Universidade Estadual de Campinas
Instituto de Computação

Paulo Henrique Hack de Jesus

Discriminative Features for Image Blur Detection

Características Discriminativas para Detecção de Borramento em Imagens

CAMPINAS
2018

Paulo Henrique Hack de Jesus

Discriminative Features for Image Blur Detection

Características Discriminativas para Detecção de Borramento em Imagens

Dissertação apresentada ao Instituto de Computação da Universidade Estadual de Campinas como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Thesis presented to the Institute of Computing of the University of Campinas in partial fulfillment of the requirements for the degree of Master in Computer Science.

Supervisor/Orientador: Prof. Dr. Hélio Pedrini

Este exemplar corresponde à versão final da Dissertação defendida por Paulo Henrique Hack de Jesus e orientada pelo Prof. Dr. Hélio Pedrini.

CAMPINAS
2018

Agência(s) de fomento e nº(s) de processo(s): CAPES
ORCID: https://orcid.org/0000-0003-1022-4924

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca do Instituto de Matemática, Estatística e Computação Científica
Ana Regina Machado - CRB 8/5467

J499d

Jesus, Paulo Henrique Hack de, 1990-
Discriminative features for image blur detection / Paulo Henrique Hack de Jesus. – Campinas, SP : [s.n.], 2018.

Orientador: Hélio Pedrini.
Dissertação (mestrado) – Universidade Estadual de Campinas, Instituto de Computação.

1. Processamento de imagens. 2. Visão por computador. 3. Classificação de imagem. I. Pedrini, Hélio, 1963-. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

Informações para Biblioteca Digital

Título em outro idioma: Características discriminativas para detecção de borramento em imagens
Palavras-chave em inglês:
Image processing
Computer vision
Image classification
Área de concentração: Ciência da Computação
Titulação: Mestre em Ciência da Computação
Banca examinadora:
Hélio Pedrini [Orientador]
Jorge Vicente Lopes da Silva
Guilherme Pimentel Telles
Data de defesa: 15-10-2018
Programa de Pós-Graduação: Ciência da Computação

Universidade Estadual de Campinas
Instituto de Computação

Paulo Henrique Hack de Jesus

Discriminative Features for Image Blur Detection

Características Discriminativas para Detecção de Borramento em Imagens

Banca Examinadora:

- Prof. Dr. Hélio Pedrini
  IC/UNICAMP
- Prof. Dr. Jorge Vicente Lopes da Silva
  CTI Renato Archer
- Prof. Dr. Guilherme Pimentel Telles
  IC/UNICAMP

A ata da defesa, assinada pelos membros da Comissão Examinadora, consta no SIGA/Sistema de Fluxo de Dissertação/Tese e na Secretaria do Programa da Unidade.

Campinas, 15 de outubro de 2018

Acknowledgements

First of all, I would like to thank my parents, Lori Hack de Jesus and Atilio Neves de Jesus, and my wife Talita Varela Utsuni de Camargo Jesus, for their incentive, patience and never-ending support. I am grateful to the Camera's team of Motorola Mobility LLC and the Eldorado Research Institute, for allowing me to publish part of the research we have developed during the execution of the projects, as well as for all support they gave me. I also thank CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior) and the Institute of Computing of the University of Campinas for the scholarship and support received from them. Last, but not least, I would like to thank my advisor, Professor Hélio Pedrini, for his guidance, patience and persistence.

Resumo

Graças, entre outros fatores, à grande capacidade de armazenamento das câmeras digitais, usuários atualmente são capazes de capturar uma grande quantidade de fotografias de uma mesma cena, esperando que, ao menos, uma delas apresente boa qualidade. Por outro lado, fabricantes das câmeras desejam testar seus sistemas pela emulação do que um usuário real faria e estimar a satisfação deste usuário. Um método automático para detectar imagens borradas pode beneficiar tanto os usuários finais quanto os fabricantes, por exemplo, ao filtrar as imagens borradas de um conjunto de fotografias e ao estimar a percentagem de imagens com possíveis problemas de foco em um conjunto de teste. Um dos desafios para esse tipo de aplicação é que algum nível de borramento nas imagens pode até ser desejável, como no caso de destacar uma pessoa do fundo (por exemplo, fotografias de perfil com o fundo fora de foco). Neste contexto, este trabalho propõe e avalia um método para extrair descritores específicos de imagens e treinar um modelo de aprendizado de máquina para categorizar as imagens em duas classes, borradas ou não borradas, tal como uma pessoa faria, ou seja, de acordo com quais porções estão borradas na imagem. Dois dos três conjuntos de descritores empregados, denominados de *Sharpness Behavior* (SHB) e *Fourier Transform Variance* (FTV), foram desenvolvidos neste trabalho de pesquisa, enquanto o terceiro conjunto utiliza o descritor *Histogram of Oriented Gradients* (HOG). Como uma base de dados de imagens rotuladas não estava disponível para este problema no período de desenvolvimento deste trabalho, uma nova base foi criada empregando-se técnicas de ampliação de dados, com mais de 14.000 imagens, para treinar e avaliar o nosso método. Resultados da classificação foram comparados com métodos da literatura para finalidades similares, já que não foi encontrado um método para a mesma aplicação. Os experimentos mostraram que o método proposto superou os demais por uma considerável margem. Além disso, uma possível combinação do método com um método de segmentação de imagens foi investigada, de forma a calcular os descritores apenas na região mais importante das imagens, o que melhorou os resultados de forma significativa.

Abstract

Thanks, among other factors, to the large storage capacity of digital cameras, users are currently able to take a large amount of photographs from the same scene, hoping that at least one of them will present good quality. On the other hand, camera manufacturers aim to test their systems by emulating what a real user would do and estimating user satisfaction. An automatic method for detecting blurred images can benefit end users as well as manufacturers, for example, by filtering blurred images from a set of photographs and estimating the percentage of images with possible focus problems in a test set. One of the challenges for this type of application is that some level of blurring in the images may even be desirable, as in the case of highlighting a person from the background (for instance, profile photographs with the background out of focus). In this context, this work proposes and evaluates a method to extract specific descriptors of images and to train a machine learning model to categorize images into two classes, blurred or not blurred, as a person would do, that is, according to which portions are blurred in the image. Two of the three sets of descriptors employed, named *Sharpness Behavior* (SHB) and *Fourier Transform Variance* (FTV), were developed in this research, whereas the third set used is the *Histogram of Oriented Gradients* (HOG) descriptor. Since a set of labeled images for this problem was not available during the development of this work, a new data set was created through data amplification techniques with more than 14,000 images to train and evaluate our method. Classification results were compared with literature methods for similar purposes, since no approach was found for the same application. The experiments demonstrated that the proposed method surpassed the others by a considerable margin. In addition, a possible combination of the approach with an image segmentation method was investigated, in order to calculate the descriptors only in the most important region of the images, which significantly improved the results.

List of Figures

List of Tables

List of Abbreviations

2D	Two-Dimensional
3D	Three-Dimensional
AUC	Area under Curve
BM	Blur Metric
CNN	Convolutional Neural Network
DCT	Discrete Cosine Transform
DoH	Determinant of Hessian
DSLR	Digital Single-Lens Reflex
FN	False Negative
FP	False Positive
FPR	False Positive Rate
FTV	Fourier Transform Variance
GIMP	GNU Image Manipulation Program
HOG	Histogram of Gradient
HSV	Hue-Saturation-Value
ISO	International Organization for Standardization
JPEG	Joint Photographic Experts Group
PNG	Portable Network Graphics
PSF	Point Spread Function
RGB	Red-Green-Blue
ROC	Receiver Operating Characteristic
ROI	Regions of Interest
SHB	Sharpness Behavior
SIEDS	Saturation Image Edge Difference Standard-Deviation
SLR	Single-Lens Reflex
SSIM	Structural Similarity
SVM	Support Vector Machine
TN	True Negative
TP	True Positive
UAV	Unmanned Aerial vehicles
YOLO	You Only Look Once

Contents

Chapter 1

Introduction

In this chapter, we describe the problem addressed in this research along with its motivation. Next, we outline the specific objectives we pursued and the limitations found. Then, the hypotheses that guided the development of our methodology are presented. Finally, we list the scientific contributions of this work and how this text is organized.

1.1 Problem and Motivation

Due to the high popularity of social media, easy-to-carry mobile devices equipped with digital cameras with cheap and large digital storage capability, the amount of photographs taken by non-professional people increases every day, as well as the demand for the manufacture of camera systems that are easy to use and also offer good quality images.

The quality of an image can be degraded by many different factors, such as hardware components (lenses, sensors), basic camera settings (exposure time, aperture, gain, speed), unstable camera, low light environment, motion, among others [23]. Blur is a common distortion that degrades the image quality, which is most often caused by movement of the camera or objects in the scene (called motion blur) and also by a wrong position of the lens in the focal system of the camera (called defocus blur). Blurred images lose detailed information and may be useless for many applications (for instance, from preserving important memories to security surveillance) [22].

Many camera users take an excessive number of pictures from the same scene in the hope that at least one of them will be very good. On the other side of the market chain, manufacturers need to test their camera systems to simulate their use in the real world, and then create a metric to assess the quality of their products and know how it is evolving over time. An automatic method for detecting blurred images would be of great benefit to both cases by filtering blurred images from a set of photos taken by the end user and quantifying the fraction of images with focus problems, for instance.

Traditional sharpness metrics, such as the derivative-based method, statistical method, have been studied for this purpose, but they work well only in a sequence of images of the same scene, since their values depend on the content of the images and, therefore, are not comparable between images of different scenes [22]. This property makes it impractical to use these methods for the applications mentioned above.

In addition, for these applications, some amount of blur may be desirable to highlight some specific object in the image, for instance, in the case of portraits with bokeh [41]. These images are considered to have good quality by users and, therefore, should not be perceived by a good classifier as blurred. For these reasons, the problem of classifying an image as blurred or sharp remains a complex task and has not been completely solved [48].

1.2 Hypotheses

Due to the subjectivity nature of the problem, the method for solving it must be adaptable and learn what is important by examples. That is why we chose to use Machine Learning (ML) models to learn the map between some features extracted from the images and their actual category.

Therefore, we hypothesized that an ML model can learn what is important from a set of labeled images, provided that the features can carry information capable of discriminating the content of the images, as well as being robust to partially blurred images and, thus, correctly classify the images as blurred or shar, in a similar way as the camera user would do.

1.3 Objectives and Limitations

Our goal with this research is to develop a method to automatically classify images as blurred or sharp in a way that it is closer to how a human would do it, that is, considering what in the image is blurred. Thus, in this sense, a blurred image is defined as an image whose blurred regions affected a significant part of it, whereas a sharp image is defined as an image in which there is no blurred region or the blurred regions correspond to an unimportant content.

This objective can be further detailed as follows:

- Select a database of images or create one.
- Extract relevant features out of the images from the database.
- Select and train different machine learning models using the features.
- Evaluate the models with respect to the proposed classification, as defined above.

Since what is important or not in an image is highly subjective, a solution to this instance of the problem is not trivial, such that simple sharpness metrics will not be successful. In addition, there are other aspects of an image that may be on the way to a successful method, such as luminance and chromatic noise, poor exposure, low luminosity and compression artifacts. Furthermore, to the best of our knowledge, there was no publicly labeled set of images for this specific application available during the development of this research.

1.4 Contributions

The major contributions of this research are:

- a framework for combining foreground and background images, generating a labeled dataset of 14,742 images suitable for the purpose of this work, of which 2,268 images will be publicly available.
- proposition and development of new feature descriptors, called Sharpness Behavior (SHB) and Fourier Transform Variance (FTV).
- evaluation of a Support Vector Machine (SVM) [11] classifier using the new feature sets along with the Histogram of Gradient (HOG) [13] features and the generated test images, comparing it against some features and metrics designed for similar applications, and analyzing the performance of the trained model at different levels of noise and compression, between the easier and harder cases and the use of a hypothetical object detection that would crop the image in the object of interest.

1.5 Text Organization

This text is organized as follows. In Chapter 2, we describe important concepts related to the problem addressed in this work, as well as relevant work of the literature. The proposed methodology for creating the dataset, for the developed features and for the evaluation adopted is presented in Chapter 3. The results achieved following the proposed methodology are shown and discussed in Chapter 4. Finally, concluding remarks and directions for future research are outlined in Chapter 5.

Chapter 2

Background

In this chapter, we present the fundamental theoretical background which underlies the work carried out in this research, followed by the review and description of some relevant approaches available in the literature for closely related applications.

2.1 Related Concepts

In this first section, we introduce the concepts directly or indirectly related to image blurriness estimation, detection and classification, starting from the idea of sharpness and blurriness, followed by a brief discussion about noise and compression, which can affect the descriptors for image blurriness. Finally, we describe the Histogram of Oriented Gradients features, which were also used in our work as descriptors and input to the classifier.

2.1.1 Image Sharpness and Blurriness

In the image domain, sharpness and blurriness can be seen as the opposite extremities in measuring the same attribute of an image, in the same way as brightness and darkness. Therefore, one can use the terms when talking about this image attribute. In general, they are directly related to how abruptly pixel values change spatially and that is why sharpness has been define in some works as the angle of the line formed by the spatial coordinates of two pixels and their respective values [14], which can be expressed as $S = \tan^{-1}(v_1 - v_0)$, where S is the sharpness value and v_0 and v_1 are the values of two consecutive pixels in some given direction. Thus, following this chain of thought, the sharper the transition, the higher the sharpness and, consequently, the lower the blurriness.

Image blurriness is affected by a number of factors, such as focus, lens aberrations and motion. While lens aberrations cause different blurriness with different intensities along the field of view, with more blurring toward the edges (borders) of the image, focus blurriness is modified by the distance of objects to the camera where objects within a distance range $R = [d_0..d_1]$, which is called the depth of field, are considered sharp and the others get blurred proportionally to their distance to the center of the depth of field [41].

The image (projection) of a single point source of light is called point spread function (PSF) and is intended to show the lens aberrations of a particular camera system, if any,

such as defocus. In a geometric optics approximation, that is, ignoring the effects of diffraction, a point source of light is rendered in the PSF as a single point if this point is at a distance $d = d_0 + \frac{d_1 - d_0}{2}$ from the camera. If the point is at a different distance, the point on the PSF becomes a spot with finite diameter, also known as circle of confusion. As long as this spot is projected within a single pixel of the sensor, the distance of the object will not cause any noticeable distortion (blur) in the image, but as it grows larger and encompasses more pixels, the blurriness of the region of the image increases [6, 41]. This concept is depicted in Figure 2.1.

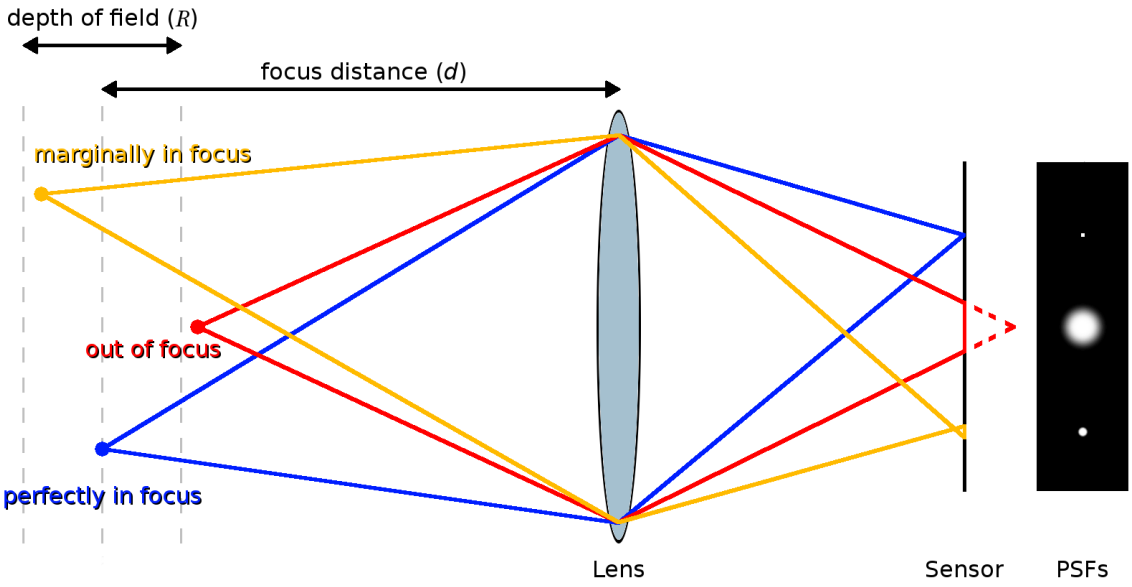


Figure 2.1: Simplified ray diagram and PSF maps.

