Instituto Tecnológico y de Estudios Superiores de Occidente

Reconocimiento de validez oficial de estudios de nivel superior según acuerdo secretarial 15018, publicado en el Diario Oficial de la Federación del 29 de noviembre de 1976.

Department of Mathematics and Physics
Master of Data Science

Image Features' Extraction Using Proportional-Integral Filters

THESIS to obtain the **DEGREE** of
MASTER OF DATA SCIENCE

A thesis presented by:
Gabriel Alejandro Morales Ruiz

Thesis Advisors:
D.Sc. Juan Diego Sánchez Torres
M.Sc. José Eduardo Carvajal Rubio

Tlaquepaque, Jalisco, July, 2022

Instituto Tecnológico y de Estudios Superiores de Occidente

Reconocimiento de validez oficial de estudios de nivel superior según acuerdo secretarial 15018, publicado en el Diario Oficial de la Federación del 29 de noviembre de 1976.

Department of Mathematics and Physics

Master of Data Science Approval Form

Thesis Title: **Image Features' Extraction Using Proportional-Integral Filters**

Author: **Gabriel Alejandro Morales Ruiz**

Thesis Approved to complete all degree requirements for the Master of Science Degree in Data Science.

Thesis Advisor, **D.Sc. Juan Diego Sánchez Torres**

Thesis Co-Advisor, **M.Sc. José Eduardo Carvajal Rubio**

Thesis Reader, **D.Sc. Riemann Ruiz Cruz**

Thesis Reader, **D.Sc. Esteban Jiménez Rodríguez**

Academic Advisor, **Juan Carlos Martínez Alvarado**

Tlaquepaque, Jalisco, July, 2022

Image Features’ Extraction Using Proportional-Integral Filters

Gabriel Alejandro Morales Ruiz

Abstract

This document aims to demonstrate that features generated as a side effect of using Proportional-Integral filters, with explicit or implicit Euler discretization, could be viable for imaging Machine Learning applications.

This thesis delves into how the Proportional-Integral and Super-Twisting filters are defined, how they are discretized with the Forward and Backward Euler method, which results in the need to calculate an error or difference scaled with a step size (discrete derivative), and how these features are used in an image classification problem with two different datasets, using a convex machine learning method (Support Vector Classifier). These results are compared with a commonly used kernel for extracting an image’s differences or edges, called the Sobel operator.

With the PI and ST robustness and stability in mind and their implicit low-pass filter from their formulation as a continuous output signal, two tests involving adding artificial uniform and Gaussian noise are realized. The features derived from the automatic control algorithms prove to be viable but more valuable in situations where noise exists.

Contents

List of Figures

List of Tables

Dedicated to ....

Miguel, who encouraged me to pursue this degree and supported me all through it.

1 Introduction

Contents

1.1 Justification

Machine learning algorithms in image processing applications often use an image's edges as features and add a procedure to calculate them in their pipeline, where edges can be interpreted as the silhouette of all different objects and contrasts within the image. The task of differentiating objects within an image is called image segmentation, where many methods from various disciplines exist and are used interchangeably depending on the application.

Humans identify different objects according to their color and texture; we naturally and automatically perform segmentation of what we observe within our brains. For example, a gigantic dog will still be perceived as a dog because size doesn't matter as much as its fur, geometry, and color. We can also identify its volume through the observed shading. [1]

[1] George Stockman Linda Shapiro. *Computer Vision*. Prentice Hall, 2001. ISBN 978-0130307965

Texture and geometry give information about the spatial arrangement of colors in an image, and edge detectors are typically employed as the first step in texture analysis. A hardship arises because changes may occur over a wide range of scales, so no single filter can be an all-end solution for every problem. [2]

[2] E. Hildreth D. Marr. *Theory of edge detection*. Proc. R. Soc. Lond., 1980. DOI: 10.1098/rspb.1980.0020

1.2 Problem Statement

Applications such as image enhancement, recognition, and classification employ machine learning algorithms. These algorithms will train their parameters accordingly to transform inputs into the desired outputs (within an error margin), wherein these inputs are called features.

Commonly used features for these image machine learning problems include the image's edges information [3]; also dubbed the image's gradient, because of the congruence between an image's edges and a discrete derivative in both horizontal and vertical directions.

[3] Scott E. Umbaugh. *Computer imaging: digital image analysis and processing*. CRC Press, 2005. ISBN 0-8493-2919-1

As mentioned before, as one may need to focus differently on varying changes in the image's texture or geometry, many different solutions exist. Would it be possible to use automatic control algorithms where a reference's derivative is implicitly computed while trying to replicate it? Discrete automatic control algorithms require step size information to calculate errors and try to follow the original signal. How could a machine learning algorithm benefit from additional hyperparameters introduced by these automatic control algorithms, such as step size? Could some problems be additionally tuned and have their performance increase? Would the physical impediments of how the control algorithms' formulation (stability and continuity) help smooth over the image if noise is added?

1.3 *Objectives*

This section will discuss general and specific targets that drive this document.

