Mathematisch-
Naturwissenschaftliche
Fakultät

Human-Computer Interaction

Bachelorarbeit

Evaluation and implementation of proven real-time image processing features for use in web browsers

Eberhard Karls Universität Tübingen
Mathematisch-Naturwissenschaftliche Fakultät
Wilhelm-Schickard-Institut für Informatik
Human-Computer Interaction
Amr Abdellatif, amr.abdellatif@student.uni-tuebingen.de, 2020

Bearbeitungszeitraum: von 27.07.2020 bis 16.10.2020

Betreuer/Gutachter: Prof. Dr. Enkelejda Kasneci, Universität Tübingen
Zweitgutachter: Dr. Wolfgang Fuhl, Universität Tübingen

Selbstständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Bachelorarbeit selbständig und nur mit den angegebenen Hilfsmitteln angefertigt habe und dass alle Stellen, die dem Wortlaut oder dem Sinne nach anderen Werken entnommen sind, durch Angaben von Quellen als Entlehnung kenntlich gemacht worden sind. Diese Bachelorarbeit wurde in gleicher oder ähnlicher Form in keinem anderen Studiengang als Prüfungsleistung vorgelegt.

Amr Abdellatif (Matrikelnummer 4145178), October 30, 2020

Abstract

We explored the requirement of proven features for real-time use in web browsers, adopting a linear SVM based face detection model as a test case to evaluate each descriptor with appropriate parameters. After checking multiple feature extraction algorithms, we decided to study the following four descriptors Histogram of oriented gradients, Canny edge detection, Local binary pattern, and Dense DAISY . These four descriptors are used in various computer vision tasks to offer a wide range of options. We then investigated the influence of different parameters as well as dimension reduction on each descriptor computational time and its ability to be processed in real-time. We also evaluated the influence of such changes on the accuracy of each model.

Experiments were performed on the publicly available FDDB data set. Moreover, the results showed that all descriptors could be used in real-time.

Acknowledgments

I wish to express my sincere gratitude to the Human-Computer Interaction department at the University of Tuebingen for the offering of such an in-depth bachelor thesis. I truly appreciate the new fundamental knowledge I gained during the last months. I am also incredibly thankful to Dr. Wolfgang Fuhl for dedicating his time and support during this thesis process ;-).

Last but not least, I am also sincerely grateful to Skimage and Sklearn development teams for offering such precise and detailed documentation and debugging systems. You guys are a lifesaver.

Contents

1 Introduction

Advances in web technology have enabled exciting new applications. These advances are also impacted by modern hardware development. Web applications are generally defined as cross-platform since they are accessible from various web browsers within different operating systems which makes them a perfect fit for user studies [1] and marketing research [2]. New technologies such as eye tracking [3] or gesture control [4] can also be used for this purpose. Eye Tracking has the advantage that it not only provides the eye signal [5, 6, 7], but also allows you to access much more information about the user. This would be the pupil diameter [8, 9], which provides information about cognitive states, attention [10, 11], sequences of eye movements [12, 13] and the different types of eye movements [14, 15, 16, 17, 14] as well es information extracted from the eye lids [18, 19, 20].

Today, web browsers run on diverse hardware types, from smartphones and tablet PCs to desktop computers. With this web application diversity, users can avoid the hassle and memory usage of installation software on every device. Users will also find web applications less demanding on older or low spec devices but those applications also rise privacy concerns [21].

Despite such rapid progress in web technology, computer vision processing on the web browsers has not been a common practice yet [22]. Computer vision usually has a high computational [23] cost due to

- Images with high resolution and high frame rates require a sheer amount of computation
- Complex algorithms to process and understand the visual data
- Real-time requirements for interactive applications

Real-time applications are standard today as they provide faster achievement of tasks like semantic segmentation [24, 25, 26] on modern hardware. Real-time applications are also subject to time constraints so that receiving data, processing them, and returning the responses are often understood to be in milliseconds [27, 28]. The general paradigm of web-based application development is deploying computationally complex logic on the server. However, with recent progress in machine learning algorithms [29, 30, 31] and the hardware technology on the client-side, web clients can handle more demanding tasks. Still, heavier workloads are augmented by using edge or cloud services.

Cameras and digital image processing have become essential tools in our modern life with many applications. Real-time image processing is an exciting software and hardware challenge as well as the validation of machine learning approaches [32, 33]. By extracting features in real-time, we reduce the amount of information that needs to be processed. These new reduced sets of features should summarize most of the information in the original set of features.

In this thesis, we describe multiple image processing feature algorithms. We also present its main characteristics and discuss its performance in terms of accuracy and execution time. In this context, accuracy is the system error rate, whereas execution time measures its speed.

2 Image processing background

The field of digital image processing is an extensive one, retaining digital signal processing techniques in addition to techniques specific to images. Its use has been increasing exponentially over the last decades. Digital image processing consists of manipulating images to obtain enhanced ones or extract useful information from them [34, 35, 36].

An image is a composition of picture elements (pixel) that could be represented as a 2D-function f(x, y), where x and y are the coordinates of that element and an amplitude value f determining its intensity [37]. For an image to be processed digitally, it has to be sampled and transformed into a matrix of finite precision numbers. Digital image processing consists of manipulating those finite precision numbers so that the input data is an image, and the output could be image, characteristics, features, or visual words associated with the image.

Digital image processing is one of the modern information society pillars with various applications from medicine to entertainment and multimedia systems, from autonomous systems to geological processing and remote sensing [38].

2.1 Pre-processing

2.1.1 Dimension reduction

Understanding a large quantity of multidimensional data requires extracting information out of them [39]. Modern applications have steadily extended their use of complex, high dimensional data. However, the particularity of analyzing a very high dimensional data set is usually computationally intractable. Another problem with high-dimensional data sets is that, in many cases, not all the measured details are essential for understanding the underlying region of interest [40].

While specific computationally expensive methods can construct predictive models with high accuracy from high-dimensional data, it is still a prerequisite in many applications, especially real-time applications, to reduce the original data dimension before modeling the data.

Dimension reduction aims to translate high dimensional data to a low dimensional representation so that the low-dimensional representation retains some meaningful properties of the original data [41].

Cropping and resizing Region of Interest (ROI) extraction is a crucial step for training a recognition system. ROI extraction aims to decide which part of the image is suitable for feature extraction, reducing unnecessary input data [42].

Resizing is the next step after selecting ROI, as the training data set does not typically have the same dimensions for all data points. See figure 2.1.

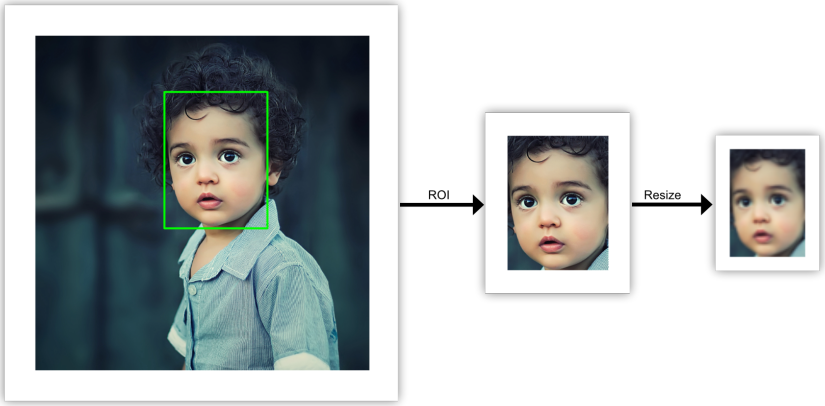

Figure 2.1: Selecting the region of interest and resizing example [L]

Gray-scale transformation Modern Descriptor Based Approaches for image recognition systems often rely on gray scale images instead of color images, as gray scale simplifies the feature extraction algorithm and therefore reduces the computational power required. Additionally, color may also introduce unnecessary information that could increase the amount of training data required to achieve good performance.[43]

Figure 2.2: Luminance of an RGB image.

2.2 Features

Visual features (also known as visual descriptors) play a significant role in the field of image processing. Many different techniques for describing local image regions have been developed. The most straightforward descriptor is a vector of image pixels.

Features describe the relevant form of information in a sample so that the task of classifying the sample is made easy by a formal procedure. Feature extraction is done after applying various image preprocessing techniques such as resizing, thresholding, and normalization to the sampled image. In image processing and pattern recognition, feature extraction is a unique form of dimensionality reduction (i.e., feature extraction seeks to obtain the most relevant information from the original data and represent that information in a lower dimensionality space). Selecting the most appropriate and robust features is a critical step in the process of classification problems [44].

"Features should contain information required to distinguish between classes, be insensitive to irrelevant variability in the input, and also be limited in number to permit efficient computation of discriminant functions and to limit the amount of training data required." [45]

In the following, the features' details are represented for all the features used in our evaluation.

2.2.1 Histograms of oriented gradients

Histogram of oriented gradients (HOG) [46] and their extensions [47] are a feature descriptor used to detect computer vision and image processing objects. The HOG descriptor technique counts gradient orientation occurrences in localized portions of an image, detection window, or region of interest (ROI) [48]. This concept of dense and local histograms of oriented gradients is a method introduced by Dalal and Triggs [49]. A standard HOG implementation follows the following steps.

