVM) or decision trees have been tested with different results, most part of them with lower performance and RAM/CPU/Time problems associated (see more in Annex B point 1).
In order to test the training, 250 images from the database are not used during the training. Also, the neural training on Matlab has an automatic split of database finally the train of Naive Bayes subset has been split in 70% for the train, 15% for testing and 15% for the validation for Neural Network training.

### 2.8.1. Naïve Bayes

A simple and short explanation of Naive Bayes methods is a probabilistic model that create statistical thresholds. A representative vector of inputs (independent variables) \( X = (x_1, x_2, ..., x_n) \) and the respective desired output vector \( Y = (y_1, y_2, ..., y_n) \) are used to define the classes \( C_k \) of method. The probability for each \( K \) possible outcomes or classes \( p(C_k | x_1, x_2, ..., x_n) \) is formulated with conditional probability:

\[
p(C_k | X) = \frac{p(C_k) \cdot p(X | C_k)}{p(X)} \quad (28)
\]

understand as posterior = \( \frac{\text{prior} \times \text{likelihood}}{\text{evidence}} \)

The most extensive and clear example is man/woman classification (\( K=2 \), binary classification) with weight, height and foot size as feature input vectors. On base of input feature vectors, the statistics like mean and deviations are defined. The algorithm calculate the probability to be men/woman based on weight, height and foot size conditional probabilities. Resolve example:

\[
\text{posterior (male)} = \frac{p(\text{male}) \cdot p(\text{weight}|\text{male}) \cdot p(\text{height}|\text{male}) \cdot p(\text{foot size}|\text{male})}{\text{evidence}} \\
\text{evidence} = p(\text{male}) \cdot p(\text{weight}|\text{male}) \cdot p(\text{height}|\text{male}) \cdot p(\text{foot size}|\text{male}) + p(\text{female}) \cdot p(\text{weight}|\text{female}) \cdot p(\text{height}|\text{female}) \cdot p(\text{foot size}|\text{female})
\]

\[
p(\text{male}) = p(\text{female}) = 0.5
\]

Blur Naïve Bayes solution propose to classify each pixel as blurred/sharped pixel base on nine features (point 2.2) and data base inputs (point 2.5) to get a sharped/blurred mask regions.

Finally, the sharp/blur image information has been defined with different levels of blur using the probability of sharp \([0-1]\) as the pixel value and the prediction as mask value to separate blurred and sharpred regions.

\[
\text{Pixel}_\text{val} = \frac{P(\text{sharp}) \cdot \prod_{k=1}^{\text{features}} p(f_k|\text{sharp})}{P(\text{sharp}) \cdot \prod_{k=1}^{\text{features}} p(f_k|\text{sharp}) + P(\text{blur}) \cdot \prod_{k=1}^{\text{features}} p(f_k|\text{blur})} \quad (29)
\]

### 2.8.2. Naïve Bayes Implementation

In order to train the Naïve Bayes classifier, all the images have been analysed getting feature’s result in the form of nine columns (features) and N*M rows (pixels) storage in one file per image. This data has been stored in the three
scales (11, 15, 21 patch pixels size) obtaining three files per image. The same format process has been applied to the groundtruth label image. Finally, a total of 19 GBytes in 1250 files per scale has been generated for the Naïve Bayes analysis.

The learning algorithm of Matlab for Naïve Bayes Classifier (NBC) needs to run over one array with all data where each item has the nine features. This type of implementation requires to load all the data in a single array to run the NBC algorithm. Once the algorithm finished ‘nb’ object is returned with the boundaries limits, it allows to classify one input as blurred or sharped pixel and the probability of success. The main problem of generic Matlab implementation is that finished the training, the boundaries are fixed and cannot be improved with posterior training. Therefore the training must be implemented only one time (at least in Matlab) with high requirements of RAM.

A first approach based on files with 50, 100 and 250 features image-files concatenated was made to probe the idea that the performance of Naïve Bayes increase with the number of images. The Fig.45, shows the improvement of the algorithm base on 50 files is enough to classify the easy photos like “dragon-fly” image, another image test as “rows of beer” need a more extend database to stipulate with more details the level of blur in the image. Finally a very difficult image like “the cyclist on motion”, allows to seeing the improvement of the algorithm when the image has some different types of blur mixed with sharp regions.

![Fig. 45 Original, NBC of 50, 100, 250 files from left to right over test images.](image)

On the other hand, there are different types of blur (point 1.3.1), one form to improve detection and classification blur is to use different labelling’s (motion, focus and natural) to detect partially each type and then combine the result. This approach of detection and classification of blur has been studied in [39] (see more in Annex B point 2). The vast majority of images belong to focus blur, followed by motion blur and combinations of both together with natural blur allowing specific blur detection.
The Big features file (950 files) has a size of 17GB (344.5 million of pixels, with 9 features associated for each pixel), that means a minimum of 20-23 GB of RAM only for load the data and pics of 65 GB of RAM during the process of the algorithm. Experimentally, it has been observed that there are no significant differences in performance with training greater than 750 files. On the other hand, it has been detected that the combination of classifiers specialized in the two main types of blur (motion and focus) get equivalent result in performance to the use of a single big file training, but with a superior qualitative analysis.

Finally, the solution adopted for Naive Bayes is the generation of 4 dependent subsets based on the original 950 training images (Fig.46 and EQ. 30).

1. Motion blur subset, based on 400 files of 250 motion, 100 focus and 50 tv-face images.
2. Focus blur subset, based on 500 files of 350 focus, 100 motion, 50 tv-face images.
3. Face blur subset, based on 500 files of 100 focus and 400 mix of motion and focus blur images.
4. Bigfile subset based on 750 files of 100 face images, 250 motion images, 350 focus images.

![Fig. 46 Naive Bayes implementation of training per each scale.](image)

Once we get the four ‘nb’ classifiers per each scale, the Naïve Bayes solution for scale is the combination of first three and the max between combination and fourth result.

\[
I_{nb} = \max \left[ I_{nb_{motion}} + I_{nb_{focus}} + I_{nb_{motion}}, \quad I_{nb_{superfile750}} \right] \tag{30}
\]
As shown in Fig. 47 (test image Naive Bayes’ results) improve the approach solution of fig 38 and fig 45 where sharpened areas with a little margin are in white and blurred areas in black. On the other hand, the blur levels can be measured in a more homogeneous and precise way than previous solutions and get a clearer mask of blur and sharp areas.

Finally a study of Precision-Recall, Accuracy-Threshold was developing for each scale getting this comparative graph:

![Graph](image)

**Fig. 47 Naive Bayes Precision-Recall and Accuracy-Threshold analysis.**

### 2.8.3. Neural Network

A neural network [25] is a computational simulation approach to biological nervous system like the brain. A simple math model of a neuron is used to solve the problem in an intuitive form based on learning.