1.3.1 *General Objectives*

- Extract features from an image (edges) using estimation algorithms based on automatic control theory.
- Compare these generated features to the commonly used baseline Sobel.

1.3.2 *Specific Objectives*

- Extract edges using a discrete Proportional-Integral filter, with both explicit and implicit discretization.
- Extract edges using a Super-Twisting filter, with both explicit and implicit discretization.
- Test and compare the extracted features using a convex optimization method, such as a support vector classifier.
- Add Gaussian and Uniform noise to the inputs before extracting the features and repeat the previous steps for different noise magnitudes.
- Complete all the previous steps with two different datasets.

- Perform hypothesis tests to verify if a statistically significant distinction between using the baseline and these features exists.

2 Literature Review

Contents

2.1 Digital Image Processing Concepts

Even though image editing is, by itself, image processing, the main difference is that image processing will refer to working directly with the discrete pixels array that describes the discretized image. Image editing mainly refers to working with software that gives non-technical people access to modify an image's parameters, such as brightness, contrast, and gamma, among other variables. [1]

[1] Mark Burge Wilhelm Burger. *Principles of Digital Image Processing: Fundamental Techniques*. Springer-Verlag London, 2009. ISBN 978-1-84800-190-9

2.1.1 Digital Image

An image is a two-dimensional (horizontal and vertical) analog signal. Older analog cameras sensed the light that bounced off what the photographed subject was and wrote it in film. Nowadays, digital cameras discretize the sensed image and save the results as a tensor, with values for each pixel on every point of the horizontal-vertical plane, plus a third dimension for color.

Modern digital images are a collection of 2d arrays, where each array corresponds to the intensity of each particular color. A basic model for this is the RGB model, where the image has a size of $x \times y \times 3$, in which x is the number of pixels in each row, and y is the number of pixels in each column; the 3 means that there are 3 $x \times y$ arrays: one for the detected intensity for colors red, green, and blue; a linear combination of these yields any possible color from the color spectrum.

Detectors calibrated to specific wavelengths sense these colors, which generate a voltage and convert it to a discrete value through an analog-digital converter (ADC). In the case of a 256-level ADC, 255 means high saturation detection, and 0 means no such color saturates that pixel. [2]

[2] Jorge Lira Chávez. *Tratamiento digital de imágenes multiespectrales*. UNAM, 2010. ISBN 978-607-00-3403-9

2.1.2 Transformations and Spatial Filtering

As with signal processing, signals can be worked on in the spatial or the frequency domain. The spatial domain is what you would call the "regular image space", or the image as it is. The frequency domain represents how quickly the value of the pixels in the spatial domain changes in the image. E.g. An image with a drop from white (255, 255, 255) to (0, 0, 0) from bottom to top (Fig 2.1) can be modeled as a linear combination of sinusoidal functions, of which the results can be seen in the frequency domain representation (Fig 2.2).

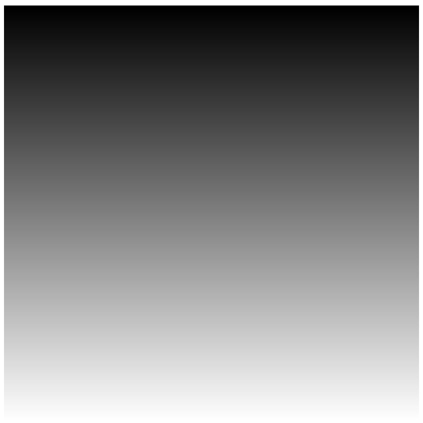


Figure 2.1: Black to white gradient.

As the color is only changing in the y axis, the frequency representation in the image will only have information (changes) in the y axis as well. If an image rotates 90 degrees, the frequency representation will rotate too. With that logic, when using figure 2.3, a picture whose pixel values change through all directions, the frequency representation of the image will have information about

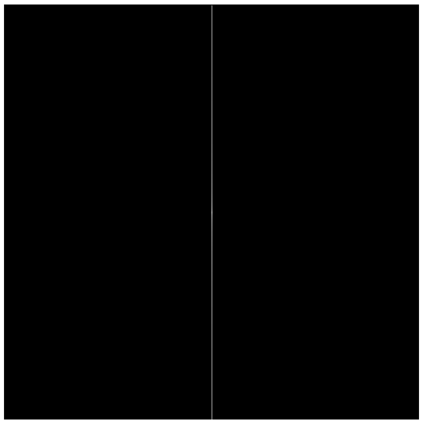


Figure 2.2: Fast Fourier Transform of black to white gradient.

every direction (Fig 2.4).

Figure 2.3: Lenna standard test image.

In this project, we will work solely on the spatial domain; the frequency domain could be something to explore in future work.

A transformation in the spatial domain can be defined as:

$$g(x,y) = T[f(x,y)] \tag{2.1}$$

where $f(x,y)$ is an input image, $g(x,y)$ is the output, and T is an operator that operates on a neighborhood of points (x,y) [3].