Compute gradients of the image: The first step is to compute the gradients. The gradient of an image is a directional variation in the intensity or color in an image. Gradient images are generated by convolving the original image with a filter (e.g., Sobel filter), which provides two central pieces of information. The gradient's magnitude shows how quickly the image is changing, while direction indicates the direction in which the image is changing most rapidly [50]. Since the gradient may differ at every location in an image, it is common to encode the direction and magnitude in a vector. The length of this vector provides the gradient's magnitude, while its direction gives the gradient direction. The most common gradient computing method is to apply 1D centered point discrete derivative mask

in both horizontal and vertical directions to get the full range of direction [49]. See Figure 2.3

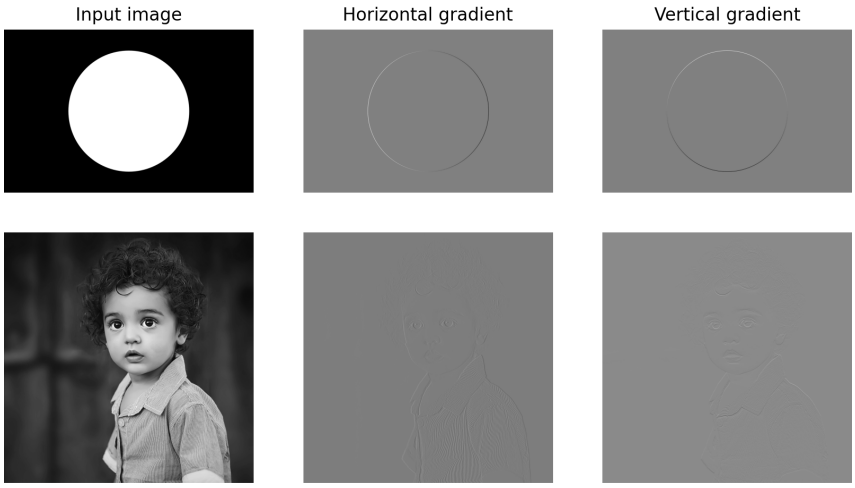

Figure 2.3: Visual representation of horizontal and vertical gradients [L]

build gradient histograms: The second stage aims to produce an encoding receptive to local image content while remaining resistant to small pose or appearance changes. The image is divided into cells. A cell can be described as a spatial region such as a square with predefined pixel size. For each cell, a local 1D histogram of gradient is accumulated over all the pixels of the cell. This combined cell-level histogram forms the underlying "orientation histogram" representation. Each orientation histogram separates the gradient angle range into a fixed number of bins. The gradient magnitudes of the pixels in each cell are the votes for each bin [49, 46, 48]. See Figure 2.4.

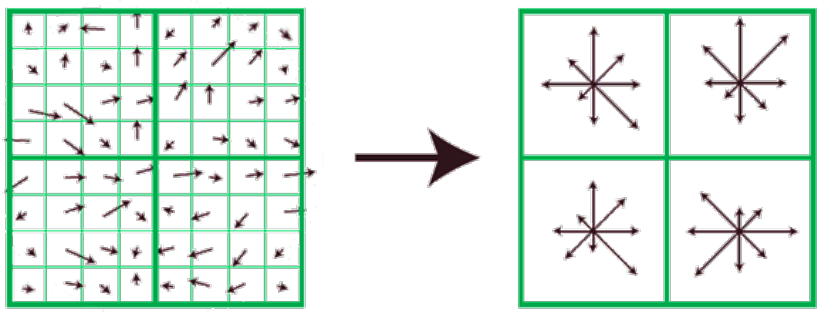

Figure 2.4: Illustration of the 2D gradient orientation histogram bins [L]

Normalizing across blocks The last stage computes normalization, which takes a block (local groups of cells) and normalizes their overall responses. Normalization introduces better robustness to illumination, shadowing, and edge contrast. A normalization factor is then calculated over the block and then applied over all histograms within the block. Typically each cell is shared between several blocks, but its normalization are block-dependent and different. Thus, each cell appears several times in the final output vector with different normalization, which revealed performance improvement. Finally, a single feature vector is generated with all normalized histograms [49, 46, 48]. See Figure 2.5.

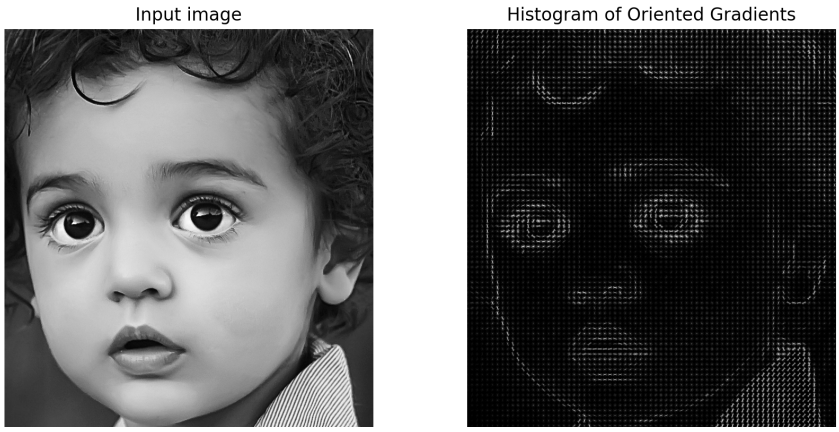

Figure 2.5: Visualization of Histogram of oriented gradient features

2.2.2 Canny edge detection

Edge detection refers to the process of recognizing and determining sharp discontinuities in an image. The discontinuities are sudden variations in pixel intensity, which distinguish the boundaries of objects in a scene [51]. Edge detection is one of the primary operations in computer vision with numerous approaches to it. The edge detection process serves to simplify the analysis of images by drastically decreasing the amount of information to be processed while maintaining useful structural data about object boundaries [52]. The Canny edge detector is one of the most precisely defined operators and widely used. The three main criteria of proper detection, good localization, and single response to an edge contribute to its popularity [53]. A standard implementation of the Canny edge detector algorithm follows the subsequent five steps.

Image Filtering: The first step of the traditional Canny algorithm is to smooth the image by applying Gaussian filter. This step will slightly blur the image to reduce the effects of visible noise on the edge detector. See figure 2.6. Such filters are essential

since all edge detection results are easily affected by the noise in the image leading to false edges [51, 54, 55].

Figure 2.6: Illustration of image smoothing [L]

Image Gradient Calculation: The second step is to find the intensity gradients of the image. An image edge may point in various directions, so the traditional Canny algorithm uses the neighboring area to get the gradient magnitude and direction in the blurred image [54, 55]. See figure 2.7.

Figure 2.7: Visualization of image gradient [L]

Non-maximum Suppression (NMS): After generating the gradient magnitude image, the non-maximum suppression algorithm is applied. The pixel with maximal value in the gradient direction will remain an edge pixel, and the rest will be eliminated. See figure 2.8. The gradient maximum value appears typically in the

center of an edge, with the increase of the distance in the gradient direction, the gradient value will decrease [51, 54, 55].

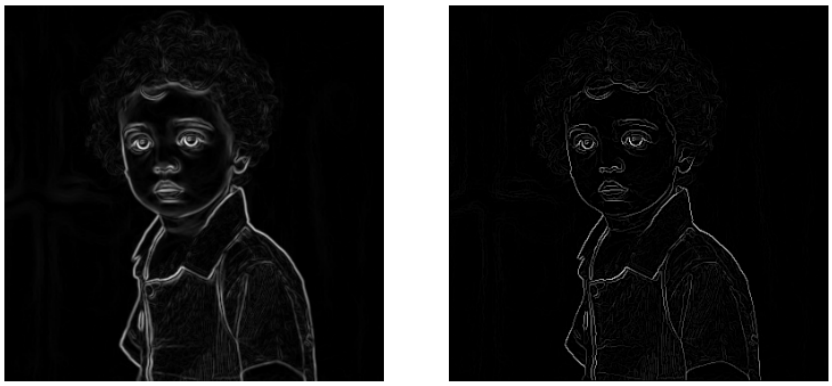

Figure 2.8: Non-maximum suppression effect on the gradient image [L]

Double threshold: The Canny algorithm adopts high and low threshold values to select edge points after applying non-maximum suppression. See figure 2.9. Strong edge points are the pixels with gradient magnitude above the high-threshold, and weak edge points are the pixels with gradient magnitude below the high-threshold and above the low-threshold. For all edge points below the low-threshold, they will be suppressed [51, 54, 55]. This step can reduce the impact of noise on the edge selection of the final edge image.

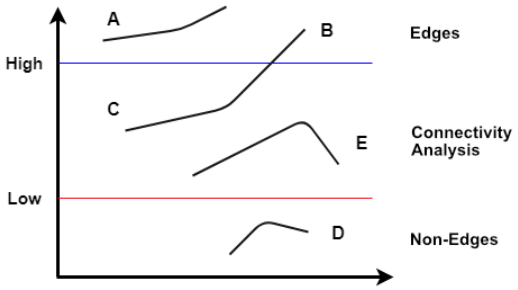

Figure 2.9: Hysteresis thresholding [L]

Edge tracking by hysteresis: So far, the strong pixels should undoubtedly be involved in the final edge, as they are a part of the image's actual edges. However, weak pixels connected to strong ones will be kept while noise and color variation

responses are unconnected to strong pixels, and therefore, they will be excluded [51, 54, 55]. See figure 2.10.

Figure 2.10: Visualization of Canny edge detection features [L]

2.2.3 Local binary patterns

Local Binary Pattern (LBP) [56, 57, 58] is a visual descriptor used for classification tasks in computer vision. The operator is a simple yet powerful means of texture description. LBP labels the pixels of an image block by thresholding each pixel [3x3]-neighborhood with the center value and indicating the result as a binary number (binary pattern) [59]. See figure 2.11.