The idea is that digital neurons can learn minimizing the error of a known input-output database as the biological neuron learn and solve complex problems.

![Neural network diagram](image)

**Fig. 48 Biological and Digital Neural model [28].**
A Neuron is a biological processor entity based on four main parts: the dendrite that connects with other neurons (inputs), the weights or parameters that activate more or less each dendrite, the nucleus is a central part that allows to sum and process all dendrite signals by an activation function and the axon output signal connected to the next neuron’s dendrite (output). Changing the input weights, the output error can be minimized over the data.

One way to understand why a neural network works is thinking on the math model and the behaviour of the net of neurons. The activation function sometimes is an exponential or tangential or step function, so the result based on Eq.31 is a sum of simple math function over the input, one threshold activator with max and min saturation values (see Fig.49).

\[
o_j = \varphi \left( \sum_{j=1}^{n} w_{1j} x_j, \theta_j \right) \quad (31) \quad \text{where our case } \varphi_{\text{eliot}}(\text{net}, \theta) = \frac{\text{net} + \theta}{1 + |\text{net} + \theta|}
\]

A layer of multiple neurons connected to the input-vector generates one output that is the sum of their different contributions. The weights can be used to minimize the total error, average error or mean squared error under certain levels of error for a certain dataset. A more complex multilayer can be used, to increase the possibilities of the output and get a better error result, see Fig.50.

The result of the interaction and sum of simple functions produce a complex nonlinear function. The use of a large number of neurons allows to model any continuous function as the \textbf{universal approximation theorem} demonstrate [29][30]. In other words, when a neural network is ‘learning’, is iterating some
specific algorithm to get the partial derivative of the error respect each weight to minimize the total error over all data of the training in the iteration.

As conclusion changing weights and threshold, “learning process”, any nonlinear math function can be simulated. The number of neurons and the graph diagram of connections defines the precision of the network to approach the nonlinear function, the size of the database defines how of generic (near) is the solution to the real function. Finally, a neural network can be interpreted as a complex system that allows to define complex boundaries to classify the pixel making an approximation to the real function that separates the different output base on input.

The use of neural networks have two benefits over other methods like Naïve Bayes, first it allows to split the database in multiple training steps reducing resource’s pic of RAM and second it gets more specific boundaries for classification with the same database. On the other hand, neural networks are more complex to develop, can memorize results instead of learning. Other characteristics are that increasing the number of iteration or the complexity of the net graph the result can be improved but exponentially increases the CPU resources, the time needed to train the net and create a big probability of overfitting (memorize results).

2.8.4. Neural Network Implementation

The first approach of a neural network was made on 3 layers of neurons, a first layer with a high number of neurons to process the input, a second layer to reduce the outputs and finish the process and the output of the net with one neuron.

The conclusion of experimental tests are that the use of more layers helps to direct the process or the intelligence minimizing errors and noise. If a neuron is not really necessary and does not contribute to the system its coefficient tends to be ignored (multiplied by zero). On the other hand, it has been experimentally detected that those solutions that decrease the number of neurons in each layer allow to obtain solutions more precise (low noise) with a less use of resources.

For image processing, it can be understood that one horizontal pyramid layer where the number of neurons progressively decreases, acts in a similar way to multiple filters that extract and contrast the pertinent information to finally process the output based on summary and concrete information. This type of behaviour is similar to the Lateral Geniculate Nucleus (LGN) in the thalamus of the brain base on multiple layers which is responsible for the first processing of visual information to extract and helps the visual system focus its attention on the most important information.

One of the most important parts in the training of neural networks is the algorithm used, how the data is used for training, number of iterations of each train and how the features are mixed. Some of this tuning work is summarised in Annex B. 3.
**Training protocol**

The training has been made using 70% data to train, 15% to test and 15% for validations, all based on Matlab implementation algorithm. The network trains until get the maximum number of iteration (epochs) or gets a gradient error over validation set less than 1*10E-6. All structures are feed-forward multilayer neural networks, consecutive full connected layers with previous and next layer.

In order to get the better solution in the minimum time, multiple neural networks have been developed and tested with some small subset in parallel over different machine hardware. After the first fit over all database of training, all solutions are evaluated by hand (visual analysis) using 5-15 test images. The best solutions have been the base for the second generation of networks, sometimes retraining the same network with different parameters like long batch iterations or using the same strategy with more neurons per layer or more layers, see Fig 51.

![Fig. 51 Strategy of networks schema solution.](image)

The first step to get the solution was developed with smaller subset of 10 million of pixels with low iteration number per subset around 50-100 iterations depending of the complexity of the net. The idea is to train with small subset to reduce the resources used, but trying to use low iterations to avoid overfitting. Approximately can be understood as try to obtain a similar result that 1 training of 100 million pixels with 1,000 iterations, using 10 subs-trainings of 10 million pixels with 100 iterations per subset. If the subset is not enough big or have much iteration networks only memorize and forget real knowledge of previous subsets trained. A more detailed explanation can be found on Annex B 3.

Finally, if the strategy and conditions are adequate the solution is the knowledge acquired by the network, common to the different similar subsets, that allows to detect the blurring in a generic way. This procedure has been used due to the long time and resources necessary to train the entire database on each network.

Before the first generation of schemas, the most appropriated strategies are used to create new net o retrain base on the same strategy but with more resources. This second generation has been trained with 50 million of pixel per subset and 200-1000 iterations over each subset. On first generation only trained on scale 11, the second generation procedure has been used for all scales. As result, only
a few structures that work in scale 11 can be useful for scale 15 and 21. Finally, a new evaluation of second generation networks discards non-useful nets Figure 52. See more in Annex B 3. The final neural networks properties used are:

<table>
<thead>
<tr>
<th>Name</th>
<th>Neurons/Layer</th>
<th>Structure</th>
<th>Iteration1</th>
<th>Iteration2</th>
<th>performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Net_11</td>
<td>106/6</td>
<td>[50,25,15,10,5,1]</td>
<td>50</td>
<td>800</td>
<td>0.7093</td>
</tr>
<tr>
<td>Net_11_2</td>
<td>134/5</td>
<td>[60,45,20,8,1]</td>
<td>70</td>
<td>1000</td>
<td>0.9393</td>
</tr>
<tr>
<td>Net_11_3</td>
<td>171/7</td>
<td>[80,40,20,15,10,5,1]</td>
<td>50</td>
<td>500</td>
<td>0.9474</td>
</tr>
<tr>
<td>Net_11_4</td>
<td>56/3</td>
<td>[50,5,1]</td>
<td>150</td>
<td>2000</td>
<td>0.8370</td>
</tr>
<tr>
<td>Net_15</td>
<td>106/6</td>
<td>[50,25,15,10,5,1]</td>
<td>50</td>
<td>800</td>
<td>0.8235</td>
</tr>
<tr>
<td>Net_15_2_1</td>
<td>96/4</td>
<td>[50,35,10,1]</td>
<td>200</td>
<td>1500</td>
<td>0.9657</td>
</tr>
<tr>
<td>Net_21</td>
<td>106/6</td>
<td>[50,25,15,10,5,1]</td>
<td>50</td>
<td>800</td>
<td>0.8401</td>
</tr>
<tr>
<td>Net_21_3</td>
<td>171/7</td>
<td>[80,40,20,15,10,5,1]</td>
<td>70</td>
<td>500</td>
<td>0.8588</td>
</tr>
</tbody>
</table>

Table 1 Neural Net Schema, Iteration training and error.