[3] Richard E. Woods Rafael C. Gonzalez. *Digital Image Processing*. Pearson Education Limited, 2018. ISBN 978-1-292-22304-9

Transformations, also called kernels, can sometimes be represented as a matrix. In this representation, they are applied to the input image with a convolution, with a convolution defined as:

$$[f * g](t) = \int_0^t f(\tau)g(t-\tau)d\tau \tag{2.2}$$

This yields the following equation:

$$g(x,y) = f(x,y) * K \tag{2.3}$$

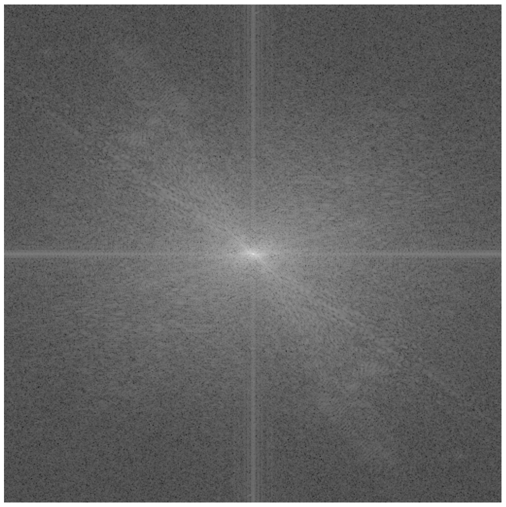


Figure 2.4: Fast Fourier Transform of Lenna standard test image.

Some standard kernels include low-pass filters (blur), high-pass filters (sharpen), and edge detection. This work will use Sobel edge detection as a baseline against which the extracted features' performance will be compared.

2.1.3 Sobel

The Sobel kernel is used in image processing to detect an image's borders. Presented in 1968 as "A 3x3 Isotropic Gradient Operator for Image Processing" [4]. Sobel is widely used in applications in image classification where the distinguishing factors are the geometric features (shape) or texture and not the color or background (e.g., face recognition).

[4] G. Feldman I. Sobel. A 3x3 isotropic gradient operator for image processing. pages 271–272, 1968

$$\text{Sobel}_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}; \quad \text{Sobel}_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{2.4}$$

These matrices are applied to the input image with the convolution operation. For a 2d array, a window with the kernel values will move across all the pixels in both x and y directions. If an image has more than one color (e.g., RGB), the convolution is applied to every color individually.

Let us imagine how the Sobel operators would work on an edge. Visualize an application to figure 2.1, specifically the y operator. (Fig 2.5)

The pixel value is continuously increasing while going downwards.

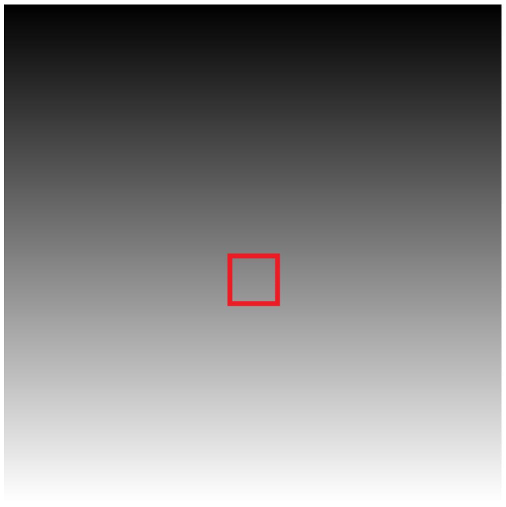


Figure 2.5: Representation of a Sobel kernel being applied to a black to white gradient.

If the intensity of the values is

$$\text{Window} = \begin{bmatrix} 127 & 127 & 127 \\ 128 & 128 & 128 \\ 129 & 129 & 129 \end{bmatrix}$$

then, when the kernel is applied to that window but before summing the values

$$\text{Window} = \begin{bmatrix} 127 & 254 & 127 \\ 0 & 0 & 0 \\ -129 & -258 & -129 \end{bmatrix}$$

The final result for the output image in the pixel that pertains to the center of the window is -8: this lets us knows that, with the Sobel operator, the image is changing intensity in that spot. As the gradient is smooth, the results will all be small numbers, as no true "edge" appears on the image; however, using the Lenna image (Fig 2.3) and applying both Sobel operators to it yields the result seen in figure 2.6.

The human eye can immediately detect edges. In image processing, edges are just spots where the image changes pixel values rapidly. One can see the commonality with edges in images and discrete derivatives, where the derivative and the edges are more prominent when the difference between pixels $p_{x,y}$ and $p_{x+1,y}$ or $p_{x,y+1}$ is greater. This commonality is the groundwork from which the problem statement of this thesis stands. Thus, this project's hypothesis is: "PI filters calculate a derivative of the signal they try to replicate employing the discretization process. Can this derivative be used as a feature for image classification problems?"