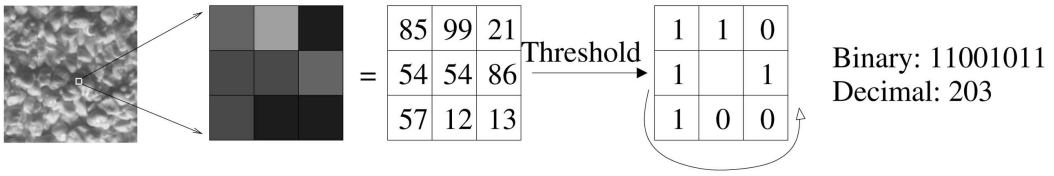

Figure 2.11: The basic LBP operator [60]

LBP features have performed very well in diverse applications due to its discriminating ability and computational simplicity, including texture classification and segmentation, image retrieval, and surface inspection [61]. One of LBP's essential properties in real-world applications is its robustness to monotonic gray-scale changes caused by illumination variations [62]. See figure 2.12.

Figure 2.12: Texture images obtained by LBP under different lighting conditions [62]

The original LPB operator has been extended with two main extensions to further improve its capabilities.

The bilinear interpolation of the pixel values extended the original operator allowing different sizes of circular neighborhoods (P, R) [57], where P is the number of sampling points on the circle with any radius R. See figure 2.13.

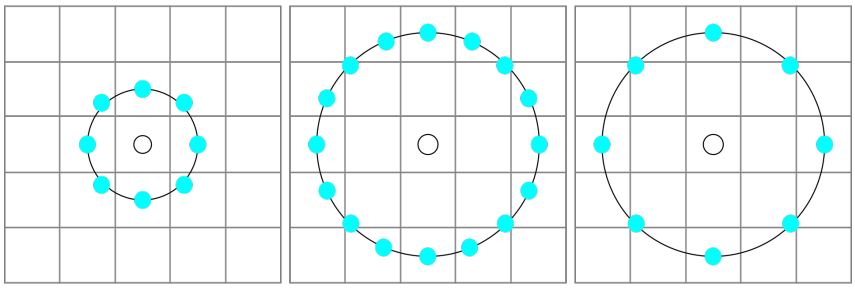

Figure 2.13: The circular (8,1), (16,2) and (8,2) neighborhoods. The pixel values are interpolated whenever the sampling point is not in the center of a pixel.

Uniform patterns are another extension to further improve LBP by only allowing at most two bit wise transitions in a binary pattern ($0 \rightarrow 1$ or $1 \rightarrow 0$), therefore significantly reducing the number of binary patterns [57, 62].

2.2.4 Dense DAISY

DAISY descriptor is a local image descriptor similar to the SIFT descriptor based on gradient orientation histograms. It is designed to allow for fast dense extraction [63, 64].

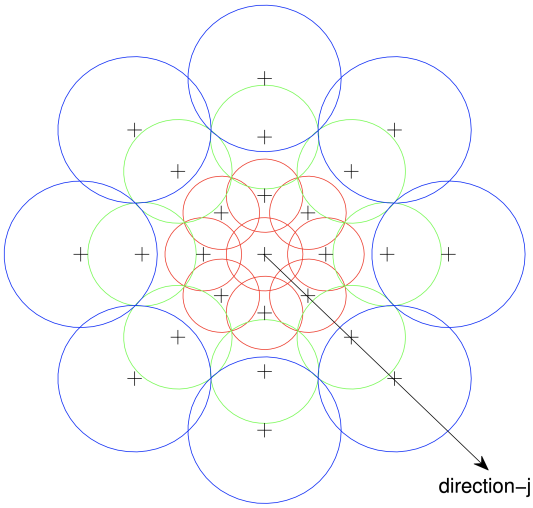

Figure 2.14: DAISY descriptor construction [63]

DAISY is constructed by several central-symmetrical circles, as shown in Figure 2.14. In general, around the center '+' sign, a concentric structure of three layers with different radius is constructed. Each layer contains eight sampling points, denoted with the '+' sign and distributed on equal intervals distribution. Each circle represents a region where the radius is proportional to the standard deviations of the Gaussian kernels, and the '+' sign represents the vector location where the descriptor is computed [65].

Since the sampling points per layer have the same Gauss value scale, the Gauss scale value linearly increases from the center to the outside. This structure makes the DAISY descriptor has better robustness for image affine and illumination variation. Besides, unlike the SIFT and SURF algorithm using a rectangular neighborhood, the DAISY descriptor uses the circular neighborhood since the circular neighborhood has a better positioning feature [65].

The next step is to normalize these vectors to the unit norm and denote the normalized vectors. The normalization is performed in each histogram independently to represent the pixels near occlusions as correct as possible [63, 64].

DAISY descriptor's primary motivation is to reduce the computational requirements by using Gaussian filters, which are separable (i.e., orientation maps can be computed for different sizes at low cost as convolutions with a large Gaussian kernel can be obtained from several successive convolutions with smaller kernels) [63, 64].

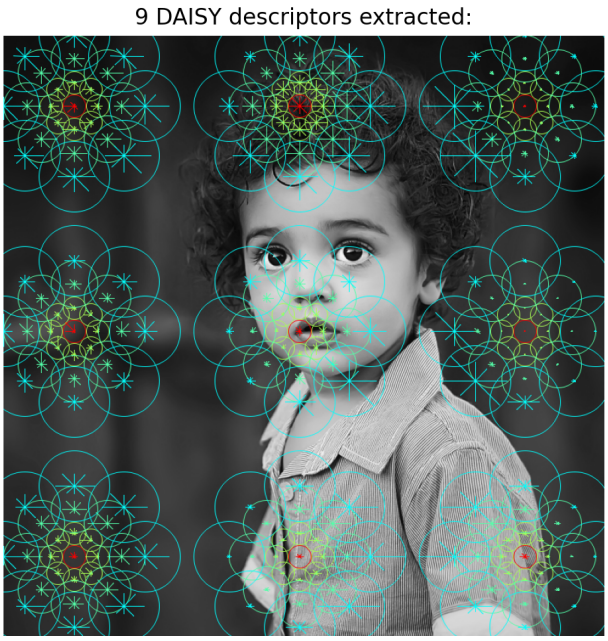

Figure 2.15: DAISY descriptor construction

2.3 Pattern recognition

Pattern recognition is the process of identifying patterns by applying a machine learning algorithm. This capability is not unique and often a prerequisite for intelligent behavior. Pattern recognition can be defined as data classification based on knowledge already obtained or on statistical information extracted from patterns or their representation [66]. A pattern can be as basic as a set of measurements or observations, represented in a d-dimensional vector. The key in many pattern recognition applications is identifying suitable attributes (e.g., features), form a good measure of similarity, and an associated matching process [67]. The main pattern recognition steps are shown in Figure 2.16.

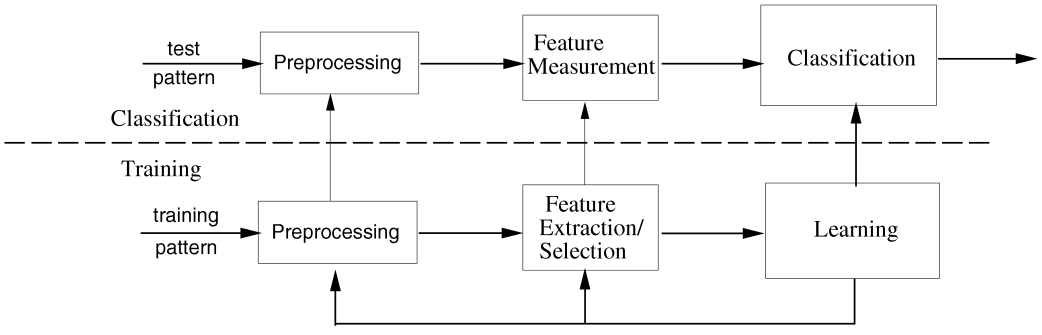

Figure 2.16: Model for statistical pattern recognition [67]

2.3.1 linear support vector machine (LSVM)

Support Vector Machine classifier is a supervised learning algorithm based on statistical learning theory, aiming to determine an optimal separating hyper plane (OSH) that separates two classes by using a training data set. The OSH separates the positive and negative training samples with the maximum margin (the distance from the chosen hyper plane to each category's nearest data point) [68, 69, 70]. SVM aims to maximize this margin to improve its generalization ability. Vectors from each class that are nearest to the discriminating surface are called support vectors. See figure 2.17. Since the support vectors include the information needed to define the classifier, the remaining samples are not needed after the support vectors are selected [71]. If no hard-margin hyper plane can separate the samples of two classes because of the noisy samples, the hard-margin LSVM will fail to find a feasible solution as it is super sensitive to outliers in the training data. So the soft margin extension is proposed to handle this problem. As the soft margin method can weaken the effect of outliers on SVM, it is widely used in classification problems. The soft margin method introduces the hinge loss function to determine the trade-off between increasing the margin size and ensuring that the data points lie on the margin's correct side [72].