It is denoted a specific good detection for each net, some cases conservative, optimistic in sharpness, clear binarization or grey solution. Finally, has been chosen a combined solution that provides a better measure in grey-level image.

$$I_{net} = \frac{\sum_{j=1}^{N} I_{net,j}}{N} \quad (32)$$

As can be observed, the best structure in the different scales using the same training has been the pattern of 106 neurons in 6 layers, first prediction in Fig. 52. The network diagrams and schemas can be found in Annex B point 3.

Fig. 52 Original (first on left), nets results left to right, combination result (last right) for each scale, 11, 15, 21 up-down rows for two test images.
Finally, the neural network gets a better visual result and grey-level solution than Naïve Bayes over the same database, reducing the number of required resources and being easy to implement in other languages like C/C++/Java/Python saving the structure and coefficients. As negative aspects, it needs a much more elaborated planning, knowledge and strategies, as well as a significantly longer training time than Naïve Bayes. By other hand has more noise and “salt” effect result that result in a lower performance in the mask, i.e. lower values for performance, recall and accuracy than can be improved dilating and erasing.

Fig. 53 Dragon-fly and rows of beers test image. First row grey image, net scale 11,15,21, second rows net zero combination, multiscale combination, final result, mask result.

Fig. 54 Neural Net Precision-Recall, Accuracy –Threshold analysis.

One unexpected result was the different rate of convergence at different scales. The different patch size changes the value of input data and finally the partial derivative when the network is learning. The experimental result shows as with the same training parameters patch of 11x11 and 21x21 have a similar rate of convergence and similar quality results, therefore the patch size of 15x15 have problems to converge (Fig.53 third image of second row), needing more iterations and getting a lower performance quality results than Naïve Bayes. This issue
forced to analyse each scale solution as a single solution with different performance at different scales.

**Fig. 55 Final colour map sharp analysis of test image by neural network.**

### 2.8.5. Combinational Result

Finally, with a NBC and Neural Net result per each scale, there are two output per scale, one for nets results and other that is the combination of Naive Bayes and Neural network results. Also, there is two types of Scale join prediction that conclude with 4 possible results per image:

\[
I_{\text{comb scale}} = I_{\text{net scale}} + I_{\text{nb scale}} \quad (33)
\]

\[
I_{\text{Resultado}} = \{I_{\text{net iterative}}, I_{\text{net zero}}, I_{\text{comb iterative}}, I_{\text{comb zero}}\} \quad (34)
\]

### 2.9. Face analysis approach

Finally, once the Blur detector algorithm is a generic form to get the level and mask over a generic image, the most popular detector of faces, eyes, nose and smiles can be implemented to get the hot areas of the images.
Fig. 57 Top results grey-scale, colour analysis Atresmedia image, focused faces foreground, blurred faces background.

One possible and simple analysis is to assume that good photograms must have the hot areas in the 70-100 best sharps regions of the images, with more of 70% of the area as sharpened.

Fig. 58 Face, eyes, smile, nose analysis of Sharp.
One of the most important points when evaluating the sharpness of a face is a correct and level of sharpness of eyes and mouth. Photograms with eye or mouth fast movements, closed eyes or small smiles are not desirable.

Therefore, as previously indicated in section 1.3.2, it is not important that flat regions are classified as blurred. What is really important is to avoid false positives, i.e. a high precision and specialization in the level of sharpness of the regions of interest. As can be observed the algorithm meets the requirements, allowing clearly identify, eyes, nose and mouth in the photographs.

In Annex C point 2 a visual analyse of results can observe the tendency that those correct faces show similar levels of sharpness in hair, eyes mouth and beard. On the other hand, those images not so suitable but correct show a less sharpness in eyes and mouth than in hair and beards. Finally, if any eye is closed or not smiled the results are significantly smaller.

3. Result and Comparative

The result of previous publications, Chakrabarti (FFT) [35], Liu [36], Su [37], Shi[14] has been downloaded with the same database of 1000 images[21] labelled for training. The main problem to compare the algorithms is that these paper didn’t explain which images are for training and which images are for test or validation.

Therefore we have comparative results Precision-Recall extracted from [14] and the predictions or results on the same database of training and test, but we do not know how to differentiate them and an objective analysis cannot be made. A first visual comparison was performed and can be found in Annex C point 1. To get an objective comparison, the best method of previous publication has been download and compile and tested over our dataset of testing to compare.

Finally, a non-objective analysis over the results of the data training has been developed to compare the result assuming certain risks as possible disadvantages of our algorithm in Annex B point 4.

3.1. Multi-scale Results of my method

Over subset of Test1, mainly based on faces following mean results have been measured, using as mask threshold 0.5 in range [0,1] (binarize by round).

<table>
<thead>
<tr>
<th>Test1, th=0.5</th>
<th>P</th>
<th>P-g</th>
<th>R</th>
<th>R-g</th>
<th>S</th>
<th>S-g</th>
<th>A</th>
<th>A-g</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neural_zero</td>
<td>0.89</td>
<td>0.82</td>
<td>0.11</td>
<td>0.18</td>
<td>0.55</td>
<td>0.56</td>
<td>0.57</td>
<td>0.59</td>
</tr>
<tr>
<td>Combination_zero</td>
<td>0.86</td>
<td>0.77</td>
<td>0.28</td>
<td>0.32</td>
<td>0.58</td>
<td>0.59</td>
<td>0.62</td>
<td>0.63</td>
</tr>
<tr>
<td>Neural_iterative</td>
<td>0.84</td>
<td>0.72</td>
<td>0.30</td>
<td>0.32</td>
<td>0.59</td>
<td>0.59</td>
<td>0.63</td>
<td>0.62</td>
</tr>
<tr>
<td>Combination_iterative</td>
<td>0.81</td>
<td>0.75</td>
<td>0.46</td>
<td>0.47</td>
<td>0.63</td>
<td>0.63</td>
<td>0.68</td>
<td>0.66</td>
</tr>
</tbody>
</table>

Table 2 Multi-scale result Analysis.