As for motion in both cases, object motion or camera motion, the sharpness of the objects or the whole scene can be changed if the motion is fast and large enough that light coming from the same point of an object is collected by different pixels in the sensor during the exposure time. In the first case, only the moving object is blurred, whereas in case the camera is moving, the whole scene/image is blurred [6, 41].

In Figure 2.2, we show an example of how light from the same point of an object is spread on the sensor as the object moves, depicting that point at three different instants t_0, t_1 and t_2, which occurs within the exposure time.

Taking into account these different factors, the blur present in an image can be modeled using a map of convolutional kernels, as shown in Equation 2.1, where there is a specific kernel for each pixel of the perfect image. With such a map, all factors can be represented and also how they can affect differently each pixel due to different object distances to the camera and possible motion.

$$\hat{I}(x, y) = I(w(K(x, y), x, y)) * K(x, y) \tag{2.1}$$

where $\hat{I}$ is the blurred image; I is the perfect image with no blur distortions; the operation $*$ denotes a convolution; K is the kernel map; w is the window for the original image with the shape of $K(x, y)$ and centered at the pixel (x, y).

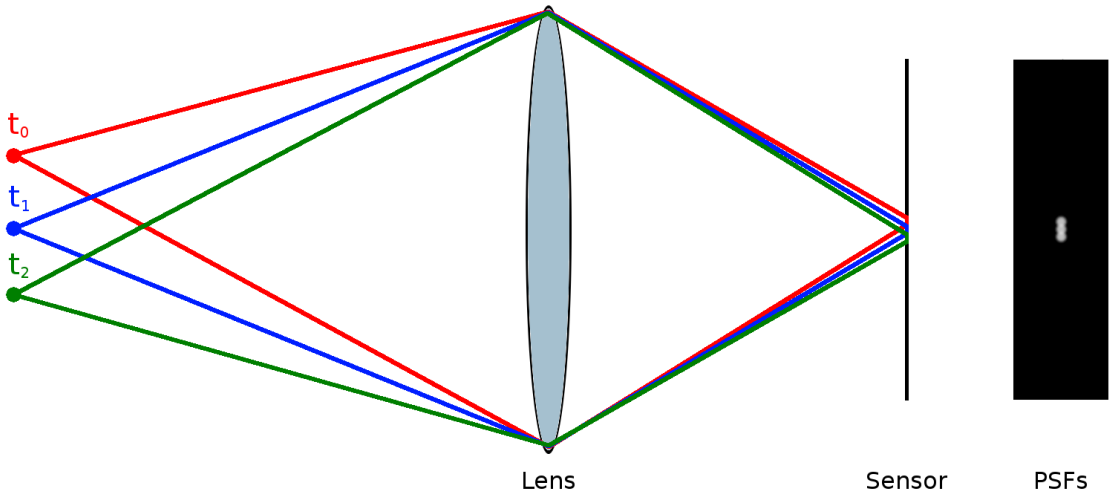
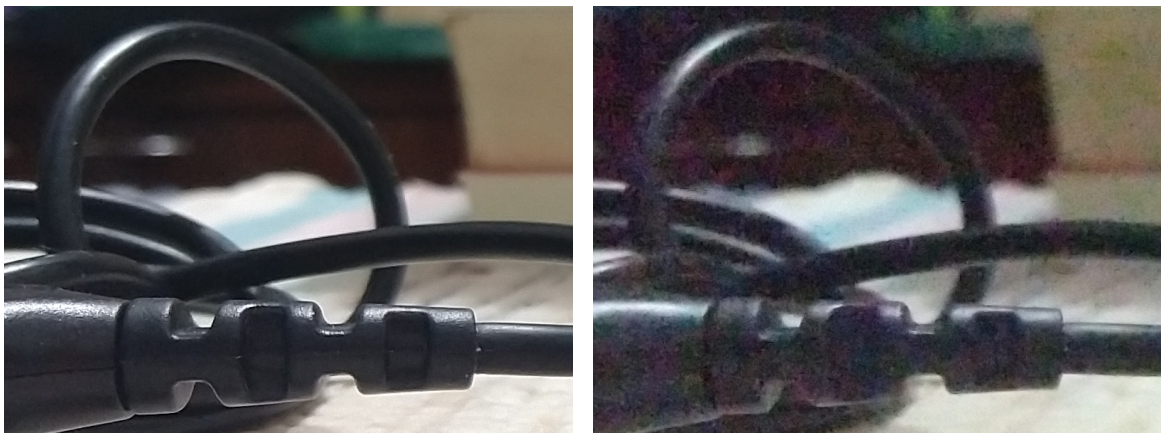


Figure 2.2: Simplified ray diagram of a moving point.

2.1.2 Image Noise and Compression

Noise is the part of a signal that is undesired, that is, the part with no useful information for a given application of that signal and, furthermore, it often becomes a problem obscuring the signal of interest [44]. Noise is, in some intensity, always present in all images taken with a digital camera and its visibility depends on different factors such as the quality of the camera components (the lens may also affect it, but it is the sensor that is the dominant source of noise in the image), lighting conditions, post-processing noise filtering, viewing distance, spatial content, coloring and location in the field of view [21, 41].

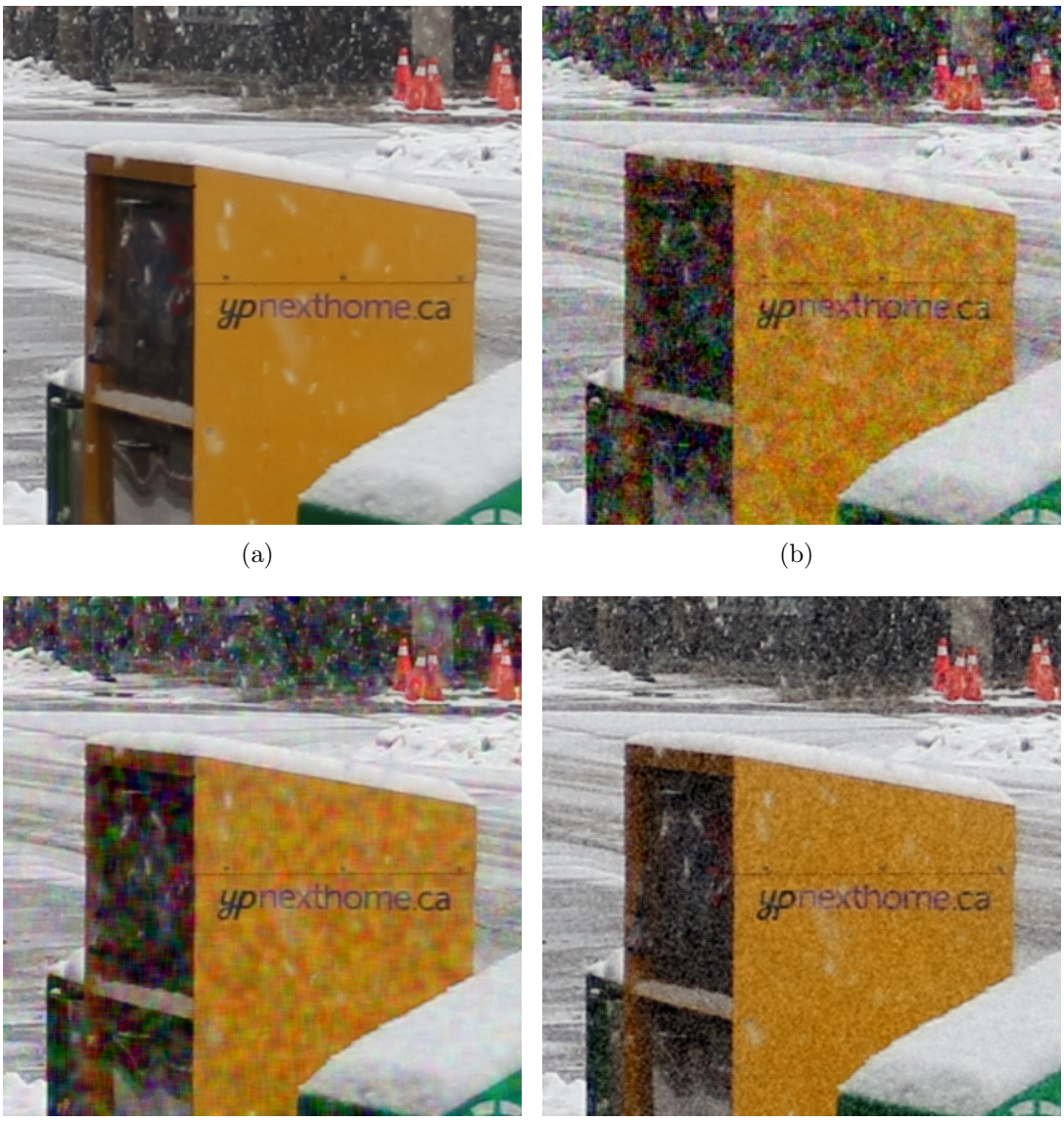
In digital cameras, a setting that directly affects the amount of noise in the final image is the International Organization for Standardization (ISO) speed [25] that specifies the sensibility of the sensor to light and which does not cause noise, but rather may or may not amplify it. Figure 2.3 shows how different ISO speeds can affect the noise intensity in the final image.



(a) (b)

Figure 2.3: Effect of ISO speed (ISO 12232) in the perceptible noise. (a) ISO speed = 100; (b) ISO speed = 3197.

A common type of noise in the images is white noise, which is defined as a noise independent of the spatial location and frequency of the image signal. This is the type of noise we assume and simulate in this work [21]. In addition, noise can be further classified in color images into luminance and chrominance categories. Luminance noise is one that only visually affects the brightness of the pixels, while the later visually changes the color of the affected pixels. Figure 2.4 demonstrates the difference between them using our method for noise simulation, explained in Section 3.1.2. In natural images, both are always present.



(a) (b)

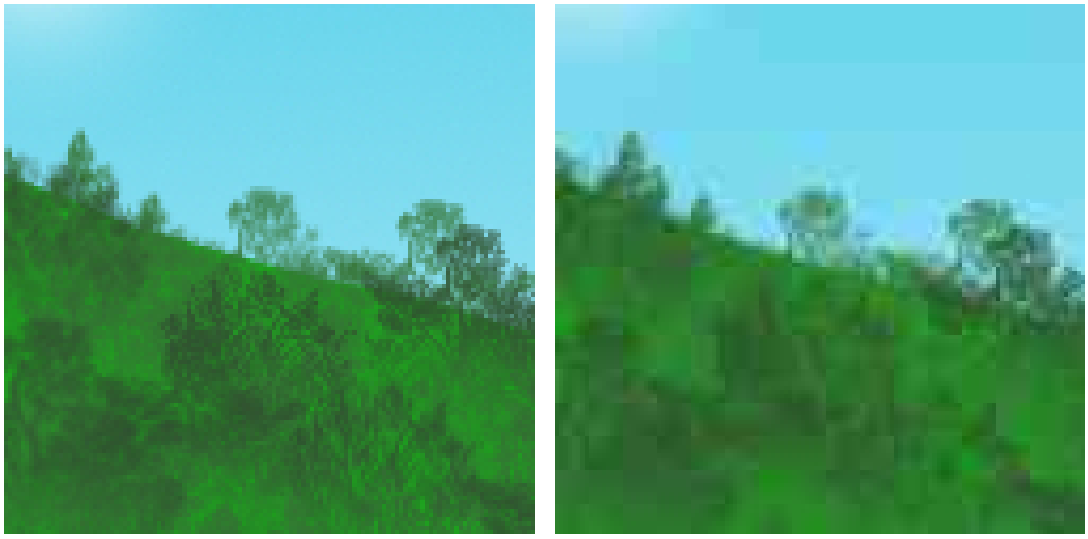
(c) (d)

Figure 2.4: Luminance and chrominance noise simulated with our method. (a) original image with very little noise; (b) image with chrominance and luminance noise; (c) image with only chrominance noise; (d) image with only luminance noise.

Many camera systems apply at the last stage in the processing pipeline, a compression algorithm, so the final image will not take up a large amount of disk space and will be faster when traveling through a network [41]. To put this in perspective, an 8 megapixel

image with 3 color channels takes 24 megabytes of disk space without considering the metadata information, and only 43 of them would occupy just over 1 gigabyte of space.

The most commonly used compression method is JPEG (Joint Photographic Experts Group), which consists basically of three steps: chroma subsampling, transform coding and entropy coding. In the first, the chrominance information is separated from the luminance by mapping the pixels of the Red-Green-Blue (RGB) color space to Luminance-Chrominance (YCbCr) space. Then, the Cb and Cr channels, where the chrominance information is, are subsampled. In the second step, the image is divided into a grid of blocks and each block is transformed into the frequency domain with the Discrete Cosine Transform (DCT), whose coefficients are quantized and mapped to a different binary code, depending on their number of occurrences (using a method such as the Huffman coding) [21, 41]. Due to the block-based transformation to the frequency domain, when the image is decoded, we can see some artifacts in a block-wise fashion, depending on the amount of loss, as shown in Figure 2.5.



(a) (b)

Figure 2.5: Example of JPEG compression artifacts. (a) image with a lossless compression; (b) image with a 70% loss compression.

2.1.3 Histogram of Oriented Gradients

The Histogram of Oriented Gradient (HOG) features have been used for image classification since 2005 [13, 16, 52, 58]. According to Dalal and Triggs [13], it is basically implemented by splitting the input image into regions in a grid pattern, where each rectangle of that grid is called a "cell". In addition, for each of them, the gradient of each pixel is computed with a [-1, 0, 1] filter and also its direction. Then, a histogram of the directions of the gradients are calculated with 9 bins in the interval [0°, 180°], one for each cell as well.

The histograms of all combined cells are the HOG features. Finally, a normalization of histograms in all sets of neighboring cells (such sets are called "blocks") is performed.

This procedure gives the features a better invariance for different illuminations. The implementation of the HOG features we used are based on the work by Dalal and Triggs [13]. In this work, we also used 9 bins for the histograms, the L1-norm for normalization, cells of 100×100 pixels and blocks of the size of a single cell.

2.2 Related Work

The set of researches related to this work can be categorized into two high-level groups: image processing and image analysis. The first one encompasses researches whose goal is to generate a new image without blurred regions, such as [9, 29, 32, 45, 62]. The second consists of researches whose results are information about the blur in the images, such as intensity or category. Since our work resides in the second group, from now on, we focus only on its representatives.

Methods such as the Structural Similarity Index (SSIM) [55] aim to estimate the quality (including but not specifically related to blurriness) of an image using a "perfect" image as a reference. These methods are called full-reference approaches. More recently, Bong and Khoo [4] have also developed a method for assessing image blurriness based on the difference of the local contrast map of the original (reference) and the blurred image. These methods can correlate very well with human perception of blur, however, they cannot be used in many real-world applications where there is not a clean reference image available. Therefore, approaches that can work only with the input image itself have been developed. We will briefly describe some of these methods in the following paragraphs.

There are many methods for estimating and detecting blur in images based on edge detection, which are basically the center of a slope (transition) between two nearly homogeneous regions with different intensities/colors, usually computed as the magnitude of the image gradient. We have quoted some representatives of such methods as follows.

Ong et al. [37] improved the work of Marziliano et al. [34] and developed a method that estimates the edge-spread value, that is, the width of the edge in the spatial dimensions, based on the fact that the sharp edges are abrupt changes, therefore, they have small widths, whereas the blurred edges are wider. Their blur estimate is given by the average edge-spread value over all edges detected in the image, which is not good for detecting motion blur, since there are also sharp edges in a motion blurred image.