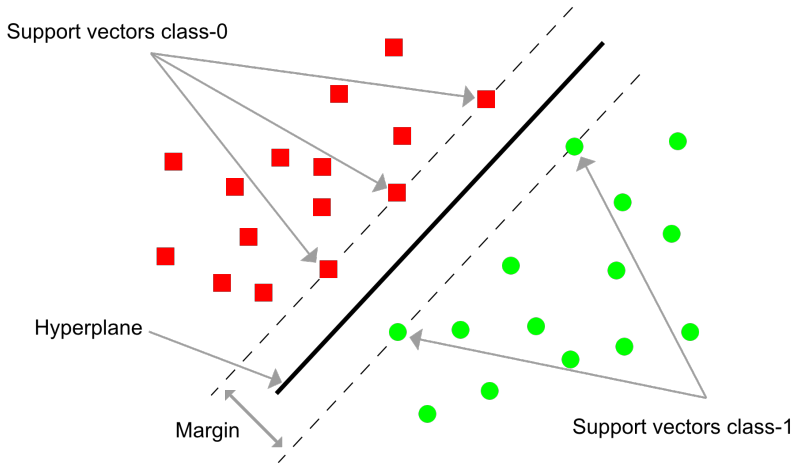

Figure 2.17: General classification hyper plane representation of SVM algorithm [L]

2.3.2 Probability calibration

Constructing a classifier to produce a posterior probability P(class | input) is very beneficial in practical classification problems. However, not all classifiers produce well-calibrated probabilities, and for some classifiers, the predicted probability does not match the output of its decision function [73]. Probability estimates are essential when the classifier uses the 'divide-and-combine' approach for decision-making. For example, in handwritten character recognition, the classifier's outputs are used as input to a high-level system that incorporates domain information, such as a language

model [74]. Nevertheless, Support Vector Machines produce an uncalibrated value that is not a probability.

John Platt introduced a method in 1999 for transforming predictions to posterior probabilities by passing them through sigmoid. This probability calibration module adds support for SVM's probability prediction.

$$P(y = 1|f) = \frac{1}{1+\exp(Af+B)}$$

This method is a logistic transformation of the classifier scores $f(x)$, where the parameters A and B are fitted by maximum likelihood estimation from fitting the training data set (f_i, y_i) [75].

2.4 Post-processing

2.4.1 Sliding Window

The sliding window approach is one of the primary strategies for object localization with bounding boxes. It has been the strategy for many years [49, 76]. This approach passes a small window with a fixed width and height over all portions of the image, generating multiple patches of the input image. See figure 2.18. For each of these patches, the classifier determines if it is necessary.

Figure 2.18: Visualization of the sliding window generated patches

2.4.2 Non maximum suppression

Non-maximum suppression (NMS) is a crucial post-processing step in various computer vision applications. In object detection, NMS is a necessity due to the sliding windows approach or detection algorithms' imperfect ability to localize regions of interest, resulting in collections of detections close to the correct locations. This relatively dense output is generally not satisfying for understanding the content of an image. Therefore, the goal of NMS is to retain only one prediction per group, corresponding to the precise local maximum of the response function. See figure 2.19.

The NMS algorithm molds a smooth response map that triggers several imprecise object window hypotheses, ideally obtaining only one bounding-box response per object. The most popular NMS approach for object detection is greedy NMS [77].

Greedy NMS Since the actual goal is to generate precisely one detection per object (or accurately one high confidence detection), The procedure starts by selecting the best scoring window and assuming that highly overlapping detections belong to the same object and suppress them. Greedy NMS repeats this procedure with the remaining detections, greedily accepting local maxima and discarding their neighbors. This method involves defining a measure of similarity between windows and setting a threshold for suppression. Eventually, this algorithm also accepts false detections, which is no problem as long as their confidence is lower than correct detections' confidence [78].

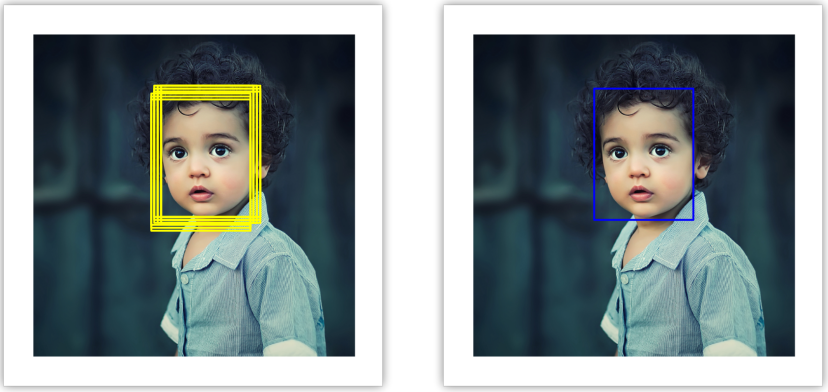

Figure 2.19: Visualization of the NMS algorithm.

2.4.3 Jaccard index

The Jaccard index, also known as mean Intersection over Union, is a classical similarity measure on sets with many practical applications in information retrieval, data mining, and machine learning. Jaccard index measures the relative size of the overlap of two finite sets A and B by dividing the size of their intersection by their union size [79]. See figure 2.20.

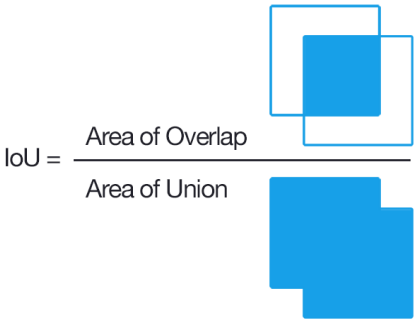

Figure 2.20: Jaccard index [L]

2.5 Image processing on the Web

The web is the most ubiquitous computing platform with billions of connected devices. Web technology is continually evolving in rich functionality as well as in scope. It rolled away from the relatively static and document-oriented beginnings to highly interactive and complex applications of today [80]. It is regularly converging with other technologies to provide more advanced functionality. One of those technologies is JavaScript. JavaScript (JS) is the primary scripting language for the web, and it is essential to modern Web applications.

JS has also rapidly grown from a programming language designed to add scripting capabilities into one of the most popular programming languages deployed on billions of devices. The JS environment has the potential to support a new and distinctive class of applications since it has the advantage of efficiency, completeness, API maturity, and community's collective knowledge [81].

However, image processing on the web is a complex and demanding task that requires sophisticated algorithms and implementations as it usually has a high computational cost, due to

- Real-time requirements for interactive applications
- A demanding amount of computation, mainly on images with higher resolution and high frame rates
- Complex algorithms to process and understand the visual data

The general paradigm of web-based applications is deploying computationally complex operations on the server. However, with recent client-side technologies, such as in time compilation, web clients can handle more demanding tasks. Still, there are requirements for a computer vision library on the web that are not entirely addressed yet [22].

3 Experiments

The web browser is probably the most used software application. It has evolved significantly over the past fifteen years. The primary functionality of a web browser is to bring information resources to the user, allowing them to view and access the information.

Web standards also help to standardize how a website can interact with assistive technologies, allowing different technologies to contribute to each other. Web standers deliver accessible sites to more people and more types of Internet devices, ensuring that most web users can have the best benefits.

Today, web browsers run on diverse hardware types, from smartphones and tablet PCs to desktop computers. Modern web browsers' capabilities allowed the development of highly interactive websites that can dynamically update information on a web page without reloading. JavaScript is a lightweight, interpreted programming language that allowed the creation of network-centric applications. JavaScript is also straightforward to implement since it is integrated with HTML as well as open and cross-platform. Advances in CSS allow browsers to display responsive website layouts and a wide array of visual effects. Cookies allow browsers to save session information and settings for specific websites.

Web applications are generally defined as cross-platform since they are accessible from various web browsers within different operating systems. The client-server system is the general architecture for such applications with a wide variety of complexity and functionality. Basic web applications achieve all or most processing from a stateless server with all user interaction consists of direct exchanges of data requests and server responses.

Meanwhile, the needs for software performance also increase the advancements and innovations of hardware technologies. This development allowed many tasks, such as real-time applications that were not possible in the past, to be possible today.

Real-time software is standard today as they provide faster accomplishment of tasks, operations, and activities on the computer. Real-time software is also subject to time constraints so that receiving data, processing them, and returning the responses are often understood to be in milliseconds.

3.1 Framework overview

For implementation, we used python as it offers a wide range of frameworks as well as it can run in the backend. Python also provides a concise and readable code.

We used OpenCV in python during our first attempt as it is one of the most popular computer vision frameworks. Later on, we had to change it due to the lack of good documentation and debugging system. We then used sklearn [82] and skimage [83] frameworks as they provide excellent documentation for all the algorithms we used as well as an ideal debugging system.

3.2 Feature extraction

For feature selection, we experimented with a wide range of feature extraction methods such as SIFT, SURF, ORB, and BRISK. These keypoints-oriented descriptors required a clustering algorithm for visual words. Three primary conditions limited our use of keypoints-oriented descriptors in our experiment

- Low accuracy in our evaluation
- Extra run-time for clustering
- Availability of binary clustering algorithm

We then decided on four different feature extraction methods used in various computer vision tasks to offer a wide range of options with reasonable execution time.

HOG: Object Recognition Histogram of oriented gradients is one of the most popular features extraction algorithms with many use case in recognition tasks [49, 84, 85, 86, 87, 88]. HOG descriptor is also used in multiple real-time applications such as Robust Facial Expression Recognition [89], Object Detection [90], and Detection and Recognition of Road Traffic Signs [91]. This popularity comes from its decent performance in recognition tasks with reasonable computational complexity.

Canny: Edge Detection The Canny edge detection algorithm is a commonly used edge detection method due to its low error rate and reliable detection. Its parameters allow it to be customized to identify the edges with various characteristics. There are many use cases for Canny algorithm such as medical application [92, 93], Road lane detection [94], Remote sensing [95], and Real-time facial expression recognition [96].

LBP: Texture Classification The local binary pattern is a discriminating method mainly used in texture classification of regions instead of individual pixels with

robustness to illumination variation and computational simplicity. LBP has many use cases such as Texture classification [97, 98], background modeling [59], Face recognition [60, 99], and Human detection [100].