The results have low accuracy due to the fact that groundtruth label all face, head and shoulders as sharp. However, the reality is that only the elements of interest,
eyes, ears, nose, mouth, hair and beard are truly classified as sharp. Measuring all face flat regions as an error.

On the other hand, it is necessary to remember that the same labelling contains a manual error from labelling with an excess of sharp labelled.

As can be seen, there is a problem with the Zero combination, the easy explanation is that zero combination is focused on grey-level of blur, therefore many pixel around 0.4-0.6 values are round to 0,1 and reduce the accuracy of the “binarization” but increase the qualitative result getting a sharp level.

### 3.2. Comparatives

As it is reflected in [14], Shi method is the current best method published based on the database [21], also its source code is available in the image processing department of the Hong Kong University. An evaluation of the dataset of Test1 was carried out by Shi method, obtaining the following mean results for a threshold of 0.5 in the range [0,1]:

<table>
<thead>
<tr>
<th>Test 1, thr=0.5</th>
<th>P</th>
<th>P-g</th>
<th>R</th>
<th>R-g</th>
<th>S</th>
<th>S-g</th>
<th>A</th>
<th>A-g</th>
</tr>
</thead>
<tbody>
<tr>
<td>My Comb. iterative</td>
<td>0.81</td>
<td>0.75</td>
<td>0.46</td>
<td>0.47</td>
<td>0.63</td>
<td>0.63</td>
<td>0.68</td>
<td>0.66</td>
</tr>
<tr>
<td>My Net Zero</td>
<td>0.89</td>
<td>0.82</td>
<td>0.11</td>
<td>0.18</td>
<td>0.55</td>
<td>0.56</td>
<td>0.57</td>
<td>0.59</td>
</tr>
<tr>
<td>Shi</td>
<td>0.66</td>
<td>0.63</td>
<td>0.80</td>
<td>0.76</td>
<td>0.67</td>
<td>0.68</td>
<td>0.67</td>
<td>0.66</td>
</tr>
</tbody>
</table>

Table 3 Test1 result for threshold 0.5, Shi and my combination iterative solution results.
The method proposed, improve the result over the test subset, respect other implementation like Shi, remember that the objective of the TFM is to get a blur analysis algorithm specialised for TV key frames, mainly based on faces. The precision and a conservative predictions are more important that the mean accuracy, avoiding generation of false positives. The visual analysis (see Annex C point 1) show a conservative trend in our method that can get lower accuracy on really blurred images.

Nevertheless, there is a second subset Test2 base only in 50 images, mainly focus blur detection that has been analyse for both methods getting mean results for threshold 0.5 in the range [0,1] (mask by round function).

<table>
<thead>
<tr>
<th>Test 2, thr=0.5</th>
<th>P</th>
<th>P-g</th>
<th>R</th>
<th>R-g</th>
<th>S</th>
<th>S-g</th>
<th>A</th>
<th>A-g</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>My Comb. iterative</strong></td>
<td>0.71</td>
<td>0.65</td>
<td>0.70</td>
<td>0.68</td>
<td>0.83</td>
<td>0.79</td>
<td>0.80</td>
<td>0.77</td>
</tr>
<tr>
<td><strong>My Net Zero</strong></td>
<td>0.80</td>
<td>0.68</td>
<td>0.33</td>
<td>0.36</td>
<td>0.73</td>
<td>0.71</td>
<td>0.74</td>
<td>0.72</td>
</tr>
<tr>
<td><strong>Shi</strong></td>
<td>0.53</td>
<td>0.51</td>
<td>0.89</td>
<td>0.85</td>
<td>0.89</td>
<td>0.86</td>
<td>0.67</td>
<td>0.66</td>
</tr>
</tbody>
</table>

Table 4 Test2 result for threshold 0.5, Shi and my combination iterative solution results.

As a conclusion, we can understand that the results based on Net Zero Eq.32 and Eq.21 result are better for measurements of levels of grey and get higher performance for the specific detection of blur in facial patterns. However, for a
more general use of the algorithm, the most appropriate solution is the combination Eq. 33 using the iterative method Eq. 23 between scales.

Finally, a large time-lag has been detected in the time necessary for the processing of an image between Shi and the algorithm propose. Based on 250 results of Shi and my algorithm using the same machine to evaluate blur, it took 15% less average time for processing each image. Taking into account that both algorithms share several features (kurtosis, linear filters and spectrum filter) and the structures such as multiscale iterative analysis, it is concluded that the Shi implementation is not optimized and does not use GPU resources adequately, besides using certain computational features more expensive than those proposed by my method. I conclude that my proposal is based on a greater number of features that are simpler from the point of view of computations, which in turn have been processed in a more segmented and optimized way.

![Comparative Result mask in difficult images for blur detection.](image1)

![Mask comparative analysis over face images using our combination iterative method.](image2)
The current conclusion is that our proposed method is the best according to the results but close to Shi[14] method, by other hand is a better proposal for especially face’s datasets like atresmedia[27] images.

Finally, there is a pre-paper from May of 2017 “Local Blur Mapping: Exploiting High-Level Semantics by Deep Neural Networks” [38] based on deep learning neural network, that cannot be compared due that code or results are not published yet. Although a visual analysis of results are better than those of this proposal see Annex C point 2.
4. CONCLUSIONS

This thesis proposes an algorithm to discern the blur areas of key images in order to automatize current manual work of selection in online web-media services like “atresplayer”, “mitele”, “rtve alacarta” or similar.

The proposed algorithm has better results than expected, fulfilling all the requirements and generating a useful base for other algorithms of higher level that analyse sharpness in order to evaluate in a better way the selected images for online platforms.

4.1. Algorithm conclusions

The obtained results are large enough to measure the level of blur and classify the areas as blurred or sharpened. Although the initial idea of the algorithm was simple, fast and low-cost, these type of algorithms cannot accomplish with the main requirements.

The implementation of multi-features analysis in 1.5D increase the quantity of work increasing the ram and CPU resources but with an acceptable timing of process due to parallel processing. Also, the process of detect blur is independent of the resolution of the image, allowing to split and distribute the work in different process and enabling one easy scalability in the cloud.

The results obtained are very close to the last state of art algorithms in these field inside the top 3 of generic blur/sharp detectors, also a summary paper of state-of-art, features used and the result obtained is in development to summarize and share knowledge obtained with this thesis.

Finally, the solution algorithm it not able to differentiate textures and flat zones with respect to diffuses regions, which it is an axiom accepted from the beginning of development. However, this disadvantage allows to obtain a great precision by differentiating and measuring types of eyes, smile, beard or hair that allow a more useful analysis for the selection of key frames. On the other hand, the results obtained are superior to the ones from recent publications. However, these results contain a slightly higher amount of homogeneous noise introduced by the neural network on the 15 scale that has not converged correctly.