Hsu and Chen [23] designed a framework to estimate the extent of image blurring and categorize the type of blur. Their method is divided into three stages. In the first one, an SVM computes the extent of the image blur based on the histogram of gradient magnitude and the histogram of gradient direction. Images with a blur range greater than 0.5 are considered to be blurred and they proceed for the next stage, while the remaining images are considered clear and not further processed. In the second stage, the blurred images are further classified as having global blur or local blur, which is based on the proportion of the number of patches considered to be blurred and the number of patches considered clear in the image, using the same SVM of the previous stage. In the third stage, a method for estimating the PSF is applied in the blurred regions, its output is passed to

an SVM that classifies the image as defocus blur and motion blur. Two SVMs are then trained, one for globally blurred images and one for locally blurred images. In addition, the latter are segmented and, for each segment, the blur extent is computed, where the PSF is estimated only for segments classified as blurred. They show an accuracy of 99.5% for the first stage. Their goals are close to ours, but the definitions of their classes are different from those proposed in this work: their blurred image is defined as an image with any region blurred by motion or defocus, whereas our definition depends on the content of what is blurred in the image.

The work of Hong et al. [22] encompasses the development of a no-reference blur estimation, where they proposed a method to estimate the PSF across lines detected on the image and then use the spread parameter of the PSF to compute the final blur metric, which they called blur metric (BM) and was used to classify the images between blurred and not blurred, using a simple threshold classifier. Their method is explained in more detail in Section 2.2.1, since we have created a modified version of it to be used in the analysis of our results.

Crete et al. [12], driven by the theory that humans detect blurred images by comparing them with other images in their minds (even unconsciously) based on previous experiences and creativity, in their method, they transformed the input image by blurring it to have this second image for comparison. The idea is that, if they are very different, it is likely that the input image has been sharp, while if they are not so different, then it is more likely that the original image is also blurred. Their final blurriness estimation is given by the difference between the estimated blurriness (using absolute variation between the neighboring pixels) in the original and blurred images.

Sieberth et al. [48] created an automatic method to filter blurred images from a set of images taken from unmanned aerial vehicles (UAV) based on a metric to quantify the amount of blurriness in the images. Their metric function, inspired by the work of [12] and called SIEDS (saturation image edge difference standard-deviation), creates an image for comparison by blurring the original image, then they apply a high-pass filter in both versions of the image (original and blurred), computes the difference between the filtered images and, finally, the standard deviation of that difference image is computed. All these processes are performed using the saturation channel of the Hue-Saturation-Value (HSV) color space, hence the metric name. However, their metric works only for similar images, that is, images with similar content. This is due to the differences in the range of the resulting high-pass filtered images. In addition, they did not present a method for classifying images into blurred and sharp categories.

Other works with similar methods include the research by Zhuo and Sim [63], Golestaneh and Karam [20], Shi et al. [47], Yi and Eramian [59] and Rugna and Konik [27].

There are some methods that analyze how abrupt are the slopes in the images, mapping them to the frequency domain, where the image (a 2D signal) is represented by the responses of a set of frequencies. The assumption in such cases is that blurred images have low responses to high frequencies. Some of the work that follow this approach are from Tong et al. [51], Chen et al. [8] and Chen and Bovik [10].

Tong et al. [51] proposed a blur detection system that uses Harr wavelet transform and can classify an image as blurred and sharp, as well as quantify the amount of blur

in an image. To do this, they apply the Harr wavelet decomposition into three levels and combine the responses from three different orientations of each decomposition level into a single map, called an edge map. Then, each of the three edge maps are divided into blocks and, for each block, the point with the maximum value and the value itself are recorded. The next step is to categorize each of these points into two categories: one-pixel-wide edge and wider edge. This is done by comparing the values of each point in the three different levels. Finally, the image is considered to be blurred if the ratio between the number of edges of the one-pixel-wide category and the total number of edges is above a given threshold. The amount of blur is estimated by the ratio of the number of edges of the wider category that are considered blurred and the total number of edges of the same category. An edge of the wider class is considered blurred if the value of its point at the highest resolution is above a threshold. Therefore, their method requires two different thresholds, one for classification and the other for quantization. Furthermore, they report a classification accuracy of 98.6%, however, the images in their dataset of the blur class were all completely blurred.

Tang et al. [49] also extracted frequency domain features to estimate a defocus map. Shi et al. [46] used both frequency and spatial domain to extract their descriptors.

2.2.1 Hong et al.'s Blur Metric

Hong et al. [22] developed a blur metric (BM) that estimates the PSF spread parameter of the image by finding straight edges by computing the gradient of the edge's slope profile curve, in the perpendicular direction and assuming the standard deviation parameter of the Gaussian distribution function given by the gradient to be the PSF spread parameter.

The first step of their algorithm is to detect the edges of the image using the Canny edge detector [7], resulting in a binary image. The next step is to find the straight edges in the binary image using the Hough transform [17]. They established two criteria to maintain a detected line: (i) the line segment must be at least 12 pixels long; (ii) no edges present in the gradient direction within a distance of 16 pixels.

The third step consists of transforming the box around the straight edges so that the edge lines are in the vertical direction. The procedure described by Hong et al. is expressed in Equation 2.2

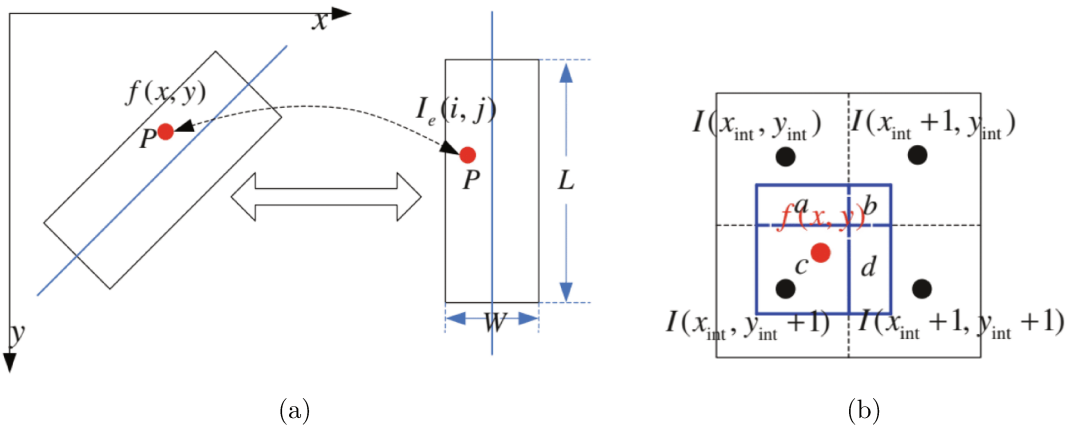
$$\begin{aligned} I_c(i,j) &= f(x,y) = \\ &= a.I(x_{int}, y_{int}) + b.I(x_{int}+1, y_{int}) + c.I(x_{int}, y_{int}+1) + d.I(x_{int}+1, y_{int}+1) \end{aligned} \tag{2.2}$$

and illustrated in Figure 2.6.

After adjusting the box orientation, the pixels are averaged across the columns, and then the gradient of that single profile line is computed. Then, the standard deviation parameter of the Gaussian distribution for each straight edge that passed both criteria is estimated directly by

$$\hat{\sigma} = \frac{1}{\sqrt{-2b_3}} \tag{2.3}$$

where $\hat{\sigma}$ is the estimated standard deviation parameter of the Gaussian distribution and



(a) (b)

Figure 2.6: (a) Illustration of region of the interest (ROI) adjustment and pixel mapping. (b) Position (x, y) and its four neighboring pixels. Source: [22].

b is given by

$$b = [b_1 \quad b_2 \quad b_3]^T = (A^T A)^{-1} A^T g \tag{2.4}$$

where g is the gradient of the profile curve and A is expressed as

$$A = \begin{bmatrix} 1 & x_1 & {x_1}^2 \\ 1 & x_2 & {x_2}^2 \\ \vdots & \vdots & \vdots \\ 1 & x_W & {x_W}^2 \end{bmatrix} \tag{2.5}$$

where W is the width of the adjusted box (that is, the number of columns), and x_1, x_2,..., x_W are the values of the average of the pixels across the columns, that is, the profile curve.

Finally, the final metric is computed as

$$\text{BM} = \frac{1}{T} \sum_{i \in E} \sigma_i \tag{2.6}$$

where σ_i is the spread parameter of the i-th straight edge detected; $E = \{i \mid |\sigma_i - \bar{\sigma}| \leq \sigma_{std} \; \forall \; i \in \{1, 2, ..., N\}\}$ is the set of σs not considered outliers; $\bar{\sigma}$ and σ_{std} are the mean and standard deviation of σs of all straight edges, respectively; N is the total number of straight edges; and T is the number of elements in E.

Chapter 3

Proposed Methodology

In this chapter, we present the procedures adopted for the development and evaluation of the methodology conducted in this research, namely (i) the database construction, (ii) the feature extraction, and (iii) the evaluation protocol.

Figure 3.1 illustrates the main steps of the proposed method, as well as how they connect with each other. In the following sections, we describe each of these stages.

3.1 Database Construction

To the best of our knowledge, there was no database available in the scientific community for such application during the development of our work. However, by the time we finished this research, a very recent work [60] collected and annotated a large data set with natural photos that fit well into this application.

One way to collect a large number of images is to download them from a Web search engine, however, it is very time consuming to manually label such a large amount of images. In addition, by manual labelling, errors and inconsistencies may be introduced into the data set. Furthermore, such data set can be highly unbalanced, since it is much easier to find sharp images than blurred ones in a Web search, for instance.

For these reasons, we have chosen to use data augmentation techniques [54] to artificially generate a very diverse data set with a considerable number of images and automatic labels. We have identified some specific types of scenes that can be encountered in this application:

1. the entire image is blurred:
   (a) due to focal lens position.
   (b) or due to motion caused by the camera movement.
2. the entire image is sharp.
3. part of the image is blurred and part is sharp:
   (a) the camera focus is on far objects (background).
   (b) or the camera focus is on near objects (foreground).

(c) or there is motion blur caused by the object movement.

To generate all these different cases, we decided to develop a method that combines background and foreground images, all of which were originally sharp, applying image

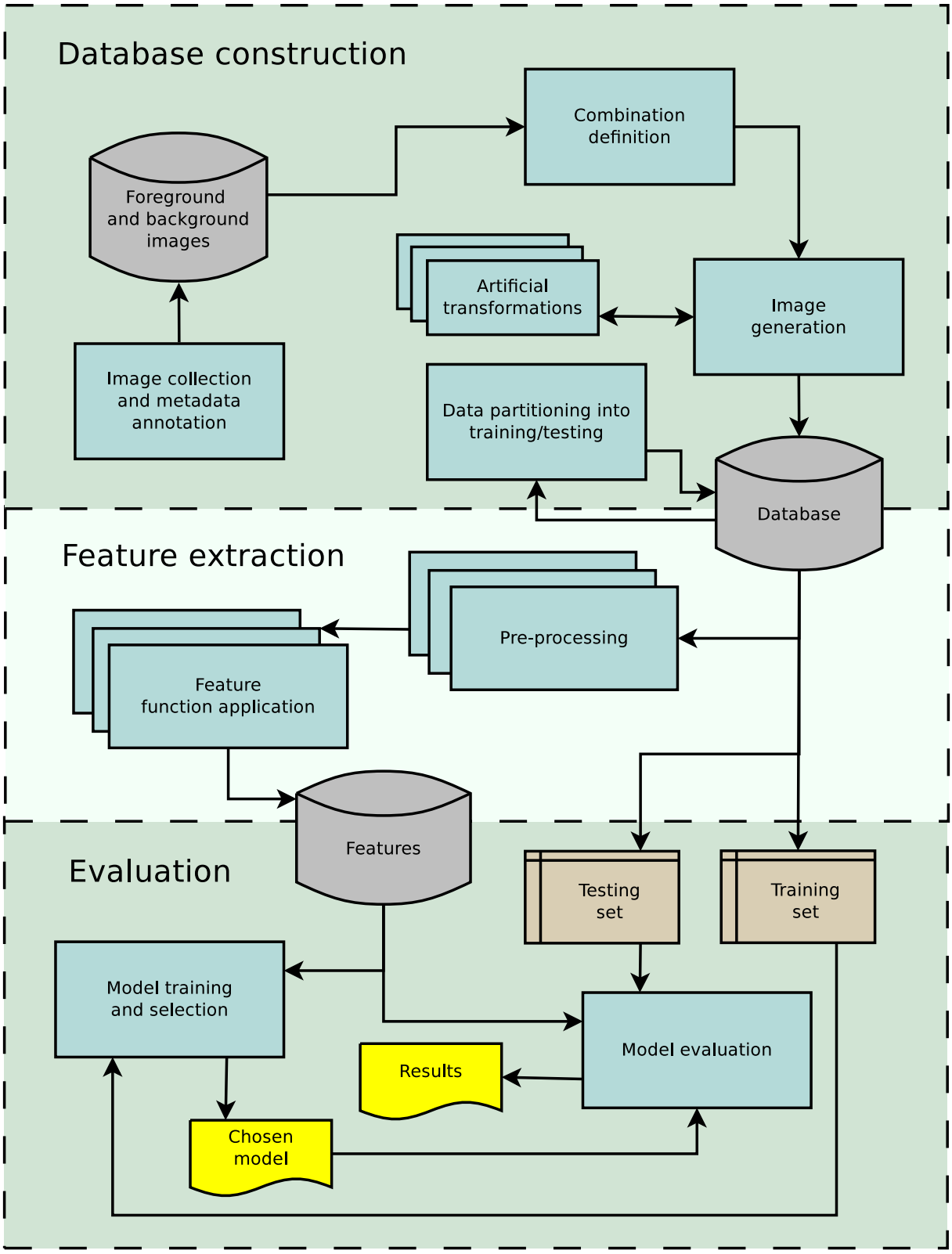


Figure 3.1: Diagram that illustrates the main steps of the proposed methodology.

processing techniques to blur them, in order to simulate focus and motion blur to either the foreground or background, or both. The scene types are exemplified in Figure 3.2.

(a) (b) (c) (d) (e) (f)

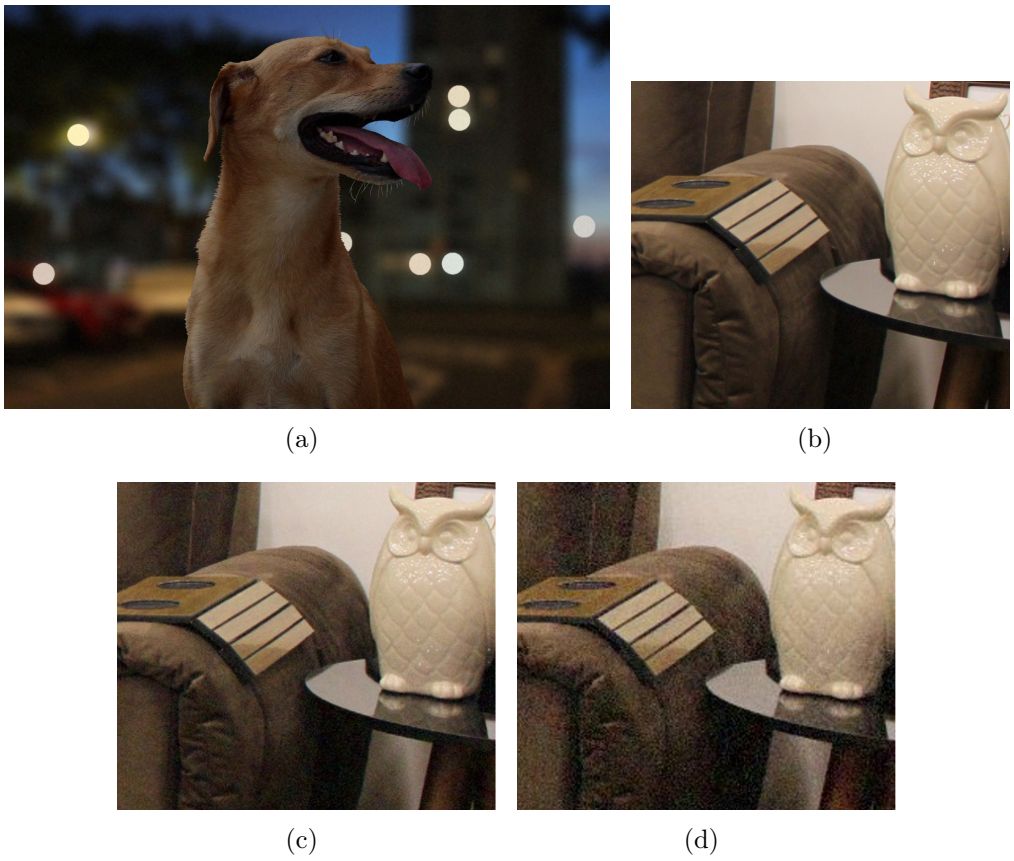
Figure 3.2: Examples of scene types (images selected from our database): (a) the entire scene is blurred due to focal lens position or (b) due to motion caused by the camera movement; (c) the entire image is sharpened due to lack of motion and very large depth of field; (d) part of the image is blurred and part is sharp when the camera focus is on distant objects or (e) the camera focus is on near objects, or (f) there is motion blur caused by object movement.