Now, if a noisy sensor retrieves an image and the Sobel operator is applied to it, it would not be able to differentiate which parts were originally from the image and which were noise. For example, let's add

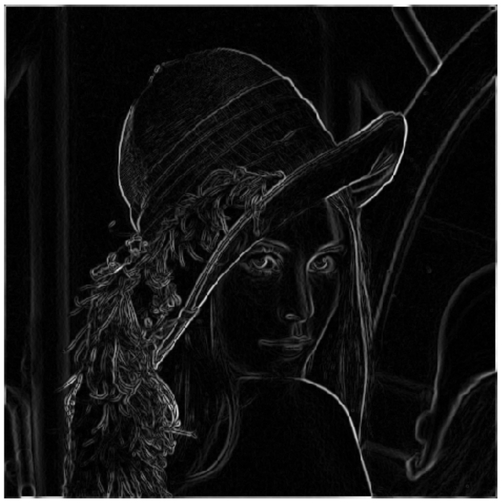


Figure 2.6: Edge detection of Lenna standard test image via Sobel kernels.

a $100\mathcal{N}(\mu = 0, \sigma = 1)$ noise to figure 2.3 and use the Sobel kernel on it. Results are shown in figures 2.7 & 2.8. Note that a machine learning algorithm might still be able to recognize Lenna, even if the human eye cannot identify her silhouette.

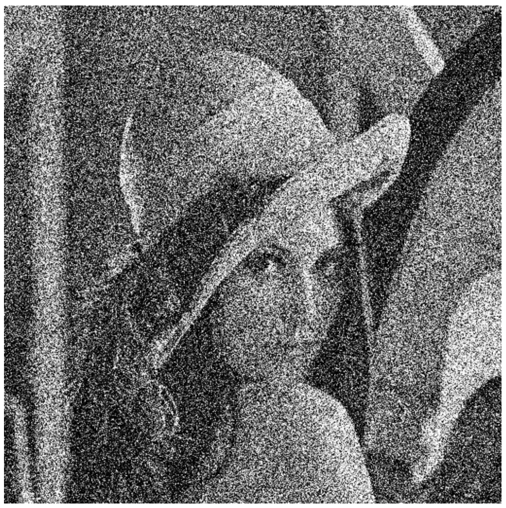


Figure 2.7: Lenna standard image with Gaussian noise.

2.2 *Automatic Control Concepts*

This section describes the automatic control notions and algorithms required to understand and follow this document.

2.2.1 *Control System*

Plainly speaking, a control system is "something" that regulates, guides, or directs variables or behaviors to achieve specific objectives. E.g.:

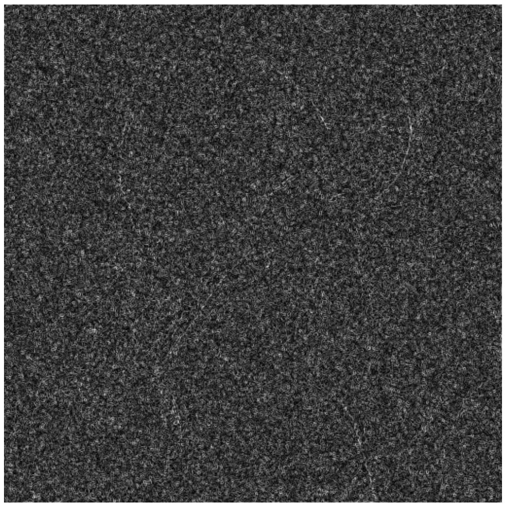


Figure 2.8: Sobel operator applied to Lenna image with Gaussian noise.

- An electric fan in a hot room has four states: off, low, medium, and high. You turn the fan's knob to the medium setting, and the fan starts spinning.
- A room has a climate system with radiators, air conditioning, and airflow control. This system is controlled by adjusting a thermostat. You turn the knob on the thermostat to a lower value, and the system adapts to achieve the selected temperature.

These two examples also demonstrate two different control systems: open-loop and closed-loop. The control system will not read the output to adjust its setting for open-loop systems, but it will for closed-loop systems. The fan will not start spinning faster by itself if it realizes that the room is still hot, but the thermostat-controlled room will increase the intensity of the AC if the room does not cool down and will lower the power once it reaches the desired temperature because there's a thermometer sensing what is happening and alerting the system. Automatic control systems range from mundane applications such as a thermostat to applications such that human control is impossible due to having to process a large amount of data in a limited time.[5]

[5] J.C. Maxwell. On governors. In *Proceedings of The Royal Society of London*, number 10, pages 270–283, 1868

A control system has three basic components[6]:

[6] Benjamin C. Kuo Farid Golnaraghi. *Automatic Control Systems*. John Wiley & Sons INC, 2010. ISBN 978-0470-04896-2

1. Control objectives
2. Control system components
3. Results or outputs

2.2.2 *PID Controller*

A closed-loop control mechanism first proposed by Nicolas Minorsky[7],

[7] S. Bennett. A brief history of automatic control. In *IEEE Control Systems Magazine Vol. 16*, number 3, pages 17–25, 1996. DOI: 10.1109/37.506394

wherein the output tries to replicate the input by adjusting itself proportionally to:

- the difference between the current output and the input, or the current error (P)
- the cumulative past errors (I). E.g., the longer it takes for the output to reach the input, the faster it will adjust to try and match it. This also helps diminish steady-state errors, as these will accumulate and force the controller to change the output.
- the difference between the previous error and the current one (D). E.g., if the input changes drastically, the derivative's magnitude will increase, and the controller will adjust its output accordingly to try and stabilize faster.[8]

[8] Tore Hagglund Karl J. Astrom. *PID Controllers*. Instrument Society of America, 1995. ISBN 1-55617-516-7

A PID control variable can be described with the following equation:

$$u(t) = k_p e(t) + k_i \int e(t)dt + k_d \frac{de(t)}{dt} \tag{2.5}$$

$$e(t) = y_{ref}(t) - y(t) \tag{2.6}$$

where y_{ref} is the desired output and $y(t)$ the current output.