4.2. Knowledge and personal conclusion

During the realization of this thesis, a great knowledge of image processing and machine learning have been obtained. The great variety and number of papers used and analysed have contributed to obtain a fairly current and generic vision of the state of the art and procedures in artificial vision field.

Personally, I feel fulfilled and satisfied of the milestones achieved in this master thesis, exceeding widely the objectives marked both of requirements as personal goals and learning by experimental test practical skills.
ACRONYMS

ACF  Autocorrelation Function
BFGS Quasi-Newton Backpropagation
DCT  Discrete Cosine Transform
DFT  Discrete Fourier Transform
DoF  Depth of Field
FR   Full Reference metrics
JNB  Just Noticeable Blur
LGN  Lateral Geniculate Nucleus
NBC  Naïve Bayes Classifier
NR   Non Reference metrics
PPV  Positive Predictive Value
PSF  Point Spread Function
RGB  Red, Green, Blue values
RR   Reduced Reference metrics
RVR  Root Variance Rate
SVM  Super vector machine
REFERENCES

[16] Xiaoyu Ma, Xiuhua Jiang, Xiaohua Lei, Hui Zhang, Ping Liu “A No-reference Image Blur Metric Based on Twopass Edge Analysis”, link (05-02-2017).
Conference on Pattern Recognition and Image Analysis, Estoril, Portugal, June 7-9, 2005


[25] Bohdan Macukow. “Neural Networks – State of Art, Brief History, Basic Models and Architecture”, Faculty of Applied Mathematics and Information Science, Warsaw University of Technology, Koszykowa 75, 00-662 Warsaw, Poland.


[27] Atresmedia Corporation, TV, audio-visual entertainment Spain link.

[28] Neural Network, theory and images from link and link2 (03-03-2017).


Annex A

This annex allows to understand the dynamics of work, planning, main elements of the software and the hardware used to obtain the results. Finally, a brief summary explains the path and the different types of files generated.

1.1. Schedule of Thesis

The realization of this thesis has been combined with various subjects and with a full working day. Due to unexpected problems as well as the delay of key elements for the realization of the same did postpone the original objectives of presentation February-May, to a later date. However, almost 80% of the conceptual work was done in early January.

The rate of work was defined by the free hours in the different stages of the thesis:

- Pre-matricula, May-July 2016 combined with php work (8h) and master classes, around 2-4 hours per week.
- Internship in Ugiat July-November 2016, combined with php work (8h), around 20-30 hours per week.
- November-January 2017, combines with php work (8h), around 10-15 hour per week.
- January-March 2017, Medical incident, prevents the advance (right arm completely plastered).
- March-April, work accumulated in php, few free hours around 5-10 hours per week).
- April-May 2017, new job, adjustment of schedules and planning, 5-10 hours per week mainly on simulation.
- May-June 2017, 10-20 hours per week.

Main step-hits of Project Schedule:

- May-July: Read documentation, papers and base ideas.
- July: OpenCV and C++ introduction with Ugiat Software.
- 20th of July – 10th August: Fist blur detector test in C++ QT IDE.
- 5th -25th August: Select variance methods, discard width methods.
- 20th - 30th August: Normalize, analyses and calibrate algorithm.
- 25th of August – 12th September: Rate of change analysis and test.
- 8th – 25th September: Patch concept and multiples features.
- 15th of September – 25th October: Software project Ugiat.
- 5th – 25th October test features implementations from reference papers.
- October-November the great majority of the time is devoted to another related project.
- 4th - 15th of December: Neural Networks analyses.
• 15th – 28th December, improve database and result.
• 20th of December – 30th January: Start to write Master Thesis Memory.
• 5th – 25th of January: Analyses results of NB.
• 5th – 25th of March: Analyses results of Neural Network.
• April: Neural Network strategies and pacification.
• 1st – 15th May: Neural Network multi-scale solution.
• 15th – 25th May Final join solution.
• June: End memory and result compilations.

1.2. Code

This section gives a summary perspective of the structure and the main folder to run the proposed algorithm over a different database.

1.2.1. Previous scripts

The first scripts bases on scale, get grey images, edges and variance are Oldcode, Rodrigo and AnalysisofResultados folders.

These folders contain different script to test concepts, blur digital image and results analysis to get patrons. They have common structure base on:

- Function folder, Scale, edge filters, interpolation, thresholds derivatives and concrete parameters functions are storage inside.
- Original Images folder or Test folder with image to analysis.
- Result folder with the analytical o array results.
- Source_dif_xx intermeddle folder where digital blurred image are storage to compare with original.
- Face, smile, eyes and other folder for concrete result or image to analysis.

1.2.2. Final Code

The final folder of script have some subfolder:

- ./feature: it is the place where the nine features functions are.
- ./IMG: it is the place where original images, ground truth and intermeddle results are storage.
- ./IMG/groundtrue: collection of PNG ground truth images.
- ./IMG/Original: collection of original images to analys.
- ./IMG/Results: intermeddle storage of results.
- ./IMG/Results/XX: storage place for image_file_feature where XX is the scale, an the name of file begins with F_originalname.mat.
- ./IMG/Results/total: storage folder of scales for the concatenated files for NBC processing.
- ./IMG/Results/total/XX: storage folder for Big files where XX is the scale of the file.
- ./IMG/Results/true: storage the scale folder for ground truth based on sharp.
- ./IMG/Results/true/xx storage folder for image_groundtruth.mat in the scale XX.
- ./newImageData: folder where new database generate edge-dilate solution to make easy the manual labelling of ground truth.
- ./Results: Storage folder for final results, bu scale, multi scale etc...
- ./Results/XX: Combinational result of NB and Net solutions for scale XX.
- ./Results/XX/net: Neural Network solution for scale XX.
- ./Results/XX/nb: Naïve Bayes solution for scale XX.
- ./Results/multiscale: storage the different multiscale solutions.
- ./Results/multiscale/net: storage multiscale iterative solution base on Neural scale solutions.
- ./Results/multiscale/net_zero: storage the simple multiscale join solution for Neural network scale solutions.
- ./Results/multiscale/total: storage the iterative combinational multiscale join solution.
- ./Results/multiscale/total_zero: storage the simple multiscale join solution for combinational scale solutions.
- ./Test: image test folder.
- ./Test/result: storage test solutions.
- ./UGM multiscale iterative functions.
  - Files nb_scale_training.mat, net_scale_trining.mat, storage of classification object by scale and trining.
  - Files Resultados_YYY, analysis result over YYY subset.
  - Learn_chineese_filter [14] filter from Hong Kong University.