DAISY: Image Matching The DAISY descriptor is an image matching descriptor inspired by SIFT but can be densely computed much faster. DAISY has multiple use cases such as Image Matching [65], object recognition [101], duplication forgery detection [102], and face recognition [103].

Dimension reduction is one of the primary keys to reduce each descriptors processing time. For all of our descriptors, we used gray scale images, reducing channel count to one channel. We also resized all the input images to multiple dimensions, attempting to find the balance between the highest accuracy with the lowest run-time.

3.3 Face Detection

To evaluation our descriptors, we chose the face detection task as it is one of the most known and studied computer vision topics with data sets availability. Since this problem has drawn more researchers' attention, multiple techniques have been developed to deal with it, such as classical feature-based techniques (e.g., cascade classifier). More recently, deep learning methods (e.g., Faster R-CNN and MTCNN) have achieved state-of-the-art performance on standard face detection benchmark data sets.

3.4 Classifier

For classification, we chose the Support Vector Machine algorithm as it is one of the most popular classification algorithms that directly minimize misclassification errors. We also used the SVM's linear variant as we are trying to predict either true or false class assignments.

Since SVM does not have Probabilistic Outputs for its prediction, we calibrated the classifier using Platt's method to produce a confidence score for every prediction.

3.5 Web

In the web browser, we employed server-side computation as the availability of front-end computer vision frameworks is not entirely addressed yet. Therefore, we utilized Python for back-end development since it is popular among back-end

technologies. There are many python frameworks for back-end development with many optimizations, such as Django and Flask. Still, we preferred the native python CGI script since it is part of Python's core library. We also build an HTTP server that invokes the CGI script to process the input submitted data through HTML elements.

4 Evaluation

This chapter describes our evaluation characteristics as well as the input and output form we followed to evaluate all models.

The primary evaluation characteristics for the features we discussed in the last chapter are

- Features capability to describe landmarks (visual words) in the training data set.
- The computational complexity required to calculate each feature.

By finding the appropriate trade-off between those two main characteristics, we can achieve real-time performance with high accuracy.

The input and output stages consist of several fixed steps applied for all models, reducing the evaluation's bias between all models.

4.1 Dataset

This section gives an overview of the method we followed to generate the training data set for our classification algorithm. We used the Face Detection Data Set and Benchmark (FDDB), which contains 2845 images with 5171 faces. The specification of face regions is elliptical regions and represented as a 6-tuple (major radius, minor radius, angle, center-x, center-y). The FDDB consists of many difficulties including occlusions, challenging poses, and low resolution and out-of-focus faces [104]. We randomly selected 250 of single-face images as our test data set and used the rest as the training data set. We generated two main categories from the training data set to train our classifier: valid and Invalid samples.

Valid samples: First, the valid samples are generated by extracting every face in the input image by bounding ellipses with windows and labeling them as a valid sample. See figure 4.1

Figure 4.1: Visualization of elliptic region bounding-box [104]

Next, to increase the count of valid samples, we cropped [0, 2, 4, 8] pixels from every edge of the extracted window if the window's width is greater than 18 pixels. See figure 4.2. This step created at maximum extra three valid samples from each window to be used in the training process.

Lastly, all windows are then resized to fixed size so that all the valid samples have the same dimensions. The total count of valid samples is 5159.

Figure 4.2: Example of valid samples [104]

Invalid samples: To generate invalid samples, we used the sliding window approach over the input image with the size and step equals the valid sample dimensions. The generated invalid samples are then accepted if they do not overlap with any valid samples window. See figure 4.3. The total count of invalid samples is 39718.

Figure 4.3: Every cell represents an invalid training sample [104]

4.2 Groundtruth

The evaluation of the image processing models we build relies on the ground-truth data used to train them. Practically deployed systems depend heavily on being trained and tested on ground-truth data from images and videos obtained from actual deployments. For evaluation, we allocated a subset of 250 random images obtained from the FDDB data set as our evaluation data set and bounded every elliptic region with a rectangle (window) to mark the face position. This evaluation data set contains only single-face images to avoid multiple issues in our evaluation process. The first issue is faces with different scales relevant to the prediction window, as shown in figure 4.4.

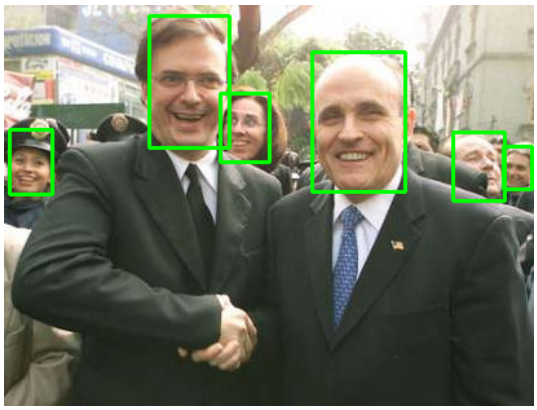

Figure 4.4: Example of an image with different face-sizes [104]

The second issue is the intersection between the faces' window. This issue leads to one of the primary deficiencies with Gready-NMS algorithm, as it will suppress windows with high intersection value. See figure 4.5.

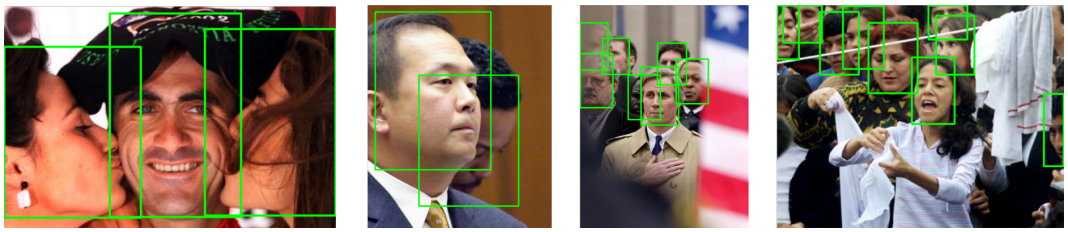

Figure 4.5: Example of intersecting faces [104]

With single-face images in the proper scale to fit inside the prediction window, we significantly reduced the computational power as well as time complexity required to evaluate different feature in different configurations as we do not have to apply multiple scales on the evaluation data set. See figure 4.6. The evaluation data set produces 3,311,069 patches to be predicted for each feature configuration.

Figure 4.6: Single face images in proper scale [104]

4.3 Evaluation metrics

Our evaluation process starts with generating patches from the input images using the sliding window approach. This technique allows us to localize precisely the faces in a photo by passing a window with a specific size over every image in the evaluation data set generating many patches from the input image. We then pass all the generated patches to our trained binary classifier to assign a class for every patch. The binary classifier predicted a 'TRUE' or 'FALSE' class for every patch with a confidence score between 0 and 1. Yet, the classifier also correctly classifies multiple patches of the same face in the image due to the sliding window approach introducing redundant predictions. One method of overcoming this problem is the non-maximum suppression algorithm. Next, we provide all the 'TRUE'-class predictions with their confidence score to the NMS algorithm to suppress all the overlapping predictions. The NMS algorithm discarded all patches with a confidence score below 0,3 and suppressed all windows with a Jaccard index higher than 0,5 (mIoU = 0,5). Finally, we selected the window with the highest confidence score from the remaining predictions as our final face location.

4.3.1 Jaccard index

Our first evaluation metric is the Jaccard index. We can estimate every descriptors prediction accuracy by calculating the average intersection-over-union between the ground truth window and the predicted one for the evaluation data set.

4.3.2 Run time

Our second evaluation metric is run time in the browser. We calculated the average time required to extract features from multiple ground truth images.

4.4 Results

For Evaluation, we used the HOG, Canny, LBP, and Daisy extraction methods implemented by [83]. We also used some of the parameters proposed in skimage documentation and denoted them above every descriptor table.

Descriptor tables contain the following data:

- Training data set dimensions
- Descriptor specific configurations
- The time required to extract features from the training data set
- The number of features extracted form every data point
- Average mIoU value for the validation data set
- The time needed to extract features in the browser

HOG:

```
orientation_bins=9, pixels_per_cell=(8, 8), block_norm='L1'
```

Histogram of oriented gradients					
Dimensions	Cells per block	Time	Features	Accuracy	Browser
(44877, 47, 67)	[2, 2]	46.79 s	1008	87.15 %	3.27 ms
	[3, 3]	36.19 s	1458	87.36 %	2.87 ms
	[4, 4]	26.43 s	1440	87.32 %	2.44 ms
(44877, 38, 46)	[2, 2]	22.22 s	432	79.06 %	1.88 ms
	[3, 3]	15.56 s	486	80.11 %	1.61 ms
	[4, 4]	10.50 s	288	78.40 %	1.33 ms
(44877, 24, 29)	[2, 2]	9.90 s	144	73.84 %	1.12 ms
	[3, 3]	6.04 s	81	75.95 %	0.99 ms

Canny:

```
low_threshold=0.1, high_threshold=0.2, use_quantiles=False
```

Canny					
Dimensions	sigma	Time	Features	Accuracy	Browser
(44877, 47, 67)	1.0	56.23 s	3149	63.63 %	2.92 ms
(44877, 38, 46)		41.67 s	1748	58.02 %	2.08 ms
(44877, 24, 29)		34.46 s	696	60.14 %	1.72 ms
(44877, 19, 27)	1.0	30.06 s	513	65.66 %	1.71 ms
	0.5	34.21 s	513	66.64 %	1.66 ms
	1.5	33.08 s	513	62.40 %	1.71 ms