Definition 1 *A Proportional-Integral (PI) filter is a structure based on the PI controller, and is defined as:*

$$\dot{x}_1(t) = f(t, x_1(t)) + u(t) \tag{2.7}$$

$$u(t) = \lambda_1 x_1 + v(t) \tag{2.8}$$

$$\dot{v}(t) = \lambda_2 x_1 \tag{2.9}$$

2.2.3 *Super Twisting*

The Super Twisting algorithm is a PI structure with non-linear terms, which implements a second-order sliding mode controller. Considering the Single-Input-Single-Output system described in equation (2.10), the super twisting sliding mode controller is given by equations 2.11 & 2.12[9]. Note from the equations that Super-Twisting is a variation of a Proportional-Integral structure; therefore, all these implementations will be referred to as Proportional-Integral filters. [10]

[9] A. Levant. Sliding order and sliding accuracy in sliding mode control. In *International Journal of Control Vol. 58*, pages 1247–1263, 1993

[10] Jorge Rivera et al. *Sliding Mode Control*. IntechOpen, 2011. ISBN 978-953-51-6002-1

$$\dot{x}_1(t) = f(t, x_1(t)) + u(t) \tag{2.10}$$

$$u(t) = k_p \sqrt{|e(t)|}\,\mathrm{sgn}(e(t)) + k_i \int \mathrm{sgn}(e(t))dt \tag{2.11}$$

$$e(t) = y_{ref}(t) - y(t) \tag{2.12}$$

Definition 2 *A Super-Twisting (ST) filter is a structure based on the ST controller, which is in itself based on the PI structure, and is defined as:*

$$u(t) = \lambda_1 \sqrt{|x_1(t)|} sgn(x_1(t)) + v(t) \tag{2.13}$$
$$\dot{v}(t) = \lambda_2 sgn(x_1(t)) \tag{2.14}$$

Homogeneous differentiators based on sliding modes have been formulated to estimate the first n derivatives of a signal, given that its $n+1$ derivative is bounded by a known constant L. Taking this into account, λ_1 and λ_2 can be obtained from combining a second order differentiator with a controller on a Single-Input-Single-Output system; the results are written in equations (2.15 & 2.16). [11]

[11] A. Levant. Higher-order sliding modes, differentiation and output-feedback control. In *International Journal of Control Vol. 76*, number 6, pages 924–941, 2003

$$\lambda_1 = 1.5\sqrt{L} \tag{2.15}$$
$$\lambda_2 = 1.1L \tag{2.16}$$

2.2.4 *Explicit Euler discretization*

Also known as Forward Euler Method, it is a simple numerical method that solves an initial value problem for an ordinary differential equation. Its goal is to derive a difference equation that approximates the original differential equation to eliminate derivatives, going from equation (2.18), which is a derivative's estimation, to equation (2.19) using the initial value problem (2.17).[12]

[12] Kendall E. Atkinson et al. *Numerical solution of ordinary differential equations*. John Wiley & Sons, Inc., 2009. ISBN 978-111-81-6449-5

$$Y'(t) = f(t, Y(t)), \qquad Y(0) = Y_0 \tag{2.17}$$

$$Y'(t) \simeq \frac{1}{h}[Y(t+h) - Y(t)] \tag{2.18}$$

$$Y(t+h) \simeq Y(t) + hf(t, Y(t)) \Rightarrow y_{n+1} = y_n + hf(t_n, y_n) \tag{2.19}$$

Definition 3 *The PI filter with explicit discretization is defined as:*

$$x_{1,k+1} = x_{1,k} + h\lambda_1 x_{1,k} + hv_k \tag{2.20}$$
$$v_{k+1} = v_k + h\lambda_2 x_{1,k} \tag{2.21}$$

Both the original signal and its first derivative are being pieced together. Applying this algorithm to all rows and columns in the Lenna image (Fig 2.3), and showing its v values results in figure 2.9 (hyperparameters $h = 0.01$, $\lambda_1 = 150$, $\lambda_2 = 3000$). Considering how the filter follows physical limitations (non-discontinuity), any noise that is added to the image (Fig 2.7) can be reduced by its design. The results for when the filter is used on the noisy Lenna image are shown in figure 2.10 (hyperparameters $h = 0.01$, $\lambda_1 = 20$, $\lambda_2 = 50$.