Main script to run:
- CreationFeatures.m: analysis the image and generate feature files.
- CreationFeatures2.m: analysis the ground truth and generate true files.
- GenDatabase.m: generate edge-dilate files to label the images.
- Compactresult_feature.m: allows to concatenate the result in big files.
- Compactresults_true.m: allows to concatenate the true files in bigtruefile.
- multiScaleBlurInterference: Adaptation for iterative multiscale-join.
- NeuralNetScript.m & SuperNeuralNetScript.m: define and train neural networks.
- Retrain.m: allow to retrain neural object with new or the same data improving the convergence.
- TestimageNetXX.m: where XX is the scale allows to show the result of nets.
- ResultanalysisWithoutScale.m: allows to evaluate each independent solution per each scale.
- ResultanalysisScale.m: allows to evaluate the scale combination solution.
- AnalysisOtherResults.m: allows to analysis other paper results.
- DemoTestanalysis.m: allow to test the algorithm with test demo images.
1.3. **Hardware and simulation**

During the pre-analysis, simulations and training a high computational capacity was required, due to a large amount of data volume and the improvement of the results a basis of trial and error.

Mainly a laptop and computer have been used, later the university server was joined and finally, all the devices were used with the objectives to obtain the best results in the fastest way (parallel analysis and strategies).

- Main computer CPU AMD FX 8120 8 cores, 24 GB RAM, 256 +60 GB SSD, NVidia 720 4 GB.
- Main laptop, CPU i7-4750HQ, 8 cores, 8 GB RAM 256 GB SSD, NVidia 740 2 GB.
- Main Server Intel Xenon, 8 cores, 16 GB of RAM, 200 GB SSD.
- Secondary laptop, i5 6500U, 4 cores 8 GB of RAM, 256 GB SSD.
- Secondary computer, i5-4500U 4 cores, 12 GB RAM, NVidia Ge Force 720 2 GB.
- Temporal Cluster, 3 computers of CPU AMD FX 8350 8cores, 16/32 GB RAM, 256 GB SSD, NVidia 920 4 GB, share at nights.

![Computational Resources of the project.](image-url)
Its main use has been nocturnal although in the last stages of the project they remained in operation several weeks.

1.4. **Data Results**

In order to describe with greater clarity the different types of files generated, outputs, intermediates and classifiers a brief description has been made.

- **Features files**: are the files that contain the 9 columns one per features and one row per pixel from original image, the mean size is around 15-25Mbytes.
- **Truefiles**: are the files that contain the ground truth, one column and one row per pixel from original image with a mean size of 250-500 Kbytes.
- **Big features files**: there are multiples types depending of the number of files concatenated. Most typical 100,250,500 concatenated files one with all features files concatenated. Have Large size from 500 Mbytes – 19 Gbytes
- **Bigtruth files**: there multiples types depending of the number of files concatenated. Most typical 100,250,500 concatenated files one with all ground truth files concatenated. Acceptable size of hundreds of megabytes.
- **Classifications files**: are the learning classification boundaries for each scale base on big features files, big truth files analysis. The net object are one object with the model properties, training methods and init parameters, also has structure matrix that define the layer and coefficients and weight for the connection between neurons. Naïve Bayes object has statistic values that define the boundaries and allow to calculate the probability of belonging to a set.
- **Results files**: are png image in grey scale. Typically there is for each image by scales (11,15,21) for neural solution, Naïve Bayes solutions, combination. Also, there is in multiscale solution for zero approximations, iterative approximation, based on neural scale solution and combination for neural + Naïve Bayes.

1.5. **Data Base**

The database is formed by 1000 images from [21], where approximately 300 are mainly motion blur, 500 are mainly focuses blur and 200 are a mix of motion, focus and natural blur.

Also, 100 images from gtav[26] have been added, these images have a good focus of face over blurred or flat background.
Finally, 150 tv images from Hollywood and atresmedia[27], have been added, these images have a mix of all types of blur, with multiples faces on images and different backgrounds.

The total database has a 41.32% of pixel labelled as sharp, including manual error and classification as sharp the flat areas inside a focus object.

The Test 1 subset based mainly on face images has a 27.61% of pixel labelled as sharps including manual error and classification as sharp the flat areas inside a focus object.

The Test 2 subset based mainly on focus images has a 43.39% of pixel labelled as sharps including manual error and classification as sharp the flat areas inside a focus object.
Annex B

This annex explains certain decisions or strategies with better visualization of the data publish in the main document. Its objective is only to clear and explain the decisions taken.

1. Decision algorithms

A brief description of the proposed methods to be used and the reason for the use of this method in the project.

1.1. Naïve Bayes Classifier Algorithm

Naïve Bayes classifiers are a family of simple probabilistic classifiers based on applying Bayes’ theorem with strong (naïve) independence assumptions between the features. The Matlab implementation is easy to use and get a good result over the data set with fast and simple training, but need high resources of RAM.

1.2. K Means Clustering Algorithm

Kmeans clustering is a technique for finding similarity groups in a data, called clusters, is often called an unsupervised learning, thus you don’t have to label the data the algorithm does it automatically. This method is mining not optimised for classification.

1.3. Support Vector Machine Algorithm

SVM is a supervised machine learning algorithm that can be employed for both classification and regression purposes. The algorithm defines hyperplanes to separate intuitively the data point’s base on coordinates. The main problem is a very long time and high memory requirements to get performance similar to NBC or Neural Net over our problem.

1.4. Linear Regression and Logistic Regression

Based on statistics line, planes, curves and cumulative logistic distributions can define boundaries to classified and make regression. The main problem is a low accuracy or difficult to learn in our problem of training.

1.5. Nearest Neighbours

KNN is a non-parametric classification method that estimates the value of the probability density function or directly the a posteriori probability that an element x belongs to the class C from the information provided by the set of prototypes. This method is the third interested method to test, that finally cannot be tested by time reason during the thesis planning.
1.6. **Artificial Neural Networks**

Neural Network is a different approach that using multiples types of layers, connections and models of the artificial neural model together with back-propagation algorithm which reduce the error of the results, have hard training strategy.

1.7. **Random Forests and Decision Trees**

A decision tree is a decision support tool that uses a tree-like graph or model of decisions and their possible consequences, including chance-event outcomes, resource costs, and utility. As a method, it allows you to approach the problem in a structured and systematic way to arrive at a logical conclusion. The implementation of the training is not an easy way that together with height CPU requirements, long and complex training that crash matlab.

1.8. **Summary decisions**

<table>
<thead>
<tr>
<th>Name Algorithm</th>
<th>Problem</th>
<th>Status</th>
</tr>
</thead>
</table>
| Naïve Bayes                   | Matlab implementation | Implemented |%
| K Means Clustering            | Minering method   | Discarded   |
| Support Vector Machine        | Long timing       | Discarded   |
| Linerar, Logistic Regression  | Bad accuracy      | Discarded   |
| Nearest Neighbours            |                   |             |
| Neural Network                |                   |             |
| Random Forest, Decision Trees |                   |             |

Table 4 Learning Machine method summary.