LBP:

```
method='uniform'
```

Local Binary Patterns					
Dimensions	(P, R)	Time	Features	Accuracy	Browser
(44877, 47, 67)	(8, 1)	16.04 s	3149	64.06 %	0.74 ms
	(8, 2)	15.35 s		66.44 %	0.72 ms
	(8, 3)	15.06 s		68.50 %	0.71 ms
	(16, 2)	26.20 s		64.71 %	1.25 ms
(44877, 19, 27)	(8, 1)	4.50 s	513	62.51 %	0.35 ms
	(8, 2)	4.45 s		67.08 %	0.34 ms
	(8, 3)	4.40 s		67.95 %	0.34 ms
	(16, 2)	6.10 s		66.37 %	0.52 ms
(P, R) = (points, radius)					

DAISY:

```
normalization='l1'
```

DAISY					
Dimensions	(s, r, n, h, b)	Time	Features	Accuracy	Browser
(44877, 47, 67)	(4, 15, 3, 8, 8)	296 s	10000	87.76 %	16.40 ms
	(4, 15, 2, 8, 4)	128 s	3400	85.50 %	7.13 ms
	(6, 15, 2, 8, 4)	128 s	1428	84.33 %	6.91 ms
	(4, 15, 1, 4, 4)	114 s	1000	82.59 %	5.85 ms
	(6, 15, 1, 4, 4)	110 s	420	80.88 %	5.70 ms
(44877, 38, 46)	(4, 11, 2, 8, 4)	88 s	1632	78.15 %	3.73 ms
	(4, 11, 1, 4, 4)	70 s	480	70.77 %	3.15 ms
(s, r, n, h, b) = (step, radius, rings, histograms, orientations)					

5 Discussion

Feature extraction improves learned models' accuracy by extracting features from the input data. Execution of real-time feature extraction tasks must satisfy timing constraints, enforcing each task instance to complete its execution before deadline. By finding the right balance between the three main key elements

- Input data dimensions
- Descriptor configuration
- Number of extracted features

that determine the computational complexity of each descriptor, we can achieve real-time performance.

We tried multiple configurations as well as dimensions for the input data for all descriptors. HOG and DAISY achieved good accuracy scores in a reasonable run time. The best results were obtained from HOG features are expected since it is one of the most popular features extraction algorithms with many applications.

Surprisingly by reducing the input data dimensions, LBP and Canny performed similar or better, achieving the best run time between all descriptors.

Overall, all the descriptors performed well on simple images with the highest confidence score indicating face-window, as shown in figure 5.1, 5.2.

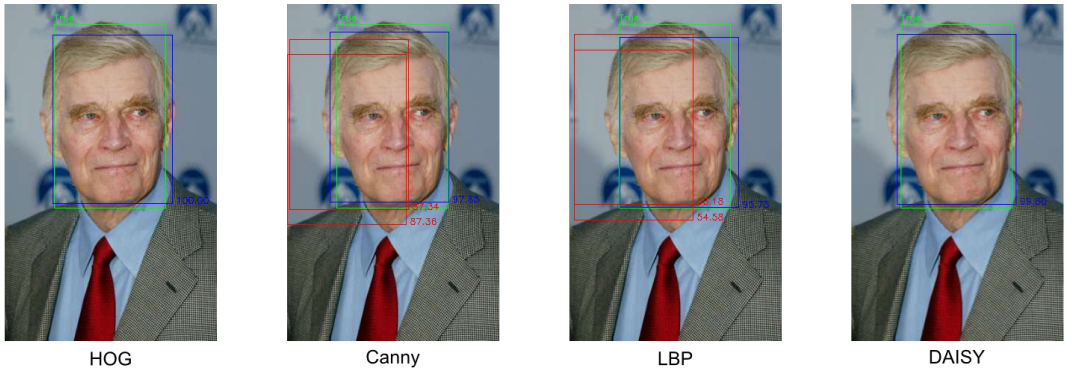

Figure 5.1: The green rectangle is the ground truth, the blue rectangle is the predicted face with the highest confidence score, and the red rectangles are ignored predictions. The confidence score is noted beside each prediction [104].

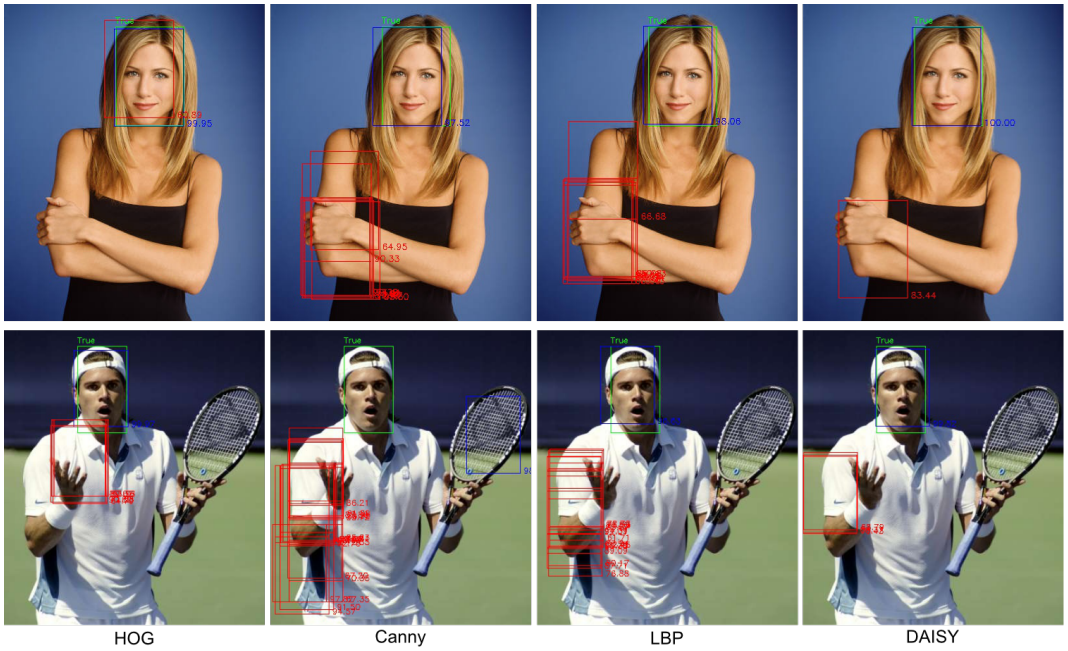

Figure 5.2: Prediction examples. [104]

Experiment limitations:

This experiment has some limitations which should be noted.

First, our evaluation data set is relatively small and only contains single face images as we are trying to evaluate the descriptors with appropriate configurations instead of building the most accurate face detection model.

Second, not all the descriptors we chose are suitable for face detection task, and the low accuracy score does not represent each descriptors full potential. Selecting the most appropriate and robust descriptor is heavily task dependant.

Third, our models often cause false detections, even if they do not have the highest confidence score. Such false detections that only include hand or arm are probably caused by valid samples shown in figure 5.3 since we manually checked the training data set for false classified samples.

Figure 5.3: Example of valid samples containing hands. [104]

6 Conclusion

We have shown that using one of the four selected descriptors gives a good run time, achieving the overall computational complexity required to extract features in real-time. We explored the influence of various descriptor parameters and input data dimensions on the performance as well as the computation time of each descriptor. Finally, with these descriptors available and an appropriate number of features, a conventional classifier can compete with the state-of-the-art methods, processing the input image close to real-time.

Further improvements:

Python is a scripting language that is interpreted at runtime instead of being compiled. Despite each descriptor's short run time, this python property makes it relatively slower than other programming languages such as C/C++.

Descriptors should also be available natively in the javascript environment (front-end) to better use modern and capable devices. The only descriptor we could find implemented in JS is the canny algorithm implemented by OpenCV.js with very similar computational complexity to the OpenCV-python variant.

Bibliography

[1] Aniket Kittur, Ed H Chi, and Bongwon Suh. Crowdsourcing user studies with mechanical turk. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 453–456, 2008.

[2] Janet Ilieva, Steve Baron, and Nigel M Healey. Online surveys in marketing research. *International Journal of Market Research*, 44(3):1–14, 2002.

[3] W. Fuhl. *Image-based extraction of eye features for robust eye tracking*. PhD thesis, University of Tübingen, 04 2019.

[4] Wolfgang Fuhl. From perception to action using supervised learning based on observations. *User Modeling and User-Adapted Interaction*, pages 1–18, 08 2020.

[5] W. Fuhl, T. Santini, C. Reichert, D. Claus, A. Herkommer, H. Bahmani, K. Rifai, S. Wahl, and E. Kasneci. Non-intrusive practitioner pupil detection for unmodified microscope oculars. *Elsevier Computers in Biology and Medicine*, 79:36–44, 12 2016.

[6] W. Fuhl, T. C. Kübler, D. Hospach, O. Bringmann, W. Rosenstiel, and E. Kasneci. Ways of improving the precision of eye tracking data: Controlling the influence of dirt and dust on pupil detection. *Journal of Eye Movement Research*, 10(3), 05 2017.

[7] W. Fuhl, H. Gao, and E. Kasneci. Neural networks for optical vector and eye ball parameter estimation. In *ACM Symposium on Eye Tracking Research & Applications, ETRA 2020*. ACM, 01 2020.

[8] W. Fuhl, D. Geisler, T. Santini, T. Appel, W. Rosenstiel, and E. Kasneci. Cbf:circular binary features for robust and real-time pupil center detection. In *ACM Symposium on Eye Tracking Research & Applications*, 06 2018.