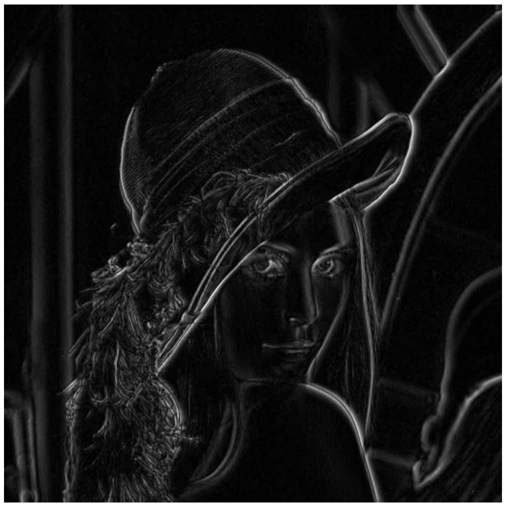


Figure 2.9: Edge detection of Lenna image with an explicit PI filter.

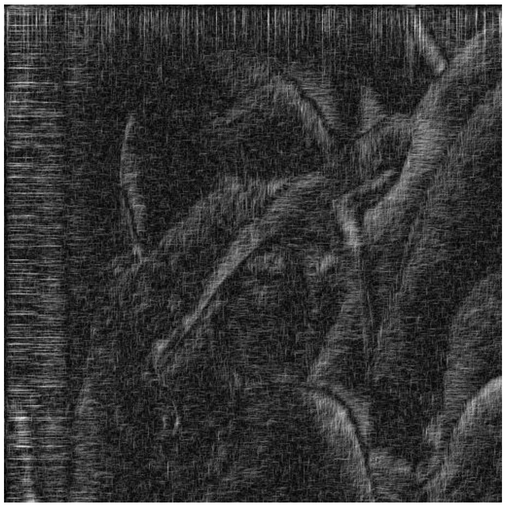


Figure 2.10: Edge detection of noisy Lenna image with an explicit PI filter.

Definition 4 *The ST filter with explicit discretization is defined as:*

$$x_{1,k+1} = x_k + h\lambda_1 sign(x_{1,k})\sqrt{|x_{1,k}|} + \frac{h\lambda_2 sign(x_{1,k})}{2} + hv_k \tag{2.22}$$

$$v_{k+1} = v_k + h\lambda_2 sign(x_{1,k}) \tag{2.23}$$

The results obtained from repeating the experiment done with the PI filter on the Lenna image are shown in figures 2.11 (hyperparameters $h = 0.1$, $L = 800$) and 2.12 (hyperparameters 0.01, $L = 9000$).

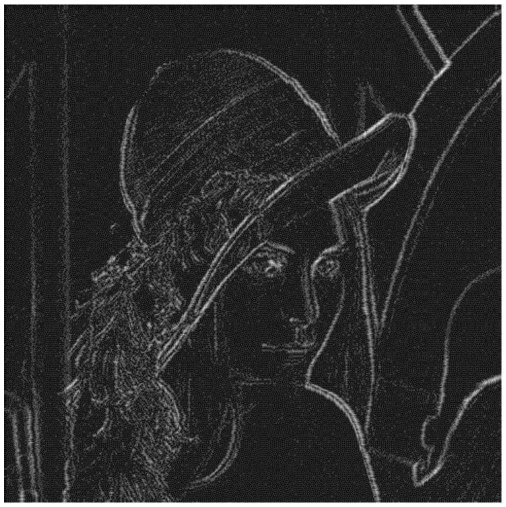


Figure 2.11: Edge detection of Lenna image with an explicit ST filter.

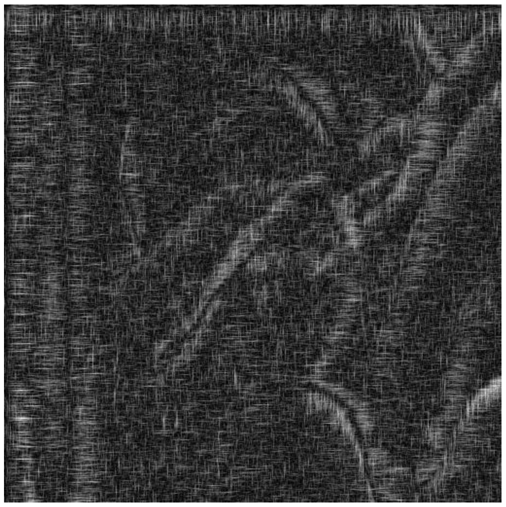


Figure 2.12: Edge detection of noisy Lenna image with an explicit ST filter.