1.1. **Visual comparative of learning methods**

Visual analysis of method from martin-thoma web blog ([link](#)) can be used to understand how different machine learning algorithm work.
2. Naïve Bayes Implementation

During the implementation of Naïve Bayes, multiple investigations have been carried out until the final solution is reached. The main problems encountered during the realization of the results are:

2.1. Creation of Big files

To perform the training of the algorithm it is necessary to have the samples in the correct format, grouped in a vector. Therefore to facilitate the task of loading data and reduce read times as well as memory management has been chosen to previously concatenate in a file the necessary information for further analysis.

Another element of interest is that the verification of the information is robust, checking that the lengths of ground truth and features are the same, as well as the elimination of format elements that affect the algorithms.

A script is defined that loads in memory progressively a number of files of features and ground truth, generating one output file with all features and all ground truth
responses and a list of errors if certain files have not passed the validations and they have been excluded.

2.2. Implementation of all database

One of the biggest problems has been the generation and training of the most massive files, the big files of each scale. Due to their size, it was necessary to use the combination and combination of several resources to obtain a computer with an effective capacity of 16-32 GBytes of ram and 100 GBytes of a 512 MB SSD disk used as SWAP.

Finally with these capacities has been achieved peaks of 52-38 GBytes of RAM to process of 1250 features, with slight or null improvements respecting classifiers based on 750-900 files. The final solution has been finely chosen by a Multi-classifier strategy that allows you to calculate different classifiers in a simpler way, combine their results and in some cases getting improvements over the use of a single classifier.

2.3. Detection of different types of blur

There are three main cases of photographs based on base data, focus blur, motion blur and a few of natural blur. As there is a great variety of each type can be made specific training with each type of blur.

The results are classifiers able to detect blur and to differentiate its origin to allow to apply the corrective algorithm. However, this type of training on a smaller subdata base gets worse blur detection. On the other hand, great amount of images have a combination of different types of blur, which generate multiple detections with different origin. In [39], a simpler method than the proposed one in this thesis has been done to realize a concrete analysis of each type of blur using Naive Bayes like algorithm of training.

As an interesting point, the use of a specialized database to detect flat zones and textures can help to declassify areas as blur and allowing improve the algorithm. However, given the burden of work on labelling this strategy has not been implemented.

Finally, multiple classifiers with a biased but not specialized detection have been proposed, in order to obtain a combined result that equals or improves to a single implementation of Naive Bayes.

In order to facilitate the processing of the base data and to have an incremental result, the Naïve Bayes training were carried out in different stages, skewed on a specific type of blur, with the aim of obtaining results equivalent to the total use of the database and use fewer resources. For each classifier the efficiency is evaluated in relation to the test database, obtaining in all cases very similar results and selecting the best case. It is explained on point 2.8.2 and Fig. 46.
2.4. Results by scale

The result obtained for umbral 0.5 in range [0,1] (round function to create mask), the same process of fig 47, for all test image get an average of test subset:

<table>
<thead>
<tr>
<th>Test1, thr=0.5</th>
<th>P</th>
<th>P-g</th>
<th>R</th>
<th>R-g</th>
<th>S</th>
<th>S-g</th>
<th>A</th>
<th>A-g</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scale 11</td>
<td>0.83</td>
<td>0.76</td>
<td>0.30</td>
<td>0.34</td>
<td>0.58</td>
<td>0.60</td>
<td>0.63</td>
<td>0.63</td>
</tr>
<tr>
<td>Scale 15</td>
<td>0.83</td>
<td>0.78</td>
<td>0.30</td>
<td>0.34</td>
<td>0.59</td>
<td>0.59</td>
<td>0.63</td>
<td>0.64</td>
</tr>
<tr>
<td>Scale 21</td>
<td>0.82</td>
<td>0.76</td>
<td>0.32</td>
<td>0.37</td>
<td>0.60</td>
<td>0.60</td>
<td>0.63</td>
<td>0.64</td>
</tr>
<tr>
<td>Combination</td>
<td>0.81</td>
<td>0.75</td>
<td>0.46</td>
<td>0.47</td>
<td>0.63</td>
<td>0.63</td>
<td>0.68</td>
<td>0.66</td>
</tr>
</tbody>
</table>

Table 5. Analysis Result of Naive Bayes per each scale.

Where P is precision, R is recall, S specificity, A is accuracy and the –g indicate the use of pseudo parameters based on grey level images.

3. Neural Network implementation

The realization of neural networks has been based on a trial, error and improvement test. Multiple solutions have been trained and tested in parallel. Finally, only a small set of final solutions have been selected and combined to generate the neuronal response.

3.1. Properties and tuning of Neural Nets

BFGS, Quasi-Newton Backpropagation [31] (secant methods) is an alternative to conjugate gradient methods for fast optimization that use less memory resources with little more time to converge. It is based on the basic step of Newton’s Methods where the Hessian matrix is update based on function on gradient to not spent high resources to compute real Hessian.

Neural model, the activation function of the neuron most used is tangential sigmoid, in our case was replaced with Elliot sigmoid which allows to compute its derivate more quickly. The last steps of the net should be implemented by activation function from [0, 1] like logarithm sigmoid but finally, the training was made over [-1, 1] (by performance reasons) and solver making a translation over ground truth and output predictions.

The Training was made in multiple steps by RAM limits constrains. Therefore is important to use a subset of the database with all the types of blur (good mix) and get enough files on the train to generalise the real solution. Under certain umbral of small subset or bad mix the algorithm doesn’t learn blur detection and try to memory or detect specific patrons of the subset. Divide the database in some subsets mean that some knowledge learn during the first subset are forgotten during the learning of other subsets. For this reason, the size and the mixture of the data are vital achieving homogeneous and incremental learning.

From the Point of view of the analysis of results without combination for all the data over each scale get a mean of:
Focus Detection and Sharpness Evaluation

Table 6 Network results combination per each scale.

<table>
<thead>
<tr>
<th>Test1, thr=0.5</th>
<th>P</th>
<th>P-g</th>
<th>R</th>
<th>R-g</th>
<th>S</th>
<th>S-g</th>
<th>A</th>
<th>A-g</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scale 11</td>
<td>0.85</td>
<td>0.81</td>
<td>0.16</td>
<td>0.20</td>
<td>0.56</td>
<td>0.57</td>
<td>0.59</td>
<td>0.60</td>
</tr>
<tr>
<td>Scale 15</td>
<td>0.83</td>
<td>0.78</td>
<td>0.30</td>
<td>0.35</td>
<td>0.59</td>
<td>0.60</td>
<td>0.63</td>
<td>0.64</td>
</tr>
<tr>
<td>Scale 21</td>
<td>0.82</td>
<td>0.76</td>
<td>0.32</td>
<td>0.37</td>
<td>0.60</td>
<td>0.60</td>
<td>0.63</td>
<td>0.64</td>
</tr>
</tbody>
</table>

Where P is precision, R is recall, S specificity, A is accuracy and the ‘−g’ indicate the use over the grey results.