[9] W. Fuhl, H. Gao, and E. Kasneci. Tiny convolution, decision tree, and binary neuronal networks for robust and real time pupil outline estimation. In *ACM Symposium on Eye Tracking Research & Applications, ETRA 2020*. ACM, 01 2020.

[10] D. Geisler, W. Fuhl, T. Santini, and E. Kasneci. Saliency sandbox: Bottom-up saliency framework. In *12th Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2017)*, 02 2017.

[11] W. Fuhl, T. Kübler, T. Santini, and E. Kasneci. Automatic generation of saliency-

based areas of interest. In *Symposium on Vision, Modeling and Visualization (VMV)*, 09 2018.

[12] Wolfgang Fuhl, Efe Bozkir, Benedikt Hosp, Nora Castner, David Geisler, Thiago C., and Enkelejda Kasneci. Encodji: Encoding gaze data into emoji space for an amusing scanpath classification approach ;). In *Eye Tracking Research and Applications*, 2019.

[13] W. Fuhl, N. Castner, T. C. Kübler, A. Lotz, W. Rosenstiel, and E. Kasneci. Ferns for area of interest free scanpath classification. In *Proceedings of the 2019 ACM Symposium on Eye Tracking Research & Applications (ETRA)*, 06 2019.

[14] Wolfgang Fuhl, Yao Rong, and Kasneci Enkelejda. Fully convolutional neural networks for raw eye tracking data segmentation, generation, and reconstruction. In *Proceedings of the International Conference on Pattern Recognition*, pages 0–0, 2020.

[15] W. Fuhl, T. Santini, T. Kuebler, N. Castner, W. Rosenstiel, and E. Kasneci. Eye movement simulation and detector creation to reduce laborious parameter adjustments. *arXiv preprint arXiv:1804.00970*, 2018.

[16] W. Fuhl, N. Castner, and E. Kasneci. Rule based learning for eye movement type detection. In *International Conference on Multimodal Interaction Workshops, ICMIW*, 2018.

[17] W. Fuhl and E. Kasneci. Eye movement velocity and gaze data generator for evaluation, robustness testing and assess of eye tracking software and visualization tools. In *Poster at Egocentric Perception, Interaction and Computing, EPIC*, 2018.

[18] W. Fuhl, T. Santini, D. Geisler, T. C. Kübler, W. Rosenstiel, and E. Kasneci. Eyes wide open? eyelid location and eye aperture estimation for pervasive eye tracking in real-world scenarios. In *ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct publication – PETMEI 2016*, 09 2016.

[19] W. Fuhl, T. Santini, D. Geisler, T. C. Kübler, and E. Kasneci. Eyelad: Remote eye tracking image labeling tool. In *12th Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2017)*, 02 2017.

[20] W. Fuhl, T. Santini, and E. Kasneci. Fast and robust eyelid outline and aperture detection in real-world scenarios. In *IEEE Winter Conference on Applications of Computer Vision (WACV 2017)*, 03 2017.

[21] Wolfgang Fuhl, Efe Bozkir, and Enkelejda Kasneci. Reinforcement learning for the privacy preservation and manipulation of eye tracking data. *arXiv preprint arXiv:2002.06806*, 08 2020.

[22] Sajjad Taheri, Alexander Veidenbaum, Alexandru Nicolau, and Mohammad R Haghighat. Opencv. js: Computer vision processing for the web. *Univ. California, Irvine, Irvine, CA, USA, Tech. Rep*, 2017.

[23] W. Fuhl, G. Kasneci, W. Rosenstiel, and E. Kasneci. Training decision trees as replacement for convolution layers. In *Conference on Artificial Intelligence, AAAI*, 02 2020.

[24] W. Fuhl, D. Geisler, W. Rosenstiel, and E. Kasneci. The applicability of cycle gans for pupil and eyelid segmentation, data generation and image refinement. In *International Conference on Computer Vision Workshops, ICCVW*, 11 2019.

[25] W. Fuhl, W. Rosenstiel, and E. Kasneci. 500,000 images closer to eyelid and pupil segmentation. In *Computer Analysis of Images and Patterns, CAIP*, 11 2019.

[26] W. Fuhl, N. Castner, L. Zhuang, M. Holzer, W. Rosenstiel, and E. Kasneci. Mam: Transfer learning for fully automatic video annotation and specialized detector creation. In *International Conference on Computer Vision Workshops, ICCVW*, 2018.

[27] W. Fuhl, T. Santini, and E. Kasneci. Fast camera focus estimation for gaze-based focus control. In *CoRR*, 2017.

[28] W. Fuhl, S. Eivazi, B. Hosp, A. Eivazi, W. Rosenstiel, and E. Kasneci. Bore: Boosted-oriented edge optimization for robust, real time remote pupil center detection. In *Eye Tracking Research and Applications, ETRA*, 2018.

[29] Wolfgang Fuhl and Enkelejda Kasneci. Multi layer neural networks as replacement for pooling operations. *arXiv preprint arXiv:2006.06969*, 08 2020.

[30] Wolfgang Fuhl and Enkelejda Kasneci. Weight and gradient centralization in deep neural networks. *arXiv*, 08 2020.

[31] Wolfgang Fuhl and Enkelejda Kasneci. Rotated ring, radial and depth wise separable radial convolutions. *arXiv*, 08 2020.

[32] W. Fuhl and E. Kasneci. Learning to validate the quality of detected landmarks. In *International Conference on Machine Vision, ICMV*, 11 2019.

[33] Wolfgang Fuhl, Yao Rong, Thomas Motz, Michael Scheidt, Andreas Hartel, Andreas Koch, and Enkelejda Kasneci. Explainable online validation of machine learning models for practical applications. *arXiv*, 08 2020.

[34] John C Russ. Image processing. In *Computer-assisted microscopy*, pages 33–69. Springer, 1990.

[35] W. Fuhl, T. C. Kübler, H. Brinkmann, R. Rosenberg, W. Rosenstiel, and E. Kasneci. Region of interest generation algorithms for eye tracking data. In *Third Workshop on Eye Tracking and Visualization (ETVIS), in conjunction with ACM ETRA*, 06 2018.

[36] W. Fuhl, T. C. Kübler, K. Sippel, W. Rosenstiel, and E. Kasneci. Arbitrarily shaped areas of interest based on gaze density gradient. In *European Conference on Eye Movements, ECEM 2015*, 08 2015.

[37] Rafael C Gonzalez. Richard e. woods digital image processing, pearson, 2018.

[38] Wai Kai Chen. *The electrical engineering handbook*. Elsevier, 2004.

[39] John A Lee and Michel Verleysen. *Nonlinear dimensionality reduction*. Springer Science & Business Media, 2007.

[40] Imola K Fodor. A survey of dimension reduction techniques. Technical report, Lawrence Livermore National Lab., CA (US), 2002.

[41] Laurens Van Der Maaten, Eric Postma, and Jaap Van den Herik. Dimensionality reduction: a comparative. *J Mach Learn Res*, 10(66-71):13, 2009.

[42] Ngo Quang Minh Khiem, Guntur Ravindra, Axel Carlier, and Wei Tsang Ooi. Supporting zoomable video streams with dynamic region-of-interest cropping. In *Proceedings of the first annual ACM SIGMM conference on Multimedia systems*, pages 259–270, 2010.

[43] Christopher Kanan and Garrison W Cottrell. Color-to-grayscale: does the method matter in image recognition? *PloS one*, 7(1):e29740, 2012.

[44] Gaurav Kumar and Pradeep Kumar Bhatia. A detailed review of feature extraction in image processing systems. In *2014 Fourth international conference on advanced computing & communication technologies*, pages 5–12. IEEE, 2014.

[45] Richard P Lippmann. Pattern classification using neural networks. *IEEE communications magazine*, 27(11):47–50, 1989.

[46] Gerard Lacey and David Fernandez Llorca. Hand washing monitoring system, January 3 2012. US Patent 8,090,155.

[47] W. Fuhl, N. Castner, and E. Kasneci. Histogram of oriented velocities for eye movement detection. In *International Conference on Multimodal Interaction Workshops, ICMIW*, 2018.

[48] Frédéric Suard, Alain Rakotomamonjy, Abdelaziz Bensrhair, and Alberto Broggi. Pedestrian detection using infrared images and histograms of oriented gradients. In *2006 IEEE Intelligent Vehicles Symposium*, pages 206–212. IEEE, 2006.

[49] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 886–893. IEEE, 2005.

[50] David Jacobs. Image gradients. *Class Notes for CMSC*, 426, 2005.

[51] Raman Maini and Himanshu Aggarwal. Study and comparison of various image edge detection techniques. *International journal of image processing (IJIP)*, 3(1):1–11, 2009.

[52] John Canny. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, PAMI-8(6):679–698, 1986.

[53] Lijun Ding and Ardeshir Goshtasby. On the canny edge detector. *Pattern Recognition*, 34(3):721–725, 2001.

[54] Cai-Xia Deng, Gui-Bin Wang, and Xin-Rui Yang. Image edge detection algorithm based on improved canny operator. In *2013 International Conference on Wavelet Analysis and Pattern Recognition*, pages 168–172. IEEE, 2013.

[55] Weibin Rong, Zhanjing Li, Wei Zhang, and Lining Sun. An improved canny edge detection algorithm. In *2014 IEEE International Conference on Mechatronics and Automation*, pages 577–582. IEEE, 2014.

[56] Timo Ojala, Matti Pietikäinen, and David Harwood. A comparative study of texture measures with classification based on featured distributions. *Pattern recognition*, 29(1):51–59, 1996.