2.2.5 *Implicit Euler discretization*

Also known as Backward Euler Method, as it takes from the previous value to get its derivative approximation (eq 2.24) and not the next value as the Forward Euler Method does. This yields the definition from equation (2.25), in which we can shift the indexes of the discrete signal by one and get equation (2.26). This method requires solving implicit equations and involves a higher computational cost, but it has better stability properties. [13]

[13] Kendall E. Atkinson et al. *Numerical solution of ordinary differential equations*. John Wiley & Sons, Inc., 2009. ISBN 978-111-81-6449-5

$$Y'(t) \simeq \frac{1}{h}[Y(t) - Y(t-h)] \tag{2.24}$$

$$y_n = y_{n-1} + hf(t_n, y_n) \tag{2.25}$$

$$y_{n+1} = y_n + hf(t_{n+1}, y_{n+1}) \tag{2.26}$$

Definition 5 *The implicit discretization for the PI filter is defined as:*

$$\widetilde{x}_{1,k+1} = x_{1,k} + h\lambda_1 \widetilde{x}_{1,k+1} + hv_{k+1} \tag{2.27}$$
$$v_{k+1} = v_k + h\lambda_2 \widetilde{x}_{1,k+1} \tag{2.28}$$

Algorithm 1 Implementation of implicit PI discretization for application

```
$x_k \leftarrow y_k - \widetilde{y}_k$   #Error
$\widetilde{x}_{1,k+1} \leftarrow \frac{x_{1,k} + hv_k}{1 - h\lambda_1 - h^2\lambda_2}$
$v_{k+1} \leftarrow v_k + h\lambda_2 \widetilde{x}_{1,k+1}$
$u_k \leftarrow \lambda_1 \widetilde{x}_{1,k+1} + hv_{k+1}$
$\widetilde{y}_{k+1} \leftarrow \widetilde{y}_k + hu_k + hv_{k+1}$
```

Definition 6 *The implicit discretization for the ST filter, taken from Brogliato's work* [14]*, is defined as:*

[14] Denis Efimov Bernard Brogliato, Andrey Polyakov. The implicit discretization of the super-twisting sliding-mode control algorithm. In *15th International Workshop on Variable Structure Systems*, pages 349–353, july 2018

$$\widetilde{x}_{1,k+1} = x_{1,k} - h\lambda_1 \sqrt{|\widetilde{x}_{1,k+1}|} sgn(\widetilde{x}_{1,k+1}) + hv_{k+1} \tag{2.29}$$
$$v_{k+1} \in v_k - \lambda_2 h sgn(\widetilde{x}_{1,k+1}) \tag{2.30}$$

Where $sgn(x)$

$$sgn(x) = \begin{cases} 1 & \text{if } x > 0 \\ -1 & \text{if } x < 0 \\ [-1,1] & \text{if } x = 0 \end{cases} \tag{2.31}$$

2.3 Machine Learning Concepts

This section describes the machine learning theory essential to understanding and following this document.

Algorithm 2 Implementation of implicit ST discretization for application

```
x_k ← $y_k - \tilde{y}_k$   #Error
$a \leftarrow h\lambda_1$
$b_k \leftarrow -x_k - hv_k$
if $b_k < -h^2\lambda_2$ then
    $\sqrt{|\tilde{x}_{1,k+1}|} \leftarrow \frac{1}{2}(-a + \sqrt{a^2 - 4(b_k + h^2\lambda_2)})$
    $v_{k+1} \leftarrow v_k - h\lambda_2$
    $u_k \leftarrow -\lambda_1\sqrt{|\tilde{x}_{1,k+1}|} + v_{k+1}$
else if $b_k \in [-h^2\lambda_2, h^2\lambda_2]$ then
    $\tilde{x}_{1,k+1} \leftarrow 0$
    $v_{k+1} = v_k + \frac{b_k}{h}$
    $u_k = v_{k+1}$
else if $b_k > h^2\lambda_2$ then
    $\sqrt{|\tilde{x}_{1,k+1}|} \leftarrow \frac{1}{2}(-a + \sqrt{a^2 + 4(b_k - h^2\lambda_2)})$
    $v_{k+1} \leftarrow v_k + h\lambda_2$
    $u_k \leftarrow \lambda_1\sqrt{|\tilde{x}_{1,k+1}|} + v_{k+1}$
end if
$\tilde{y}_{k+1} \leftarrow \tilde{y}_k - hu_k + hv_{k+1}$
```

2.3.1 *Ordinary Least Squares Regression*

The ordinary least squares regression problem consists of finding a linear combination of features X that best describe the desired output y. For $i = 1, \ldots, n$ the mean of the conditional distribution of y_i is given by a feature vector x_i, in the form of

$$y_i = x_i^T\theta + \epsilon_i. \tag{2.32}$$

θ and x_i are vectors with size $k \times 1$, and ϵ_i is a vector of independent and identically distributed variables such that $\epsilon_i\ N(0, \sigma^2)$.[15] The problem's maximum likelihood function is:

[15] C.S. Ong M.P. Deisenroth, A.A. Faisal. *Mathematics for Machine Learning*. Cambridge University Press, 2020. ISBN 9781108470049

$$L(Y \mid X, \theta, \sigma^2) \propto (\sigma^2)^{-n/2}e^{-\frac{1}{2\sigma^2}(Y-X\theta)^T(Y-X\theta)} \tag{2.33}$$

To maximize it, the expression to minimize is:

$$\min_{\theta} \frac{1}{2}(Y - X\theta)^T(Y - X\theta) \tag{2.34}$$

2.3.2 *Support Vector Classifier*

In support vector classification, the input space is mapped into the feature space, where an optimal hyperplane is given by

$$f(x) = w^T\varphi(x) + b, \tag{2.35}$$

where $\varphi : X \to \mathcal{F}$ is a function that makes each input point x correspond to a point in $\mathcal{F}$, where $\mathcal{F}$ is a Hilbert space. As seen from equation (2.34), OLS utilizes the squared residuals to fit the parameters θ. However, large residuals caused by outliers may worsen the accuracy significantly. [16].