We are also concerned with addressing other aspects of a picture that may impact the classification process. They are illustrated in Figure 3.3 and listed as follows:

- sources of light when out of focus. Since they produce blobs of usually a single color and with a well-defined edge, instead of a blurred effect, they produce a sharp

region [6]. This can be a difficult case for a simple sharpness metrics to deal with:

- luminance and chromatic noise.
- compression.
- two different illumination types.



(a) (b)

(c) (d)

Figure 3.3: Examples of image aspects covered in our data set, using images from our constructed database. (a) out of focus sources of light and low light; (b), (c) and (d) increasing noise and compression, bright light.

Therefore, we have developed a method to simulate the effect of out-of-focus light sources and another to simulate an adaptive luminance and chromatic noise, while adopting the well-known and widely used JPEG compression algorithm to have different levels of compression. As for the different illumination types, we merely use those of the original background and foreground collected images. All of the methods mentioned so far are explained in the following subsections.

This approach used to generate the database allowed us to collect some images from the Web and then obtain a set of approximately 11 thousand images. Due to licensing issues, we took a separate set of images to be used as foreground and background using a Canon EOS Rebel T5i camera. These images were used to generate the testing set, such that it could be published and made freely available for non-profitable research purposes. The test set, as well as the code to generate it, are publicly available at [3]. Thus, the

scientific community can improve the method, expand the data set and compare the results consistently.

3.1.1 Image Collection

We now describe how we collected, processed and annotated the background and foreground pictures. First, we downloaded a set of images manually from Google Image search engine with a filter set to retrieve only images labelled for reuse or non-commercial reuse, which does not guarantee unlimited usage rights [33]. For this reason, we took our own pictures in the field to generate our test set, while the images collected from the Web were used in the training set (see Section 3.1.4 for details on how the data was divided). In both cases, we aim for high-resolution images and for them to be as clean as possible in terms of noise and compression artifacts.

Background images can be used without further changes, however, foreground images must have the *alpha* channel, which allows for transparency, a necessary property so that we can place the foreground objects seamlessly on the background. For this reason, for all foreground images that did not have this property, we manually edited them using the free image editor GNU Image Manipulation Program (GIMP) [30] to add the *alpha* channel, segment the images around the object borders and save them in Portable Network Graphics (PNG) format.

To enable the generation of images with some degree of cohesion, we defined a notation system in the image filenames so that we can combine the proper background and foreground pairs. This system can be easily extended to include new attributes to make the final images generated even more semantically coherent. The fields we are using are:

- name: a very short description of the image.
- base: a numerical estimation of the real length represented by the bottom edge of the image, in centimeters.
- environment: a text string to specify whether the scene in the image is outdoor or indoor. Possible values include *in*, *out* and *both* (the latter is applicable for the foreground images only).
- light: a text string to specify whether the scene has low light or high light condition. Possible values include *bright* and *dark*.
- light spots: a text string to specify if there are any source of light in the image. Possible values include *no_ lights lights*.

Each field is separated by a *hifen*, such as in `car_1-400-out-dark-lights.png`. Thus, our algorithm parses the filenames to decide which background-foreground pairs are allowed; more specifically, a background and a foreground can only be combined if and only if they have passed all of the following criteria:

1. both must have the same environment value, unless the foreground environment is *both*, in which case it will always pass this criterion.

2. both must have the same light value.

We collected in total 136 images (66 foreground images and 70, for background) from the Google search and 39 images (18 foreground images and 21, for background) using a Canon EOS Rebel T5i DSLR digital camera, covering different places, objects and illumination types. We chose a DSLR camera to take our own pictures due to its well known high quality images.

3.1.2 Artificial Transformations

In this subsection, we explain how we define the functions to simulate each of the aforementioned conditions in the data set. We start with the focus blur, which is defined as a 2D convolution with a Gaussian kernel:

$$\hat{I} = I * K \tag{3.1}$$

where I is the input image, K is the Gaussian kernel, whose σ component is calculated as $\sigma = \dfrac{s-1}{3}$, where s is the kernel size.

For motion blur, we also use a 2D convolution, but with a directional kernel instead of the Gaussian one. Given the kernel size (see Section 3.1.3 for details on how the kernel sizes for focus and motion blur were chosen) and a direction angle, such a kernel is computed by setting its elements, which would form a line that crosses the kernel in the center and has the specified angle to one, whereas the remaining elements are set to zero. Formally, given an angle in radians θ and a kernel size s, we can express:

$$K(E) = 1 \tag{3.2}$$

$$\begin{aligned} E = &\left\{(x,y) \mid y = \frac{\cos(\theta)x}{\sin(\theta)} \quad \forall\, x \in [0\,..\,s-1]\right\} \cup \\ &\left\{(x,y) \mid x = \frac{\sin(\theta)y}{\cos(\theta)} \quad \forall\, y \in [0\,..\,s-1]\right\} \end{aligned} \tag{3.3}$$

The luminance and chromatic noise were simulated as follows: considering an 8-bit image, we must compute a weighting mask and the noise mask itself. The weighting mask will simulate the visible effect in real pictures, where the noise is not homogeneous throughout the image, it is rather stronger in dark regions, while it is not present in saturated ones. These weights are defined as:

$$W(I) = \frac{255 - I_{gray}}{\frac{\sum I_{gray}}{255N}} \tag{3.4}$$

The values for the noise masks N_{chr} and N_{lum}, for chromatic and luminance noise, respectively, are generated randomly. The resulting image is given by

$$\hat{I} = I + (\gamma_{chr} N_{chr} W(I)) * K_{chr} + (\gamma_{lum} N_{lum} W(I)) * K_{lum} \tag{3.5}$$

where γ_{chr} and γ_{lum} are scalars to control the intensity of the noise, for the chromatic and luminance noise, respectively; K_{chr} and K_{lum} are square-shaped kernels to blur both noise masks. This is necessary since noise in the real world rarely has the exact area of one pixel and is not always sharp, specially for the chromatic one; the operator $*$ denotes a convolution. Section 3.1.3 explains how the values for these parameters were selected.

Finally, we describe the algorithm to simulate the defocus of light spots, which begins by detecting blobs in the grayscale image with an approximation of the Determinant of Hessian (DoH) method, presented by Bay et al. [2]. Then, we discard all blobs with brightness below a threshold, which depends on the distribution of blob brightness for the given image, that is, the brightness threshold $\tau_{brightness}$ is computed as

$$\tau_{brightness} = \left\lfloor \left(\frac{\max(B) - \min(B)}{100} \right) 80 \right\rfloor \tag{3.6}$$

where B is the set containing the brightness values of all blobs detected for the given image. The constants were defined empirically.

We define brightness of a blob as the simple average of the grayscale pixels belonging to the blob. The color of a blob is also computed and is defined as the average of the pixels in the blob area with its RGB (R+G+B) sum smaller than 80% the maximum possible value, which is 765.

Next, for each blob with the brightness above $\tau_{brightness}$, we find all the pixels belonging to the circle given by the center and radius of the blob. We also convert the blob color from the RGB color space to HSV, so we can change its intensity without changing the color. If all blob radii are equal, we set their intensity (the V channel) to 0.9, otherwise we set the intensity of each blob as

$$v = 0.6 \frac{r - r_{\min}}{r_{\max} - r_{\min}} + 0.4 \tag{3.7}$$

where v is the final V channel value; r is the blob radius; r_{max} and r_{min} is the largest and smallest radius among the blobs, respectively. The constants were defined empirically.

Therefore, the intensity varies from 0.4 to 1.0, proportionally to the blob radius. Finally, we increase the saturation of each blob by 60% and then convert it back to the RGB color space.

Before drawing the circles, we slightly blur the edges and add some random noise. The blurring is done with a Gaussian kernel of 4×4 and $\sigma = 1$, while the noise can go up to 10% of the maximum value of the image. In addition, the RGB values of the blobs are normalized, so they will not be less than 70% of the maximum possible value. Then, we just put the circles on the original image.

An example of the results of this method is shown in Figure 3.3(a). As can be seen that image, our method is very good for perfectly reproducing the real effect in the whole image that happens in large aperture camera systems, but it is enough for the purposes of this work, that is, to simulate the defocus effect of light sources in some regions in the images.

3.1.3 Image Generation

Our algorithm works by iterating over a data structure where it is possible to specify how many and which types of combinations must be generated. In the first layer, the blur method is specified (whether to blur the foreground or background, whether it is camera or object movement) and its intensity, the amount of noise and compression, and the rescaling factor.

For each of these combinations, in the second layer, the specifications for the foreground must be defined, that is, the number of foreground objects, their horizontal position and relative depth. Here is an example of this data structure with two complete specifications:

```
Spec(
    foreground_specs=[
        ForeSpecs(positions=[0], depths=[0]),
        ForeSpecs(positions=[1], depths=[2]),
        ForeSpecs(positions=[0], depths=[4]),
    ],
    blur_method='macro',
    blur_level=6,
    noise_level=1,
    compression=5,
    scale=0.65
)
Spec(
    foreground_specs=[
        ForeSpecs(positions=[0], depths=[0]),
        ForeSpecs(positions=[1], depths=[2]),
        ForeSpecs(positions=[-2,1], depths=[3,0]),
    ],
    motion_method='camera_motion',
    motion_level=7,
    motion_angle=45,
    noise_level=1,
    compression=0,
    scale=0.8
)
```

For the specification of the horizontal position of the foreground objects, we defined the possible values $\{-2,\ -1,\ 0,\ 1,\ 2\}$, where 0 corresponds to the middle of the image, negative values are for positions in the left side, while the positive values indicate positions in the right side of the image. Higher absolute values are for the positions in the extremities. As for the relative depth, they merely tell which objects are to be rendered in front of which objects. Furthermore, we have defined a mapping from the noise level

(l) to the actual parameters of our noise function, as expressed in Equations 3.8 to 3.11.

$$\gamma_{chr} = \frac{0.285(l-1)}{9} + 0.015 \tag{3.8}$$

$$S_{K_{chr}} = \frac{14(l-1)}{9} + 7 \tag{3.9}$$

$$\gamma_{lum} = \frac{0.095(l-1)}{9} + 0.005 \tag{3.10}$$

$$S_{K_{lum}} = \frac{l-1}{9} + 1 \tag{3.11}$$

where $S_{K_{chr}}$ and $S_{K_{lum}}$ are the sizes of the kernels for the chromatic and luminance noise, respectively. All constants were defined through trial and error iterations along with visual inspection.

Similarly, we mapped the blur and motion levels to their corresponding kernel sizes:

$$s_f = (0.0015\ l_f + 0.003) \cdot \min(w, h) \tag{3.12}$$

$$s_m = \frac{(60-\tau) \cdot (l_m - 1)}{9} + \tau \tag{3.13}$$

where s_f and s_m are the kernel sizes for the focus and motion blur, respectively; l_f is the level for the focus blur (called just "blur level" in our JSON structure); l_m is the motion level; w is the width of the image; h is the height of the image; and $\tau = max(3,\ 0.003 \cdot \min(w, h))$. All constants were defined through trial and error iterations along with visual inspection.

In our experiments, we defined three different levels of noise, compression and rescaling, each. Combined with the blurring methods, it resulted in 54 different first-layer specifications, and for each of them we have defined three different foreground specifications, which means that, for each background image we collected, 162 images were generated. Therefore, we generated 11,340 images with images downloaded from Google search and 3,402 with images taken with the DSLR camera. The pseudocode for the data set generation is shown in Algorithm 1.

It is also important to mention that our method automatically generates and saves the label information in the filename of the generated images, which is in the format `combination_<ID>_<CLASS>.jpg`. It also records all the specifications provided for each image in a CSV file, linking the specifications to the image by its ID number.

The classes are defined as `sharp` and `blurred`, numerically 1 and 0, respectively. The class assigned to each image depends on the foreground object state, that is, if the foreground object (the one defined with the smallest relative depth) is blurred, either due to motion or focus, then the class of that image will be `blurred (0)`, otherwise it is `sharp (1)`.

Data: Paths to the folders of the background and foreground images, the path to the folder where the generated images will be saved and the specifications object.
Result: Images generated and saved in the given directory.

```
for each background image b do
    for each specification s do
        for each foreground specification s_f do
            if blur method is 'macro' then
                if there are light sources in b then
                    blur b simulating the light sources defocus;
                else
                    blur b;
                end
            end
            for each foreground position p_f and depth d_f in s_f do
                get a foreground image f that can be combined with b;
                if blur method is 'hyperfocal' then
                    blur f according to d_f: the smaller d_f, the more intense the
                     blurring;
                else
                    if blur method is 'macro' then
                        blur f according to d_f: the bigger d_f, the more intense the
                         blurring;
                    end
                end
                if motion method is 'object_motion' then
                    directionally blur f;
                end
                combine f and b into one new image c;
            end
            if blur method is 'whole' then
                if there are light sources in b then
                    blur c simulating the light sources defocus;
                else
                    blur c;
                end
            end
            if motion method is 'camera_motion' then
                directionally blur c;
            end
            apply noise as specified in s;
            save image with compression specified in s;
        end
    end
end
```

Algorithm 1: Generation of the data set.

3.1.4 Data Partitioning

We have divided the data set into training and testing sets, establishing a formal protocol for evaluating classifiers with this database. To ensure that all 54 different specifications, as well as all types of scenarios (given by the different background images), would be represented in both sets in a balanced manner, we randomly chosen 2 out of the 3 foreground specifications of the images taken with the DSLR camera to go to the test set, while another part was assigned to the training set.

As for the Google search images, all of them were used in the training set. Therefore, it resulted in a training set with 12,474 images and a testing set with 2,268 images.

3.2 Feature Extraction

We designed a pipeline of pre-processing transformations followed by feature extraction functions, as well as most traditional computer vision approaches. Each of the pre-processing transformations is addressed in Section 3.2.1. In Section 3.2.2, we introduce two novel feature sets developed in this work to assess image blurriness/sharpness. In Section 3.2.3, we describe some improvements made in the metric proposed by Hong et al. [22] to improve it.

3.2.1 Pre-Processing

These transformations are intended to prepare the raw data for the feature extraction step, to remove information considered to be not relevant to the problem in question, and to normalize it with respect to some important aspect for the given application.

In our case, we discard the color information from the images, adjust the size of the images (spatial resolution), so that the larger side of all images has the same lenght, and remove noise while preserving edges and texture. These processes are explained as follows.

Single Channel

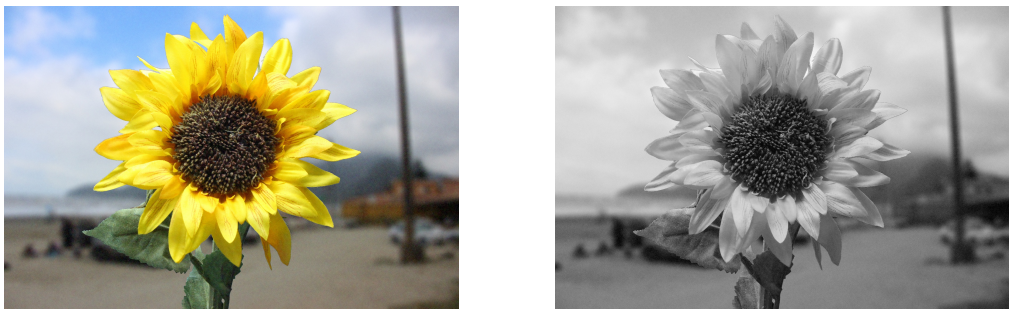
The raw input data is composed of images in the RGB color space, implemented as 3D arrays. Assuming that colors do not affect the blurriness/sharpness of an image, we remove this information from the images, resulting in grayscale images, represented as 2D arrays.