3.2. **Visual representation of Neural structure**

Matlab view plot and python script plot of network connections.

Fig. 66 Network base structure of Net_11, Net_15, Net_21.
Fig. 67 Network base structure of Net_11_2

Fig. 68 Network base structure of Net_15_2_1

Fig. 69 Network base structure of Net_11_4.
4. Results Analysis

As indicated in section 3.2, in addition to an analysis of the two subsets of tests, the entire database has also been evaluated, including training elements. This non-objective analysis has been performed since the results of other algorithms are available for the whole database including training.

Although a comparison is not correct including the training subset as it may lead to erroneous conclusions, a cautious comparison has been made detailing possible misinterpretations.

4.1. **Final Scale results over Database include training**

Finally, a comparative between neural network results, Naïve Bayes result and a combination of Naïve Bayes and Neural network can be made for each scale using threshold for binarize of 0.5 in range [0,1] in all database.

<table>
<thead>
<tr>
<th>Patch 11</th>
<th>P</th>
<th>P-g</th>
<th>R</th>
<th>R-g</th>
<th>S</th>
<th>S-g</th>
<th>A</th>
<th>A-g</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neural</td>
<td>0.85</td>
<td>0.81</td>
<td>0.16</td>
<td>0.20</td>
<td>0.56</td>
<td>0.57</td>
<td>0.59</td>
<td>0.60</td>
</tr>
<tr>
<td>NBC</td>
<td>0.83</td>
<td>0.76</td>
<td>0.30</td>
<td>0.34</td>
<td>0.58</td>
<td>0.60</td>
<td>0.63</td>
<td>0.63</td>
</tr>
<tr>
<td>Combination</td>
<td>0.85</td>
<td>0.77</td>
<td>0.27</td>
<td>0.30</td>
<td>0.58</td>
<td>0.58</td>
<td>0.62</td>
<td>0.62</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Patch 15</th>
<th>P</th>
<th>P-g</th>
<th>R</th>
<th>R-g</th>
<th>S</th>
<th>S-g</th>
<th>A</th>
<th>A-g</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neural</td>
<td>0.83</td>
<td>0.78</td>
<td>0.30</td>
<td>0.35</td>
<td>0.59</td>
<td>0.60</td>
<td>0.63</td>
<td>0.64</td>
</tr>
<tr>
<td>NBC</td>
<td>0.83</td>
<td>0.78</td>
<td>0.30</td>
<td>0.34</td>
<td>0.59</td>
<td>0.59</td>
<td>0.63</td>
<td>0.64</td>
</tr>
<tr>
<td>Combination</td>
<td>0.85</td>
<td>0.79</td>
<td>0.25</td>
<td>0.26</td>
<td>0.58</td>
<td>0.58</td>
<td>0.62</td>
<td>0.62</td>
</tr>
</tbody>
</table>
Table 6 Scale Result Analysis over all database.

<table>
<thead>
<tr>
<th></th>
<th>P</th>
<th>P-g</th>
<th>R</th>
<th>R-g</th>
<th>S</th>
<th>S-g</th>
<th>A</th>
<th>A-g</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neural</td>
<td>0.82</td>
<td>0.76</td>
<td>0.32</td>
<td>0.37</td>
<td>0.60</td>
<td>0.60</td>
<td>0.63</td>
<td>0.64</td>
</tr>
<tr>
<td>NBC</td>
<td>0.82</td>
<td>0.76</td>
<td>0.32</td>
<td>0.37</td>
<td>0.60</td>
<td>0.60</td>
<td>0.63</td>
<td>0.64</td>
</tr>
<tr>
<td>Combination</td>
<td>0.84</td>
<td>0.77</td>
<td>0.31</td>
<td>0.35</td>
<td>0.59</td>
<td>0.60</td>
<td>0.63</td>
<td>0.64</td>
</tr>
</tbody>
</table>

As can be expected, similar values are obtained in some cases higher than the analysis on the subset of the test. This increase in results is understood as a certain conditioning for the training test or memorized of results. However, if the results as we can see are similar to the subset of tests they allow to make comparisons with the other methods without allowing to extract conclusions.

All result of the database has been analysed for threshold of 0.5 in range [0 , 1], round mask function, calculate the mean result over all subset, variance and scatter plots of the results.

Fig. 71 Precision – Recall Scatter plot, proposed algorithm thr=0.5.

Fig. 72 Precision-Specificity Scatter plot, proposed algorithm thr=0.5
The results allow to compare the behaviour over the entire database. In the great majority of cases, the proposed algorithm obtains better values than previous algorithms. However, conclusions cannot be drawn as this may be due to a greater memorization of the training subset. Also, it can be observed that the predictions are more limited and have a lower dispersion, obtaining lower variances in all cases.
Fig. 74 Su[37] Scatter P-R.

Fig. 75 Su[37] Scatter P-A.

Fig. 76 Chakrabarti [35] Scatter P-R.

Fig. 77 Chakrabarti[35] Scatter P-A.

Fig. 78 Liu [36] Scatter P-R.

Fig. 79 Liu[36] Scatter P-A.
Focus Detection and Sharpness Evaluation

Fig. 80 Shi [14] Scatter P-R.

Fig. 81 Shi[14] Scatter P-A.
1. Visual Comparative

Visual comparative where clear areas are black and blurred areas are white. Each row has original, groundtruth, combination zero, Chakrabarti[35], Liu[36], Shi[14], Su[37] results for visual comparative.
The best methods to measure grey-scale level are Su and our proposal, the best methods for mask areas are Liu, Shi and our iterative proposal (see fig 62).

2. Some images Results

The following results are predisposed in a first rows like Eq.34 with the original image, net zero result, net iterative result, combination zero result and combination iterative result. The second row is the colormap net solution.
Fig. 84 Result proposed algorithm test image

Fig. 85 Result proposed algorithm test image.
Fig. 86 Result proposed algorithm test image.

Fig. 87 Result proposed algorithm test image

Fig. 88 Result proposed algorithm test image
Fig. 89 Result proposed algorithm test image

Fig. 90 Result proposed algorithm test image

Fig. 91 Result proposed algorithm test image
Fig. 92 Result proposed algorithm test image

Fig. 93 Result proposed algorithm test image