[57] Timo Ojala, Matti Pietikainen, and Topi Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on pattern analysis and machine intelligence*, 24(7):971–987, 2002.

[58] Matti Pietikäinen, Timo Ojala, and Zelin Xu. Rotation-invariant texture classification using feature distributions. *Pattern recognition*, 33(1):43–52, 2000.

[59] Marko Heikkila and Matti Pietikainen. A texture-based method for modeling the background and detecting moving objects. *IEEE transactions on pattern analysis and machine intelligence*, 28(4):657–662, 2006.

[60] Abdenour Hadid, Matti Pietikainen, and T Ahonen. Face description with local binary patterns: Application to face recognition. *IEEE transactions on pattern analysis and machine intelligence*, 28(12):2037–2041, 2006.

[61] Abdenour Hadid, Matti Pietikainen, and Timo Ahonen. A discriminative feature space for detecting and recognizing faces. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 2, pages II–II. IEEE, 2004.

[62] Timo Ahonen, Abdenour Hadid, and Matti Pietikainen. Face description with local binary patterns: Application to face recognition. *IEEE transactions on pattern analysis and machine intelligence*, 28(12):2037–2041, 2006.

[63] Engin Tola, Vincent Lepetit, and Pascal Fua. A fast local descriptor for dense matching. In *2008 IEEE conference on computer vision and pattern recognition*, pages 1–8. IEEE, 2008.

[64] Engin Tola, Vincent Lepetit, and Pascal Fua. Daisy: An efficient dense descriptor applied to wide-baseline stereo. *IEEE transactions on pattern analysis and machine intelligence*, 32(5):815–830, 2009.

[65] Li Li and John Zic. Image matching algorithm based on feature-point and daisy descriptor. *journal of multimedia*, 9(6):829–834, 2014.

[66] Robert J Schalkoff. Pattern recognition. *Wiley Encyclopedia of Computer Science and Engineering*, 2007.

[67] Anil K Jain, Robert P. W. Duin, and Jianchang Mao. Statistical pattern recognition: A review. *IEEE Transactions on pattern analysis and machine intelligence*, 22(1):4–37, 2000.

[68] Saeed Meshgini, Ali Aghagolzadeh, and Hadi Seyedarabi. Face recognition using gabor-based direct linear discriminant analysis and support vector machine. *Computers & Electrical Engineering*, 39(3):727–745, 2013.

[69] Ignas Kukenys and Brendan McCane. Support vector machines for human face detection. In *Proceedings of the New Zealand computer science research student conference*, pages 226–229. Citeseer, 2008.

[70] Jin Wei, Zhang Jian-Qi, and Zhang Xiang. Face recognition method based on support vector machine and particle swarm optimization. *Expert Systems with Applications*, 38(4):4390–4393, 2011.

[71] Yaguo Lei. *Intelligent fault diagnosis and remaining useful life prediction of rotating machinery*. Butterworth-Heinemann, 2016.

[72] William S Noble. What is a support vector machine? *Nature biotechnology*, 24(12):1565–1567, 2006.

[73] John Platt et al. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74, 1999.

[74] Bianca Zadrozny and Charles Elkan. Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 694–699, 2002.

[75] Alexandru Niculescu-Mizil and Rich Caruana. Predicting good probabilities with supervised learning. In *Proceedings of the 22nd international conference on Machine learning*, pages 625–632, 2005.

[76] Henry A Rowley, Shumeet Baluja, and Takeo Kanade. Human face detection in visual scenes. In *Advances in Neural Information Processing Systems*, pages 875–881, 1996.

[77] Rasmus Rothe, Matthieu Guillaumin, and Luc Van Gool. Non-maximum suppression for object detection by passing messages between windows. In *Asian conference on computer vision*, pages 290–306. Springer, 2014.

[78] Jan Hosang, Rodrigo Benenson, and Bernt Schiele. Learning non-maximum suppression. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4507–4515, 2017.

[79] Sven Kosub. A note on the triangle inequality for the jaccard distance. *Pattern Recognition Letters*, 120:36–38, 2019.

[80] Shadi Abou-Zahra, Judy Brewer, and Michael Cooper. Artificial intelligence (ai) for web accessibility: Is conformance evaluation a way forward? In *Proceedings of the Internet of Accessible Things*, pages 1–4. 2018.

[81] Daniel Smilkov, Nikhil Thorat, Yannick Assogba, Ann Yuan, Nick Kreeger, Ping Yu, Kangyi Zhang, Shanqing Cai, Eric Nielsen, David Soergel, et al. Tensorflow. js: Machine learning for the web and beyond. *arXiv preprint arXiv:1901.05350*, 2019.

[82] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[83] Stefan Van der Walt, Johannes L Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D Warner, Neil Yager, Emmanuelle Gouillart, and Tony Yu. scikit-image: image processing in python. *PeerJ*, 2:e453, 2014.

[84] Harihara Santosh Dadi and GK Mohan Pillutla. Improved face recognition rate using hog features and svm classifier. *IOSR Journal of Electronics and Communication Engineering*, 11(4):34–44, 2016.

[85] Mohamed Dahmane and Jean Meunier. Emotion recognition using dynamic grid-based hog features. In *Face and Gesture 2011*, pages 884–888. IEEE, 2011.

[86] Reza Ebrahimzadeh and Mahdi Jampour. Efficient handwritten digit recognition based on histogram of oriented gradients and svm. *International Journal of Computer Applications*, 104(9), 2014.

[87] Takuya Kobayashi, Akinori Hidaka, and Takio Kurita. Selection of histograms of oriented gradients features for pedestrian detection. In *International conference on neural information processing*, pages 598–607. Springer, 2007.

[88] Hilton Bristow and Simon Lucey. Why do linear svms trained on hog features perform so well? *arXiv preprint arXiv:1406.2419*, 2014.

[89] Pranav Kumar, SL Happy, and Aurobinda Routray. A real-time robust facial expression recognition system using hog features. In *2016 International Confer-*

ence on Computing, Analytics and Security Trends (CAST), pages 289–293. IEEE, 2016.

[90] Kosuke Mizuno, Yosuke Terachi, Kenta Takagi, Shintaro Izumi, Hiroshi Kawaguchi, and Masahiko Yoshimoto. Architectural study of hog feature extraction processor for real-time object detection. In *2012 IEEE Workshop on Signal Processing Systems*, pages 197–202. IEEE, 2012.

[91] Jack Greenhalgh and Majid Mirmehdi. Real-time detection and recognition of road traffic signs. *IEEE transactions on intelligent transportation systems*, 13(4):1498–1506, 2012.

[92] Hairong Qi, Wesley E Snyder, Jonathan F Head, and Robert L Elliott. Detecting breast cancer from infrared images by asymmetry analysis. In *Proceedings of the 22nd Annual International Conference of the IEEE Engineering in Medicine and Biology Society (Cat. No. 00CH37143)*, volume 2, pages 1227–1228. IEEE, 2000.

[93] Naveen Singh, Dilip Gandhi, and Krishna Pal Singh. Iris recognition system using a canny edge detection and a circular hough transform. *International Journal of Advances in Engineering & Technology*, 1(2):221, 2011.

[94] Prema M Daigavane and Preeti R Bajaj. Road lane detection with improved canny edges using ant colony optimization. In *2010 3rd International Conference on Emerging Trends in Engineering and Technology*, pages 76–80. IEEE, 2010.

[95] Mohamed Ali and David Clausi. Using the canny edge detector for feature extraction and enhancement of remote sensing images. In *IGARSS 2001. Scanning the Present and Resolving the Future. Proceedings. IEEE 2001 International Geoscience and Remote Sensing Symposium (Cat. No. 01CH37217)*, volume 5, pages 2298–2300. IEEE, 2001.

[96] Peng Zhao-Yi, Zhu Yan-Hui, and Zhou Yu. Real-time facial expression recognition based on adaptive canny operator edge detection. In *2010 Second International Conference on MultiMedia and Information Technology*, volume 2, pages 154–157. IEEE, 2010.

[97] Yang Zhao, Wei Jia, Rong-Xiang Hu, and Hai Min. Completed robust local binary pattern for texture classification. *Neurocomputing*, 106:68–76, 2013.

[98] Timo Ojala, Matti Pietikäinen, and Topi Mäenpää. A generalized local binary pattern operator for multiresolution gray scale and rotation invariant texture classification. In *International Conference on Advances in Pattern Recognition*, pages 399–408. Springer, 2001.

[99] Timo Ahonen, Abdenour Hadid, and Matti Pietikäinen. Face recognition with local binary patterns. In *European conference on computer vision*, pages 469–481. Springer, 2004.

[100] Yadong Mu, Shuicheng Yan, Yi Liu, Thomas Huang, and Bingfeng Zhou.

Discriminative local binary patterns for human detection in personal album. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008.

[101] Chao Zhu, Charles-Edmond Bichot, and Liming Chen. Visual object recognition using daisy descriptor. In *2011 IEEE International Conference on Multimedia and Expo*, pages 1–6. IEEE, 2011.

[102] Jing-Ming Guo, Yun-Fu Liu, and Zong-Jhe Wu. Duplication forgery detection using improved daisy descriptor. *Expert Systems with Applications*, 40(2):707–714, 2013.

[103] Carmelo Velardo and Jean-Luc Dugelay. Face recognition with daisy descriptors. In *Proceedings of the 12th ACM workshop on Multimedia and security*, pages 95–100, 2010.

[104] Vidit Jain and Erik Learned-Miller. Fddb: A benchmark for face detection in unconstrained settings. Technical Report UM-CS-2010-009, University of Massachusetts, Amherst, 2010.

List of Figures