The function used to perform such mapping is expressed in Equation 3.14, reported in [26] as an approximation of the luminance as a linear combination of the RGB components, that is

$$I_{gray}(i,j) = I(i,j,red)w_{red} + I(i,j,green)w_{green} + I(i,j,blue)w_{blue} \tag{3.14}$$

where I_{gray} is the resulting 2D array, I is the original RGB image (3D array), i and j are the indices of the rows and columns of the images, respectively, whereas $w_{red} = 0.2125$, $w_{green} = 0.7154$ and $w_{blue} = 0.0721$ are the weights of the three color channels indicated in the subscript.

By applying this processing, we simplify the domain in which the feature extraction methods work, allowing them to be more efficient. In addition, this is the first pre-processing step applied to all images of the data set. Figure 3.4 shows an example of this transformation.



(a) (b)

Figure 3.4: Example of merging the three color channels into one, simulating the luminance of the scene. (a) original image; (b) transformed image.

Image Size Normalization

Our goal was to reduce difference of spatial resolution between the images, which will also simplify the domain and, therefore, allow the features to be more robust with that respect.

Our algorithm is straightforward: we compute the maximum value between the width and height of the image, and then calculate by what factor this value is less than a constant (800, in our experiments). Finally, we rescale the image by this factor.

For instance, if the width and height of an image are 1200 and 920, respectively, and the constant is 800, we rescale the image by the factor $f = \frac{800}{\max(1200, 920)}$. This transformation is applied to all images as the second pre-processing step.

Denoising

Since noise can have a major impact on sharpness/blurriness metrics, affecting spatial variation and local contrast, which is particularly easier to confirm in homogeneous regions in an image, the application of a denoising method can increase the accuracy of the used metrics/features, especially if the method is able to preserve edges and different texture patterns.

Looking for this characteristic in a denoising method, we chose the Wiener filter [56] to be applied to the images just after the rescaling. In our experiments, we used the noise-power as the average local variance of the input image and a 3×3 filter window. An example of the application of this filter is shown in Figure 3.5.

(a) (b)

Figure 3.5: Example of the denoising processing. (a) before denoising; (b) after denoising.

3.2.2 Feature Function Application

In this step, we apply a number of feature functions to determine the level of sharpness or blurriness of the images.

Sharpness Behavior

This is the first of the two new feature sets we propose in this work. Similarly to Sieberth et al. [48], we also draw inspiration in the work of Crété-Roffet et al. [12], that is, the idea that the difference between the sharpness of the image and the sharpness of a blurred version of the image should be greater when the original image is sharp than when the original image is already blurred. Based on this assumption, we developed a set of five features called sharpness behavior (SHB).

The blurred version of the image is computed with a mean filter

$$I_b(i,j) = \frac{1}{N_k} \sum_{i \in R_i} \sum_{j \in R_j} I(i,j) \tag{3.15}$$

where $I_b(i,j)$ is the pixel of the i-th row and j-th column of the blurred version of the image I, $N_k = k^2$ is the number of pixels in the kernel region, k is the kernel size of a square-shape kernel, R_i and R_j are the indices of the pixels that belong to the regions with the same area as the kernel, centered in i and j, respectively.

We used a very simple gradient estimation function as the basis for computing the sharpness values, which is given by

$$g_x(I(i,j)) = |I(i,j) - I(i,j+1)| \tag{3.16}$$

$$g_y(I(i,j)) = |I(i,j) - I(i+1,j)| \tag{3.17}$$

$$g(I) = \max(g_x(I), g_y(I)) \tag{3.18}$$

The first three features of SHB can be seen as sharpness metrics and are defined in the following equations

$$f_1(I) = \frac{1}{N} \sum_{(i,j) \in A} g(I(i,j)) \tag{3.19}$$

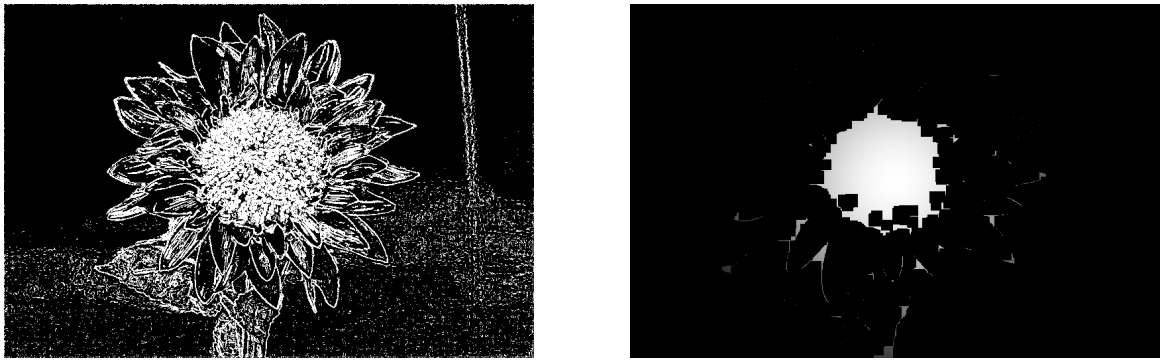
$$f_2(I) = \frac{1}{N} \sum_{(i,j) \in A} g(I_b(i,j)) \tag{3.20}$$

$$f_3(I) = \frac{1}{N} \sum_{i,j} g(I) \tag{3.21}$$

where N is total number of pixels in the image I, A is a set of pixels that satisfies the condition $A = \{(i,j) \; \forall \; g(I(i,j)) > m + g(I_b(i,j))\}$, and m is the mean value of $g(I_b)$. f_1 and f_2 can be described as the average of the estimation of the gradient of the original image and its blurred version, respectively, considering only the pixels from the set A. f_3 is the same as f_1, but taking into account all pixels of the original image.

The set A allows us to consider only the pixels that are sharper in the original image than in the blurred version. This is interesting because it ignores the pixels of a homogeneous area near the border of the objects in the image whose sharpness is increased when the image is convolved with a low-pass filter (due to the spreading effect of the object pixels), making the gradient of those pixels higher in the blurred version than in the original image.

The other two features aim to consider the spatial distribution of gradients based on the assumption that the region of interest in the image lies in the center and, therefore, the closer to the center, the more important is that pixel. To compute them, we use a set of pixels other than A, called here as B, where the regions formed by the pixels belonging to it are much larger than A's, as shown in Figure 3.6, and also use a Gaussian distributed weights, so that the largest weight is in the center of the image, following the assumption already stated.



(a) set A (b) set B

Figure 3.6: Example of the difference between sets A and B, extracted from the image in Figure 3.4. For the set A, the white pixels represent those that belong to the set. For the set B, pixels with a value other than zero (black) belong to the set and the brighter they are, the greater their weight.

The set B is defined as the gradient pixels of the original image, after a closing operation, whose value is above a threshold τ. Formally, we have

$$B = \{(i,j) \ \forall \ g(I(i,j)) \bullet s > \tau\} \tag{3.22}$$

where s is the structuring element of the closing operation, a full 15×15 square-shaped element. τ is a gradient value of the original image, where its accumulated distribution is above 97% of the total. These constants were chosen based on the visual inspection of multiple executions with different images.

These features are then specified as

$$f_4(I) = \frac{\sum_{(i,j) \ \in \ B} G(i,j)}{\sum_{i,j} G(i,j)} \tag{3.23}$$

$$f_5(I) = \frac{1}{N_B} \sum_{(i,j) \ \in \ B} g(I)G \tag{3.24}$$

where G is the Gaussian mask with its σ equal to a fifth of the maximum dimension of the image, normalized from 0 to 1, $N_B = |B|$ is the number of pixels in B.

Fourier Transform Variance (FTV)

The decision to use these features came from comparing the Fast Fourier Transform (FFT) [1] coefficients in both directions between different blurred and sharp images. An example is shown in Figure 3.7.

After computing the FFT components (real and imaginary parts), we calculate their corresponding magnitude coefficients and map them on a logarithmic scale. The next step was to find the absolute difference between each pair of coefficients in the sequence for both directions, considering only the coefficients where one of the frequency of the directions is always zero (in the coefficients magnitude matrix, which corresponds to the first row and first column). In the last step, we calculate the average of the differences, excluding the coefficients corresponding to the lowest and highest frequencies for each direction. In our experiments, we divided the frequency range into eight parts and considered only the second and third ones. These values were chosen through graphical analysis of the coefficients of different images, both blur and sharp.

These procedures are repeated for a blurred version of the original image, generated in the same way as the SHB features described in Equation 3.15. This results in four features: the mean absolute difference in the coefficients for horizontal and vertical directions for the original and blurred version images.

3.2.3 Our Implementation of BM

As the code proposed by Hong et al. [22] (See Section 2.2.1 for the description of their method) is not available and their paper does not provide all the necessary implementation details, we did our own implementation of the Hong et al.'s BM to compare our classification results. We created an additional step to be tested separately. The implementation

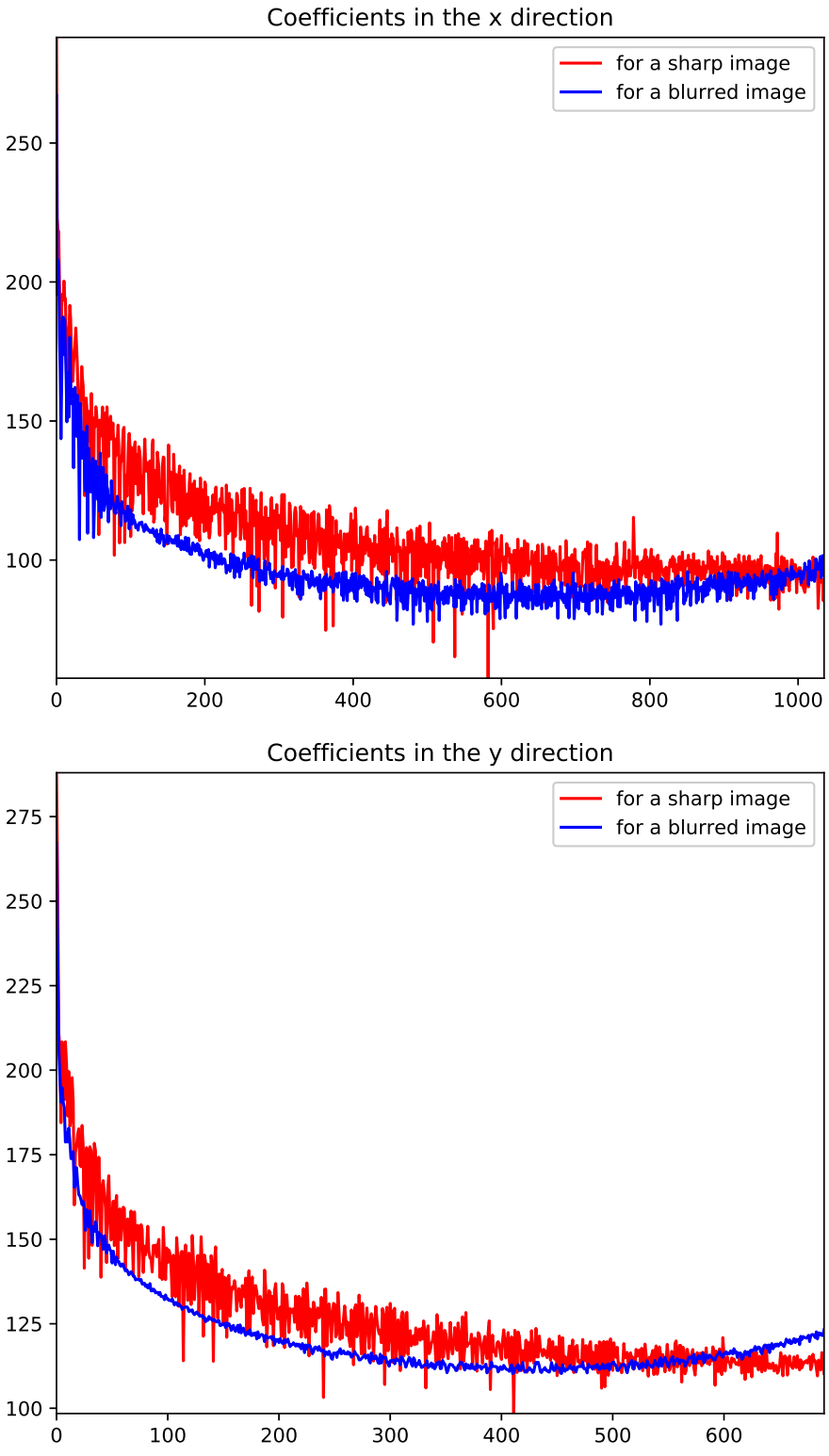


Figure 3.7: How FFT coefficients differ for blurred and sharp images.

is described in this section.

For the Canny edge detector, since the value of the standard deviation of the Gaussian blur applied in the Canny edge detection is not specified, we set it to 0.1 because it is the default value of the Canny algorithm we use. In the next step, no details other than

the minimum size of the straight edge segments was provided. Thus, we chose some sets of parameters to be used in a first attempt and, in the case of not finding lines in this attempt, we changed the set of parameters to one that allows more lines to be found. These parameters are chosen from a finite set that we define, so that a maximum of 10 attempts would be executed and if, after all attempts, no line is found, our method will return a high value, which indicates a high intensity blurred image.

The first criterion for maintaining a straight line is met in our implementation by setting it in the Hough transform function as a parameter. For the second one, we create a map with the same size as the original image, marking all found lines in it and then traversing the straight edges, we check that there are no lines marked on the map in the rectangle centered on the current line with 32 pixels in the gradient direction (16 for each side of the line) and the same length of the line in the parallel direction. This region is denoted as R for future reference.

As stated by Hong et al., only regions of interest (ROI) with fairly uniform intensity should be preserved. Therefore, we define a simple metric to model this desired characteristic: $u(l) = maximum(std(S_0), std(S_1))$, where $std(.)$ is a function that returns the standard deviation of a set of values, and S_0 and S_1 are the set of gray-scaled pixels belonging to each side of the ROI of the line l. The lower u is, the more uniform the ROI of the line is considered.

Before checking the second criterion, we remove all lines whose value u is greater than an empirically defined threshold. It is important to notice that Hong et al. did not define how they measured this uniformity in their work, nor did they define the ROI dimensions, which in our work was set to 10 pixels in the edge direction and 11 in the other.

To assure the second criterion, if there is any part of any line in the area R marked on our map, instead of ignoring both lines, we eliminate only the current one if its value u is greater than the one line. Figure 3.8 shows an example of the detected lines that passed both criteria and also their ROI, considered to be fairly uniform.

In the third step, we followed the exact procedure they defined and after adjusting of the box orientation, the pixels are averaged along the columns, and then the gradient of this single profile line is computed using second order accurate central differences in the inner points and first or second order accurate one-side differences at the extremities.

As for the estimation of the standard deviation of the Gaussian function that models the gradient of the edge profile curve, we chose, however, to use the non-linear least squares algorithm [35] to fit the Gaussian function using the profile gradient as input, due to the poor results we achieve with the implementation of the direct approach. Finally, we implemented the exact Equation 2.6 to compute the final metric.

So far, we have described the adapted version of Hong et al.'s BM. We now present the modified version that adds a step between the mean of the pixel values in the columns of the adjusted box and the gradient calculation: a transformation in the profile curve (that is, the result of the aforementioned averaging) that returns only the slope part plus two additional corner points. Figure 3.9 demonstrates this effect and the difference it causes in the gradient that is computed later.

The algorithm itself is simple: start at the midpoint of the profile and keep track of the sign of the differences of two consecutive points as we move in both directions. When

Figure 3.8: Example of the lines that were detected and passed all criteria along with their ROIs.

the sign changes in either side, remove all points in that direction, starting from the one which caused the sign to change. Finally, replicate the last point on both extremities, placing each as the new corresponding extremity. The purpose of this transformation is to make it easier for the Gaussian fitting method to find the correct parameters by making the gradient curve to resemble more a Gaussian curve, and at the same time, to make the computed estimate closer to the actual PSF spread parameter value.

3.3 Evaluation

In this section, we describe our methods to train the models and select the best one for further evaluation (Section 3.3.2). Moreover, we provide details on the chosen evaluation metrics (Section 3.3.1).

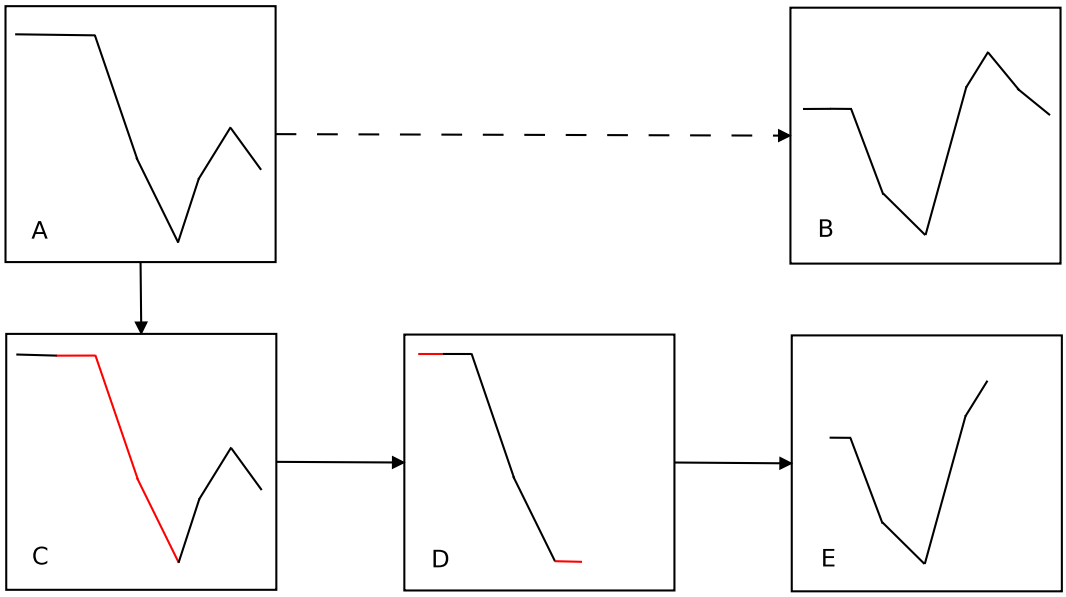


Figure 3.9: Example of the resulting profile after applying our transformation. Box A shows the original profile curve and the box B shows its corresponding gradient and therefore the dashed line exemplifies the original pipeline. The box C illustrates the detected slope highlighted in red, while in the box D, the extracted slope along with the added extremeties in red. At last, box E shows the resulting gradient curve with our modification.

3.3.1 Metrics

We present here the metrics used to guide our choice of the best classifier-hyperparameter-feature combination in the validation procedure and to evaluate the best one with the test set. Since our problem is a binary classification, we choose the metrics accordingly.

Confusion Matrix

In the binary category, the confusion matrix [39] elements have specific terms: true positive (TP), number of samples of the positive class correctly classified, true negative (TN), number of samples of the negative class correctly classified, false positive (FP), number of samples of the negative class classified as positives, and false negative (FN), number of samples of the positive class classified as negatives.

The positive class in our work is the *sharp* one, whereas the *blurred* class is the negative. Therefore, the confusion matrix is a 2D matrix in the form of Table 3.1.

Table 3.1: Example of confusion matrix for our binary problem.

		Predicted	
		Blurred	**Sharp**
Ground-Truth	**Blurred**	TN	FP
	Sharp	FN	TP

Recall, Precision and F-Score

These three metrics are derived from the confusion matrix. According to [15], but using the binary classification problem terminology, (i) recall (R) is the fraction of samples of the positive class that are classified correctly, (ii) precision (P) is the fraction of the samples classified as positives that are actually positive, and (iii) F-Score (F) is the harmonic average between the recall and precision. Formally

$$\mathrm{R} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{3.25}$$

$$\mathrm{P} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \tag{3.26}$$

$$\mathrm{F} = 2\frac{\mathrm{R} \ . \ \mathrm{P}}{\mathrm{R} + \mathrm{P}} \tag{3.27}$$

ROC Curve and AUC

The Receiver Operating Characteristic (ROC) curve shows the performance of a binary classifier as its discrimination threshold varies. Each 2D point in the curve represents the recall, also named true positive rate (TPR), and the false positive rate (FPR) of the classifier for a different threshold used in the decision function [18].

The Area Under Curve (AUC) can be understood as the integral of a curve. In our case, the ROC curve with respect to the FPR. Since we selected a discrete number of thresholds to be considered, we also have a discrete number of points in the ROC curve, so the AUC can be defined in Equation 3.28. Since recall and FPR values are in the range [0..1], the AUC values are confined to this same range

$$\mathrm{AUC} = \sum_{i}^{N} \delta_{\mathrm{FPR}} \mathrm{R}_i \tag{3.28}$$

where δ_{FPR} is the absolute difference of FPR in each consecutive pair of points, R_i is the recall of the i-th point in the ROC curve, and N is the total number of points in the ROC curve.

3.3.2 Model Training and Selection

In order to focus on the features and in the characteristics of the application itself, we chose to experiment with two traditional and widely used classifiers: SVM [11] and AdaBoost [19] with Decision Trees [42], each with its own training algorithm.

The hyperparameter training is performed with the K-fold cross-validation method [31] (with $K = 5$ in all our executions) by iteratively running for all hyperparameter combinations we define, that is, a simple brute-force grid-search method.

For SVM, we set the kernel as the Radial Basis Function (RBF) and varied the parameter C with values $\{0.01, 0.1, 1, 5\}$, and the parameter γ with values $\{0.01, \frac{1}{N}, \frac{1}{M}\}$, where N is the number of features and M is the number of samples in the training set.

For AdaBoost, we vary the maximum depth of the trees in the range [1..4], the minimum number of samples to perform a split with the values $\{4, 6, 8\}$, the number of decision trees with the values $\{11, 21, 31\}$ and the learning rate with the values $\{0.1, 0.5, 1\}$. All those constants, including the ones for the SVM were chosen as either educated guesses or refined guesses after a few iterations of the validation process.

Moreover, we try different weights for the two classes, forcing the classifiers to "pay more attention" to one or another class. The combinations of weights used are shown in Table 3.2.

Table 3.2: Class weights used in the grid search.

Class	**Weight Settings**			
Sharp	1	2	3	4
Blurred	1	1	1	1

In each of the 5 cross-validation iterations, the AUC of the ROC curve is computed and then the average of the 5 AUC values is calculated. This procedure is performed for the two classifiers in all possible combinations of the aforementioned parameters and of the three feature vectors, namely HOG, SHB and FTV.

The classifier with the highest AUC is considered the best one and chosen for the final evaluation. The results of this validation are shown in Section 4.2.

3.3.3 Model Evaluation

In this section, we list and describe all the evaluations that the chosen model has passed using the separate test set. The first one is a direct test, where we use the entire test set and computed all the metrics defined in Section 3.3.1. In the second evaluation, we select specific subsets of samples from the test set, regarding the amount of noise and compression applied to these images. The idea is to analyze the influence of these distortions on the performance of our classifier and, consequently, also the robustness of the classifier for these variations.

In the next experiment, we also select specific subsets, but this time, filtering the samples by the blurriness category, clustered into easy and hard cases. We consider the cases where the image is completely blurred or sharp as the easy cases, where the hard cases encompass the images that have sharp regions and blurred regions simultaneously.

In the fourth test, we developed a proof of concept study to analyze the following hypothesis: by reducing the region of the image used to compute the features in the bounding box of the foreground object (object of interest), the classifier performance improves considerably.

In our experiment, we used the bounding box of the foreground object used to generate the image (as explained in Section 3.1.3) and also its label, since our goal here is just a proof of concept study. On the other hand, in a real implementation, this would be done by an object detection model, such as YOLO (You Only Look Once) v.2 [43], or by a saliency detector model, such as the one described by Zhao et al. [61].

In the last evaluation, we compare our classifier against five other methods:

1. A simple sharpness metric based on the gradient of the image, which is computed using second order accurate central differences in the interior points and first or second order accurate, forward or backward, differences at the borders [50]. The final metric is given by: $\text{GSM} = \frac{1}{N}\sum_{i}\frac{G_x(i)+G_y(i)}{2}$, $\forall i \in \{0,\ 1,\ ...,\ N-1\}$, where G_x and G_y are, respectively, the absolute gradient values in the horizontal and vertical directions, and N is the total number of pixels;

2. An FTV-based metric, expressed as: $\text{FTVM} = \frac{f_{\text{FTV}}^{(0)} + f_{\text{FTV}}^{(1)}}{2}$, where $f_{\text{FTV}}^{(0)}$ and $f_{\text{FTV}}^{(0)}$ are the first and second features of the FTV feature set, that is, the variance of the Fourier coefficients of the original image in the horizontal and vertical directions, respectively;

3. Our adaptation of the model proposed by Hong et al. [22];

4. The modified version of Hong et al.'s method;

5. The method of Sieberth et al. [48].

We performed this comparison using the entire test set divided into the easy and hard sets.

Chapter 4

Experiments

This chapter describes the data used in our experiments for selecting the best features, the machine learning (ML) model and its hyperparameters, as well as for performing the subsequent evaluations of the chosen classifier, as described in Sections 3.3.2 and 3.3.3, respectively. Analysis of the results are also provided.

4.1 Hardware and Software Platform

In our experiments, we used a desktop computer with a 3.90GHz x 4 Intel(R) Core(TM) i3 7100 CPU, 8 GB RAM, running an Ubuntu 16.04 LTS 64 bits operational system.

All algorithms were implemented in Python 3.6.1, using the following libraries: Numpy [36] (version 1.12.1), Scipy [28] (version 0.19.1), Scikit-Image [53] (version 0.13.0), Scikit-Learn [40] (version 0.18.2), OpenCV [5] (version 3.1.0) and Matplotlib [24] (version 2.0.2).

4.2 Model Selection

Following the procedure described in Section 3.3.2, we compute the AUC of the ROC curve for combinations of the ML algorithm, hyperparameters and features. We summarize these results in Table 4.1, where we present the AUC of the ROC curve for the models with the best hyperparameters for each ML algorithm and feature combinations.

As shown in the table, the best classifier in the validation step was an SVM using all feature sets. It is possible to observe that all Adaboost models had a similar performance compared to the SVM models, sometimes a little better, sometimes a little worse, however, none reached the value of 0.9, which might be explained by the usage of decision trees as the base classifiers of the ensemble, since decision trees deal with features as independent variables and it may be the case that a combination of the features yields a higher discrimination power.

For instance, the first two features of the SHB feature set, which can be interpreted as a measure of sharpness of the original image and of its blurred version, respectively, are not particularly good for discriminating between our two classes independently, due to dependency of these sharpness measures in the content of the image, however, a relationship

Table 4.1: Results for AUC of the ROC curves for all best models of each ML algorithm/feature combinations.

Model	**Features**	**AUC**
SVM	FTV	0.742
SVM	SHB	0.835
SVM	HOG	0.862
SVM	FTV + SHB	0.861
SVM	FTV + HOG	0.890
SVM	SHB + HOG	0.912
SVM	**FTV + SHB + HOG**	**0.916**
Adaboost	FTV	0.750
Adaboost	SHB	0.832
Adaboost	HOG	0.768
Adaboost	FTV + SHB	0.852
Adaboost	FTV + HOG	0.841
Adaboost	SHB + HOG	0.888
Adaboost	FTV + SHB + HOG	0.891

between them may be much better for that purpose.

The ensemble of decision trees can approximate a relationship between variables by deeper trees and the way the result of the trees is combined. Additionally, decision trees can more easily handle with unbalanced data sets. Those factors certainly contributed to achieve results close to the SVMs. Although a much more precise and assertive investigation could be conducted in order to understand the performance of the different models, this is beyond the scope of this work.

As another interesting observation, one can also notice that the difference in the AUC of the ROC curve between the best model and the SVM that uses the SHB + HOG features are small. So in practice, depending on the time of execution that is added by the use of the FTV along the other feature vectors, and the time constraints of the application, the best model to chose would actually be this second best SVM model.

4.3 Selected Model

The hyperparameters of our selected SVM model with RBF kernel are $C = 5$, $\gamma = \frac{1}{N}$, where N is the number of features, whereas the class weight for the *sharp* class was set to 2 and for the *blurred* class to 1.

Once selected, we retrain the best model using the entire training set and evaluated it with the test set using the metrics described in Section 3.3.1. These results are shown in Tables 4.2 and 4.3, as well as illustrated in Figure 4.1.

Our method achieved an F1-score of 0.871 and an AUC of the ROC curve of 0.954. Its predictions for the *blurred* class are more reliable since the precision and recall rates are higher for this class than for the *sharp* class. The reason for this may be related to the imbalance in terms of number of samples per class, although we have tried to address this

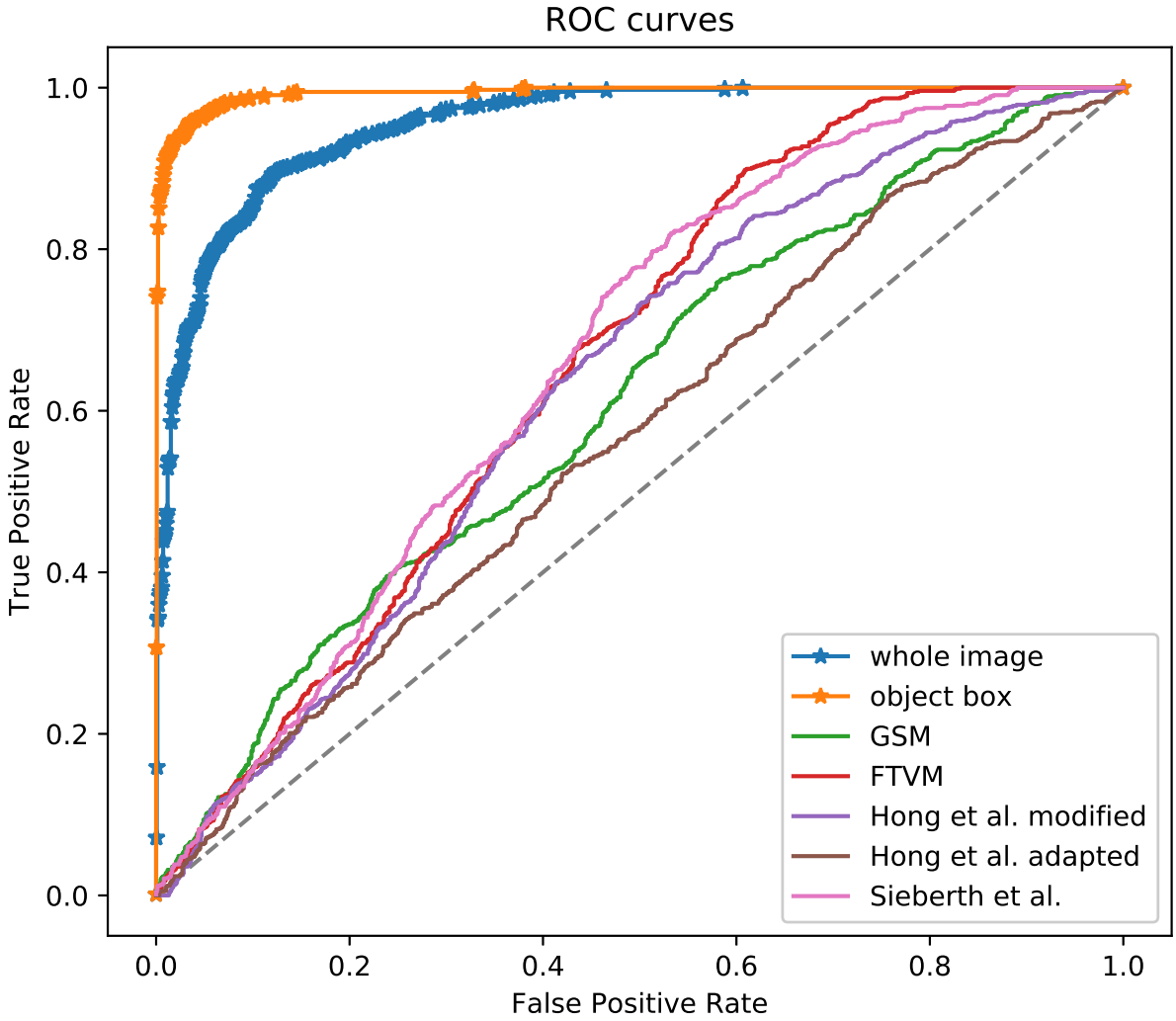


Figure 4.1: Comparison of ROC curves between our trained models and other blurriness/sharpness scores. From top to bottom: (i) our selected SVM models (lines with stars) trained with features from the entire image and only from the object bounding box, respectively, (ii) the GSM metric, (iii) the FTVM metric, (iv) the modified version of Hong et al.'s method [22], (v) its adaptation, and (vi) Sieberth et al.'s method [48].

Table 4.2: Results from evaluating the best model from the model selection step in the test set.

	Blurred	**Sharp**	**Average**
Precision	0.925	0.808	0.867
Recall	0.898	0.855	0.877
F1-Score	0.912	0.831	0.871

issue by setting different class weights during the training process. Further investigation would be needed to reach a more accurate answer.

In Figures 4.2 to 4.5, we show examples of the extreme cases for our model, while we try to point out some of the aspects of the images that may have led the model to these results. In Figure 4.2, some cases are shown where the classifier predicted that they were

Table 4.3: Confusion matrix for the best model from the model selection step evaluated in the test set.

		Predicted	
		Blurred	**Sharp**
Ground-Truth	**Blurred**	1359	153
	Sharp	109	647

sharp when actually blurred. We can notice that most of them are low light scenarios, some have the highest noise level, all of them have a clear background, while only the subject is blurred and some foreground objects are not in the center of the image.

Figure 4.2: The six blurred images with the highest scores from the SVM.

For the errors made by the classifier with samples of the sharp class, shown in Figure 4.3, it is possible to mention that some objects are not in the center, some background scenarios are actually blurred and none of them has the highest noise level. However, some images such as (a), (b) and (f) are completely sharp with the object in the center, which makes it difficult to understand the model error. A more in-depth analysis of the features of these images and how they are used in the model could tell us more, however, since the model is an SVM with the RBF as kernel, this type of mapping is unfeasible to achieve.

Figure 4.4 illustrates the cases where the classifier correctly predicted that the samples belong to the blurred class. There are some low light scenes, all of them are completely blurred, most subjects are in the center of the images and none of them has the highest noise level. Finally, we plot in Figure 4.5 some examples of images successfully classified

(a)
combination_1300_sharp.jpg
score = -1.3482784719103975
noise = 1

(b)
combination_1637_sharp.jpg
score = -1.3039875264886707
noise = 2

(c)
combination_1001_sharp.jpg
score = -1.0336832718484539
noise = 1

(d)
combination_985_sharp.jpg
score = -0.9403466427388087
noise = 2

(e)
combination_353_sharp.jpg
score = -0.9085517002360852
noise = 1

(f)
combination_989_sharp.jpg
score = -0.9080864010790846
noise = 2

Figure 4.3: The six sharp images with the lowest scores from the SVM.

(a)
combination_3064_blur.jpg
score = -3.8965500371395128
noise = 2

(b)
combination_1222_blur.jpg
score = -3.570527038303336
noise = 1

(c)
combination_2898_blur.jpg
score = -3.556536317903633
noise = 2

(d)
combination_2735_blur.jpg
score = -3.5390515438938847
noise = 1

(e)
combination_408_blur.jpg
score = -3.456724901133591
noise = 1

(f)
combination_2568_blur.jpg
score = -3.413850520229762
noise = 1

Figure 4.4: The six blurred images with the lowest scores from the SVM.

as sharp, where they all have the foreground object close to the center of the image, none of them has low light scenes, none of them has level 1 of noise, while most of them have

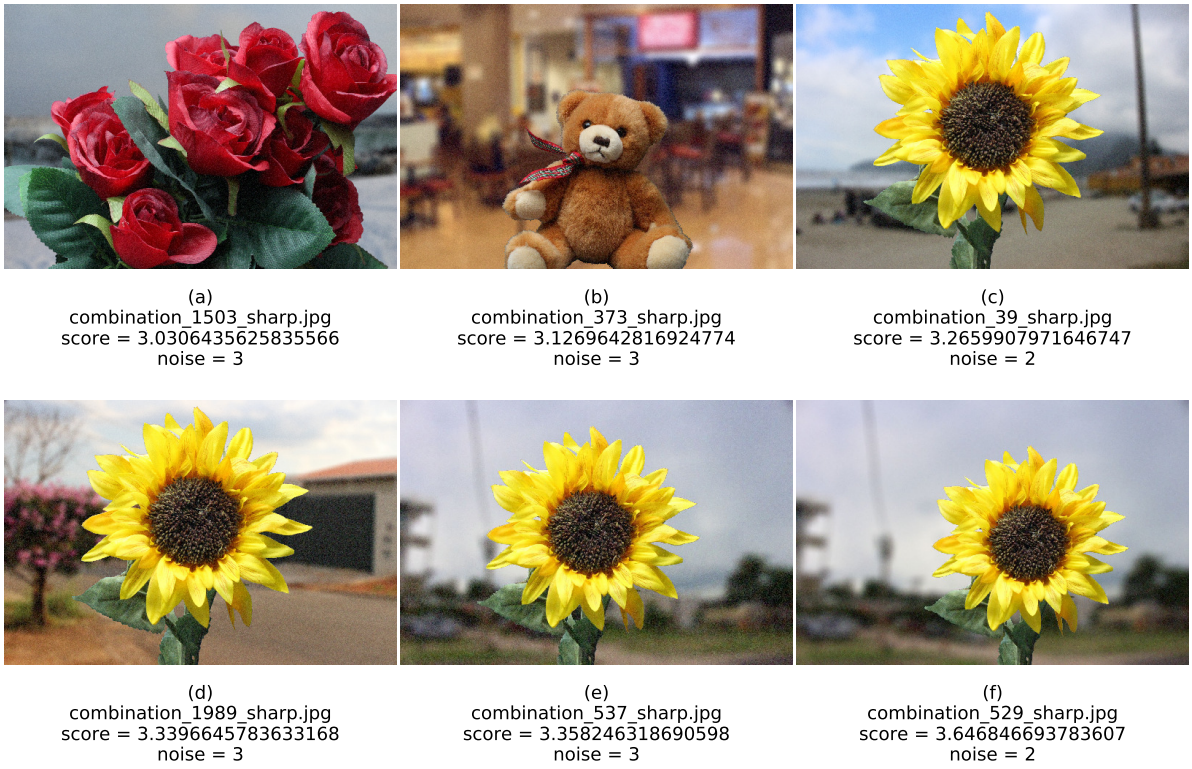


Figure 4.5: The six sharp images with the highest scores from the SVM.

the highest noise level and 4 out of 6 have the sunflower as the foreground object.

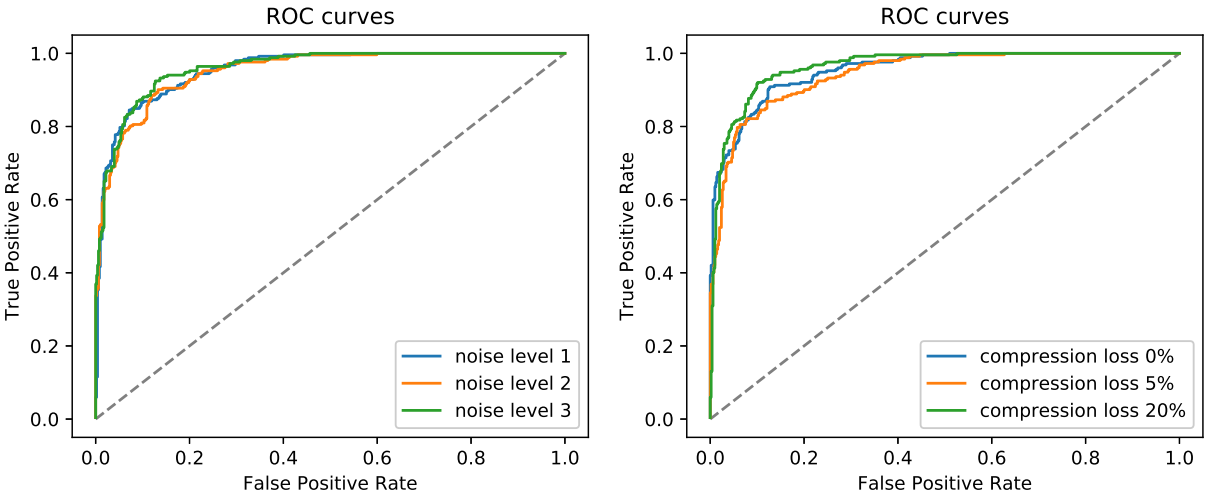
4.4 Results for Noise and Compression Robustness

As mentioned in Section 3.3.3, we created six different subsets of the test set according to the level of noise and compression used in the generation of these images: three subsets for each level of noise and three subsets for each level of compression. Then, we test the chosen classifier with each of these subsets, whose results are presented in Table 4.4 and illustrated in Figure 4.6.

Table 4.4: Performance of the classifier for different levels of noise and compression.

	Noise Level			**Compression**		
	1	**2**	**3**	**0 %**	**5 %**	**20 %**
Precision	0.889	0.850	0.864	0.859	0.858	0.885
Recall	0.886	0.858	0.886	0.876	0.865	0.889
F1-Score	0.888	0.854	0.873	0.866	0.861	0.887

It is possible to observe very small variation from one subset to the other, even tough one would expect the performance to drop as the level of noise and compression increases. Since we have samples with the same amount of noise and compression in the training set as we have in the testing set, in addition to the use of a denoising function in the pre-processing steps, our model learned how to combine the features so that robustness



(a) (b)

Figure 4.6: ROC curves for the best model per amount of (a) noise and (b) compression applied in the generation of the test images.

to these levels of distortion has been achieved.

4.5 Per Category of Blurriness Analysis

In this experiment, we split the test set into two subsets according to the type of blurriness applied to the images, placing in the first subset the cases in which we estimate their classification to be easier. In this subset, we include all cases where the entire image was equally blurred or none of it was. In the second subset, we put the cases where only one region of the image was blurred (either the background or the foreground), which are considered more difficult in our judgment due to the presence of blurred and sharp regions in the image simultaneously, with their locations and intensity dependent highly on the image content.

We then run the chosen model on these two disjoint subsets and evaluated their predictions, whose results are shown in Tables 4.5, 4.6 and 4.7. It is possible to observe that our estimate is confirmed by the results, since the classifier performed considerably better in the estimated "easier" cases, achieving an F1-score of 0.928, on average for both classes, while only obtaining 0.819 for the same metric in the "harder" cases. In addition, it is perceived that it greatly increases the amount of false positives for the sharp class, thus consequently decreasing its precision.

Table 4.5: Confusion matrix for the easier cases of the test set.

		Predicted	
		Blurred	**Sharp**
Ground-Truth	**Blurred**	745	11
	Sharp	59	319

Table 4.6: Confusion matrix for the harder cases of the test set.

		Predicted	
		Blurred	**Sharp**
Ground-Truth	**Blurred**	614	142
	Sharp	50	328

Table 4.7: Performance of the classifier for different categories of blurriness, grouped by difficulty.

	Easier cases	**Harder cases**
Precision	0.946	0.811
Recall	0.914	0.839
F1-Score	0.928	0.819

4.6 Analysis of the Possible Combination with an Object Detector

Based on the results presented in the previous section, we conjecture that segmenting the image, extracting only the region of interest (ROI), would greatly improve the performance of a classifier, as doing so would make the "harder" cases seem more like the "easier" cases, since most of the images would then either be blurred or sharp. In order to evaluate this hypothesis, we segment all images based on the bounding box of the foreground object (when there is one), which is known to us when we combine the foreground and background images to generate the final one. Then, we compute the features only using this region, train again the same classifier (with the parameters chosen in the validation process) and evaluate it in the test set. It is important to mention that, since we are using the ground-truth bounding boxes, these results represent a proof of concept and an upper bound for our classifier when using rectangular bounding boxes to segment the image targeting the object of interest.

Tables 4.8 and 4.9 present the results for the performance of such a classifier, where we can clearly notice a considerable improvement in the predictions when compared to the first classifier, yielding an F1-score of 0.958 (on average, for both classes) against 0.871 of the previous SVM. It also obtained an AUC of the ROC curve equal to 0.993 versus 0.954 of the former. Figure 4.1 illustrates how much improvement is achieved with this technique, comparing both approaches at the theoretical level, where the orange line with stars corresponds to the SVM trained with the features extracted only from the ROI, whereas the blue line corresponds to the first SVM, trained with the features from the entire image.

In practice, an object detector would be used to acquire the bounding box information. In addition, a saliency map generator/detector could achieve even higher performance as it attempts to perfectly match the border/contour of the object (as opposed to a rectangular bounding box), decreasing the number of pixels that do not belong to the object of interest

Table 4.8: Results for the SVM model trained with the features extracted only from the object bounding boxes.

	Blur	**Sharp**	**Average**
Precision	0.969	0.950	0.959
Recall	0.975	0.937	0.956
F1-Score	0.972	0.944	0.958

Table 4.9: Confusion matrix for the SVM model trained with the features extracted only from the object bounding boxes.

		Predicted	
		Blur	**Sharp**
Ground-Truth	**Blur**	1475	37
	Sharp	47	709

and, therefore, may negatively affect the feature values.

These results contribute to show how complex it really is to automatically determine if an image is blurred or sharp, since it depends on its content, whose domain is huge and, therefore, also emphasizes the need to use methods capable of analyzing, or at least, responding differently according to the image content.

4.7 Comparison with Different Methods

The first results of this comparison are shown in Figure 4.1, where it is possible to see that our trained SVM performed significantly better than all direct scores/metrics/features (that is, functions defined without any kind of training) with which we compared. Besides, it can also be noticed that our adaptation of Hong et al.'s method [22] did not outperform simpler methods such as GSM and FTVM, while the modified version performed slightly better than GSM. Unfortunately, due to the lack of important implementation details in Hong et al.'s paper and the explicit difference in our version (explained in Section 3.2.3), we cannot guarantee a perfect correspondence between our implementation and the original algorithm.

From Figure 4.7, which depicts the ROC curves calculated by applying our trained SVM and other direct functions to the images of the easier and harder cases separately, we can observe that our selected model achieves a much higher robustness for the harder cases than any of the direct functions, as well as outperformed all of them for easier cases. Another interesting aspect shown in these results is that the adaptation of the Hong et al.'s method was surpassed by the modified version, helping to confirm our hypothesis about the additional step we created. In addition, the direct functions present a similar behavior, that is, good performance for easier cases and very poor one for harder cases. Finally, it is possible to observe that the curve for the easier cases of Sieberth et al.'s method is very close to that of the SVM model.

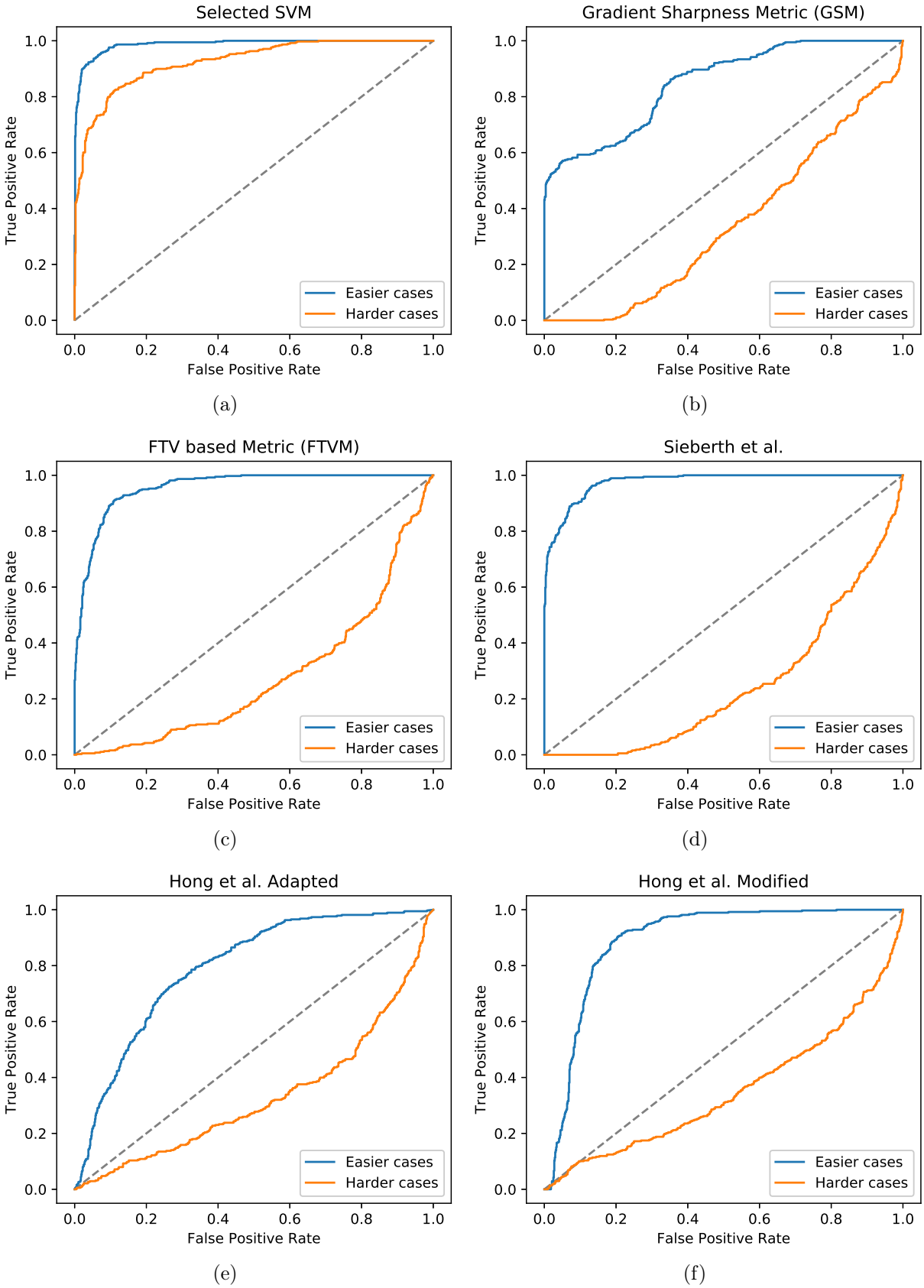


Figure 4.7: ROC curves of the scores obtained from the easier and harder cases, separately.

We also conjecture that one reason our SVM model has achieved such performance, especially for harder cases, is that it has learned some semantic information about the scenes. More precisely, it learned to identify, at some level, the foreground objects present in the scenes and, thus, map the features to the correct classes based on this knowledge. This is probably thanks to the HOG features, which is known to work on object recognition problems. An undesirable consequence of this is that it would possibly not work well on a different data set, composed of different objects.

Semantic information is very important in this application because of how blurred and sharp images have been defined, where it depends directly on a specific region of the image, that is, the foreground object, which is (or may be) very different in all the images. Thus, on the other hand, this hipothesis also explains why the direct methods have performed so poorly in the harder cases, since they do not take into account any semantic information.

Chapter 5

Conclusions and Future Work

Automatic classification of images as blurred or not, as a digital camera user would do, that is, considering which part of the image is blurred and which is not, is a challenging and useful task. For example, it can help users filter a large number of pictures and show them the blurred ones, so they can delete those images. However, this is not easily accomplished due to the subjectivity of what is important, which also implies that a good classifier must understand the image content and the preference of one or more human users.

Due to the lack of a publicly available data set during the development of our work, we have created a new framework capable of generating a set of images to simulate multiple real-world conditions, such as defocus blur, camera and object motion blur, and different levels of noise and compression. In addition, it can be extended to include new transformations to simulate different conditions. Using this framework, we generated 14,742 images, automatically labeled as sharp or blurred, depending on the state of the main object in the foreground.

To help achieve our goal, we developed two novel features, namely Sharpness Behavior (SHB) – with five descriptors – and Fourier Transform Variance (FTV) – with four descriptors. These features are based on image processing techniques to extract information related to the blurriness of images, combined with the idea that an image, when artificially blurred, changes little if it is already blur, whereas the changes are considerably larger if it they are initially sharp.

We then used these features along with the Histogram of Oriented Gradient (HOG) features to train different SVM and AdaBoost models with different combinations of feature sets and hyper-parameters. We applied the grid search algorithm and K-fold cross-validation on the training dataset to choose the best final model, which, in our case, turned out to be an SVM using all feature sets.

In the last part of our methodology, we evaluated this final model using our test set, where it achieved an F1-Score of 0.871 and an AUC of the ROC curve of 0.954. We analyzed the impact of different levels of noise and compression on the results of our model, where we found that the model is, in fact, robust for different degrees of noise and compression, since its performance is very similar for all levels tested. In the following experiment, we divided the test set into "easy" cases (totally blurred or sharp images) and "hard" cases (images with blurred and sharp regions simultaneously).

The model was evaluated with each of these disjoint sets. The results confirmed our hypothesis, since the classifier performed much better in "easy" cases with an F1-Score of 0.928, whereas the same metric for the "hard" cases was 0.819. The hypothesis that the model would achieve better results if it knew where the object of interest was before extracting the features was then evaluated by extracting the features only from the bounding box of the main object in the foreground. Again, our conjecture proved to be true when the model achieved an F1-Score of 0.958 and an AUC of the ROC curve of 0.993.

In the last evaluation procedure, the results of our model were compared with different features / metrics, where most of them aimed at different but closely related applications. The ROC curves for the "easy" and "hard" cases were used for comparison, where the superiority of our method was evident, which was not surprising, since these methods were neither trainable nor designed for such application.

The results presented in this work demonstrated that simple features and metrics cannot be successful in this application because they do not carry semantic image information, which is fundamental due to the nature of the application. On the other hand, our trained SVM was able to learn some semantic information from the training set, achieving a good performance in the test set, confirming the main hypothesis of this work and achieving all of our objectives. It is likely, however, that our model would not work well in a different set of images, where each class depends on what is considered the region or object of interest in each image of the data set.

As directions for future work, we intend to test our model in the recently published data set [60] and evaluate the use of Convolutional Neural Networks (CNNs) [38, 57, 60] in the classification process.

Bibliography

[1] N. Ahmed and K. R. Rao. *Fast Fourier Transform*, pages 54–84. Springer Science & Business Media, Berlin, Heidelberg, 2012.

[2] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool. Speeded-Up Robust Features (SURF). *Computer Vision and Image Understanding*, 110(3):346–359, 2008.

[3] Blurred Image Detection, 2018. `https://github.com/ph-hack/bid`.

[4] D. B. L. Bong and B. E. Khoo. Objective Blur Assessment based on Contraction Errors of Local Contrast Maps. *Multimedia Tools and Applications*, 74(17):7355–7378, 2015.

[5] G. Bradski and A. Kaehler. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[6] R. Brinkmann. *The Art and Science of Digital Compositing: Techniques for Visual Effects, Animation and Motion Graphics*. Academic Press, 2008.

[7] J. Canny. A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, Nov. 1986.

[8] C. Chen, W. Chen, and J. A. Bloom. A Universal Reference-Free Blurriness Measure. In *Proceedings of SPIE*, volume 7867, pages 1–14, 2011.

[9] J. Chen, L. Yuan, C.-K. Tang, and L. Quan. Robust Dual Motion Deblurring. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, June 2008.

[10] M.-J. Chen and A. C. Bovik. No-Reference Image Blur Assessment using Multiscale Gradient. In *International Workshop on Quality of Multimedia Experience*, pages 70–74, July 2009.

[11] C. Cortes and V. Vapnik. Support-Vector Networks. *Machine Learning*, 20(3):273–297, 1995.

[12] F. Crété-Roffet, T. Dolmiere, P. Ladret, and M. Nicolas. The Blur Effect: Perception and Estimation with a New No-Reference Perceptual Blur Metric. In *SPIE Electronic Imaging Symposium on Human Vision and Electronic Imaging*, volume 6492. International Society for Optics and Photonics, 2007.

[13] N. Dalal and B. Triggs. Histograms of Oriented Gradients for Human Detection. In *IEEE International Conference on Computer Vision and Pattern Recognition*, pages 886–893, 2005.

[14] E. R. Davies. *Computer and Machine Vision: Theory, Algorithms, Practicalities*. Academic Press, 4th edition, 2012.

[15] J. Davis and M. Goadrich. The Relationship Between Precision-Recall and ROC Curves. In *23rd International Conference on Machine Learning*, pages 233–240, Pittsburgh, PA, USA, 2006. ACM.

[16] O. Déniz, G. Bueno, J. Salido, and F. D. la Torre. Face recognition using Histograms of Oriented Gradients. *Pattern Recognition Letters*, 32(12):1598–1603, 2011.

[17] R. O. Duda and P. E. Hart. Use of the Hough Transformation to Detect Lines and Curves in Pictures. *Communications of the ACM*, 15(1):11–15, Jan. 1972.

[18] T. Fawcett. An introduction to ROC Analysis. *Pattern Recognition Letters*, 27(8):861–874, 2006.

[19] Y. Freund and R. E. Schapire. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[20] S. A. Golestaneh and L. J. Karam. Spatially-Varying Blur Detection Based on Multiscale Fused and Sorted Transform Coefficients of Gradient Magnitudes. *CoRR*, abs/1703.07478, 2017.

[21] R. C. Gonzalez and R. E. Woods. *Digital Image Processing*. Pearson, 4 edition, 2017.

[22] Y. Hong, G. Ren, E. Liu, and J. Sun. A Blur Estimation and Detection Method for Out-of-Focus Images. *Springer Science and Business Media New York*, i75(18):10807–10822, 2015.

[23] P. Hsu and B.-Y. Chen. Blurred Image Detection and Classification. In S. Satoh, F. Nack, and M. Etoh, editors, *Advances in Multimedia Modeling*, pages 277–286, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[24] J. D. Hunter. Matplotlib: A 2D Graphics Environment. *Computing in Science and Engineering*, (9):90–95, 2007.

[25] International Organization for Standardization. ISO 12232: Photography - digital still cameras - determination of exposure index, ISO speed ratings, standard ouptut sensitivity, and recommended exposure index. Technical report, ISO, International Organization for Standardization, Switzerland, 4 2006.

[26] ITU-R, The Radiocommunication Sector of the International Telecommunication Union. Recommendation ITU-R BT.709: Parameter Values for the HDTV for Production and International Programme Exchange. Technical report, International Telecommunication Union (ITU), 1990.

[27] H. K. Jerome Da Rugna. Automatic Blur Detection for Metadata Extraction in Content-based Retrieval Context. In *Procedings of SPIE*, volume 5304, pages 1–10, 2003.

[28] E. Jones, T. Oliphant, and P. Peterson. SciPy: Open Source Scientific Tools for Python, 2018. `http://www.scipy.org/`.

[29] N. Joshi, S. B. Kang, C. L. Zitnick, and R. Szeliski. Image Deblurring Using Inertial Measurement Sensors. *ACM Transactions on Graphics*, 29(4):30:1–30:9, July 2010.

[30] S. Kimball, P. Mattis, and M. Natterer. GIMP, The GNU Image Manipulation Program, 1998. `http://www.gimp.org`.

[31] R. Kohavi. A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. In *International Joint Conference on Artificial Intelligence*, volume 14, pages 1137–1145, 1995.

[32] A. Levin. Blind Motion Deblurring Using Image Statistics. In *19th International Conference on Neural Information Processing Systems*, pages 841–848, Canada, 2006. MIT Press.

[33] G. LLC. Google Search Help: Find Free-to-Use Images, 2018. `https://support.google.com/websearch/answer/29508?hl=en-BR&ref_topic=3180360`.

[34] P. Marziliano, F. Dufaux, S. Winkler, and T. Ebrahimi. A No-Reference Perceptual Blur Metric. In *IEEE International Conference on Image Processing*, volume 3, pages III–57–III–60, 2002.

[35] J. J. Moré. The Levenberg-Marquardt Algorithm: Implementation and Theory. *Numerical Analysis*, pages 105–116, 1978.

[36] T. E. Oliphant. A Guide to Numpy. USA: Trelgol Publishing, 2006.

[37] E. Ong, W. Lin, Z. Lu, X. Yang, S. Yao, F. Pan, L. Jiang, and F. Moschetti. A No-Reference Quality Metric for Measuring Image Blur. In *Seventh International Symposium on Signal Processing and Its Applications*, volume 1, pages 469–472, July 2003.

[38] J. Park, Y.-W. Tai, D. Cho, and I. S. Kweon. A Unified Approach of Multi-scale Deep and Hand-crafted Features for Defocus Estimation. *CoRR*, abs/1704.08992, 2017.

[39] J. Parker. Rank and Response Combination from Confusion Matrix Data. *Information Fusion*, 2(2):113–120, 2001.

[40] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and Édouard Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, (12):2825–2830, 2011.

[41] J. B. Philips and H. Eliasson. *Camera Image Quality: Benchmarking*. John Wiley and Sons Ltd, 2018.

[42] J. Quinlan. Simplifying Decision Trees. *International Journal of Man-Machine Studies*, 27(3):221–234, 1987.

[43] J. Redmon and A. Farhadi. YOLO9000: Better, Faster, Stronger. *CoRR*, abs/1612.08242, 2016. `http://arxiv.org/abs/1612.08242`.

[44] B. Richard. *Signals in the Presence of Noise*, chapter 121. American Cancer Society, 2005.

[45] Q. Shan, J. Jia, and A. Agarwala. High-quality Motion Deblurring from a Single Image. *ACM Transactions on Graphics*, 27(3):73:1–73:10, Aug. 2008.

[46] J. Shi, L. Xu, and J. Jia. Discriminative Blur Detection Features. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2965–2972, June 2014.

[47] J. Shi, L. Xu, and J. Jia. Just Noticeable Defocus Blur Detection and Estimation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 657–665, June 2015.

[48] T. Sieberth, R. Wackrow, and J. H. Chandler. Automatic Detection of Blurred Images in UAV Image Sets. *ISPRS Journal of Photogrammetry and Remote Sensing*, 122:1–16, Dec. 2016.

[49] C. Tang, C. Hou, and Z. Song. Defocus Map Estimation from a Single Image via Spectrum Contrast. *Optics Letters*, 38(10):1706–1708, May 2013.

[50] The SciPy Community. Numpy Reference: numpy.gradient, 2018. `https://docs.scipy.org/doc/numpy/reference/generated/numpy.gradient.html`.

[51] H. Tong, M. Li, H. Zhang, and C. Zhang. Blur Detection for Digital Images using Wavelet Transform. In *IEEE International Conference on Multimedia and Expo*, volume 1, pages 17–20, June 2004.

[52] P. A. Torrione, K. D. Morton, R. Sakaguchi, and L. M. Collins. Histograms of Oriented Gradients for Landmine Detection in Ground-Penetrating Radar Data. *IEEE Transactions on Geoscience and Remote Sensing*, 52(3):1539–1550, Mar. 2014.

[53] S. van der Walt, J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart, and T. Yu. Scikit-image: Image Processing in Python. *PeerJ*, 2(e453), 2014.

[54] D. A. van Dyk and X.-L. Meng. The Art of Data Augmentation. *Journal of Computational and Graphical Statistics*, 10(1):1–50, 2001.

[55] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image Quality Assessment: from Error Visibility to Structural Similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, Apr. 2004.

[56] N. Wiener. *Extrapolation, Interpolation, and Smoothing of Stationary Time Series*. The MIT Press, 1964.

[57] R. Yan and L. Shao. Blind Image Blur Estimation via Deep Learning. *IEEE Transactions on Image Processing*, 25(4):1910–1921, Apr. 2016.

[58] X. Yang, C. Zhang, and Y. Tian. Recognizing Actions Using Depth Motion Maps-based Histograms of Oriented Gradients. In *20th ACM International Conference on Multimedia*, pages 1057–1060, Nara, Japan, 2012. ACM.

[59] X. Yi and M. Eramian. LBP-Based Segmentation of Defocus Blur. *IEEE Transactions on Image Processing*, 25(4):1626–1638, Apr. 2016.

[60] S. Zhang, X. Shen, Z. Lin, R. Mech, J. Costeira, and J. Moura. Learning to Understand Image Blur. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 6586–6595. IEEE/CVF, 2018.

[61] R. Zhao, W. Ouyang, H. Li, and X. Wang. Saliency Detection by Multi-Context Deep Learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1265–1274, June 2015.

[62] L. Zhong, S. Cho, D. Metaxas, S. Paris, and J. Wang. Handling Noise in Single Image Deblurring Using Directional Filters. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 612–619, June 2013.

[63] S. Zhuo and T. Sim. Defocus Map Estimation from a Single Image. *Pattern Recognition*, 44(9):1852–1858, 2011.