[16] Shigeo Abe. *Support Vector Machines for Pattern Classification, 2 Ed.* Springer-Verlag London, 2010

SVC uses a piecewise linear function to counter this, in which a hyperparameter ϵ, also known as the margin, lets errors that are less or equal to it be 0, and errors greater than it be error $- \epsilon$. Any prediction inside the radius of ϵ counts as a correct prediction. The problem to solve

$$\begin{aligned} \min_{w,b,\xi} \mathcal{P}_\epsilon(w,b,\xi) = \frac{1}{2}w^T w + c\sum_{k=1}^{N} \xi_k \\ \text{s.t. } y_k[w^T\varphi(x_k) - b] \geq 1 - \xi_k, \quad k = 1, ..., N \\ \xi_k \geq 0, \quad k = 1, ..., N \end{aligned} \tag{2.36}$$

The Lagrangian for this problem is given by

$$\begin{aligned} L(w,b,\xi;\alpha,\eta) = \frac{1}{2}w^T w + c\sum_{k=1}^{N}\xi_k \\ - \sum_{k=1}^{N}\alpha_k[y_k(w^T\varphi(x_k) + b) - 1 + \xi_k] \\ - \sum_{k=1}^{N}\eta_k\xi_k \\ \text{s.t.} \quad \alpha, \eta, \succeq 0 \end{aligned} \tag{2.37}$$

where the Lagrange multipliers must be greater than or equal to zero to not disturb the inequalities.

The stationary conditions derived from the problem's Lagragian are the following:

$$\begin{aligned} \nabla_w L = w - \sum_{k=1}^{N}\alpha_k y_k \varphi(x_k) = 0 \\ \Rightarrow w = \sum_{k=1}^{N}\alpha_k y_k \varphi(x_k) \end{aligned} \tag{2.38}$$

$$\frac{\partial L}{\partial b} = -\sum_{k=1}^{N}\alpha_k y_k = 0 \Rightarrow y^T\alpha = 0 \tag{2.39}$$

$$\begin{aligned} \frac{\partial L}{\partial \xi_k} = c - \alpha_k - \eta_k = 0, \quad k = 1, ..., N \\ c - \alpha_k = \eta_k, \quad \eta_k \geq 0 \,\forall\, k \Rightarrow c - \alpha_k \geq 0 \therefore \; \alpha_k \leq c \end{aligned} \tag{2.40}$$

Wolfe's dual problem is obtained from substituting equations (2.38), (2.39) and (2.40) back into (2.37), and its solution is to find the α that maximizes its output.

$$\begin{aligned} D(\alpha) &= -\frac{1}{2}\sum_{k,l=1}^{N} y_k y_l \varphi(x_k)^T \varphi(x_l)\alpha_k\alpha_l + \sum_{k=1}^{N}\alpha_k \\ \text{s.t.} \quad & 0 \preceq \alpha \preceq c \\ & y^T\alpha = 0 \end{aligned} \tag{2.41}$$

When $0 < \alpha_k < c$

$$\eta_k\xi_k = (c-\alpha_k)\xi_k = 0,\ \alpha_k < c\ \therefore\ c - \alpha_k > 0 \Rightarrow \xi_k = 0$$

Thus, the bias term is defined as

$$b = y_k - w^T\varphi(x_k) - \epsilon, \quad 0 < \alpha_k < c \tag{2.42}$$

Parameter b is obtained from averaging all of these possible solutions.

2.3.3 *Gaussian Processes*

Some statistics problems admit a representation through a probability distribution. Gaussian processes are non-parametric regression models that extend the multivariate normal distribution (2.43) and are used to model functions on infinite domains.

$$f(x_1, x_2, ..., x_k) = \frac{\exp(-\frac{1}{2}(\vec{x}-\vec{\mu})\mathbf{\Sigma}^{-1}(\vec{x}-\vec{\mu}))}{\sqrt{(2\pi)^k|\mathbf{\Sigma}|}} \tag{2.43}$$

A Gaussian process is a collection of random variables, any finite number of which have a joint Gaussian distribution [17]. It is defined by its mean and covariance functions of real observations as

[17] C.K.I. Williams C.E. Rasmussen. *Gaussian Processes for Machine Learning*. MIT Press, 2006

$$f(\vec{x}) \sim p(f) = \mathrm{GP}(f; \vec{\mu}, \mathbf{K}) \tag{2.44}$$

where

$$\vec{\mu} = \mathbb{E}[f|\vec{x}] \tag{2.45}$$

$$\mathbf{K}(\vec{x}, \vec{x}') = \mathrm{cov}[f, f'|\vec{x}, \vec{x}'] = \mathbb{E}[(f(\vec{x}) - \vec{\mu})(f(\vec{x}') - \vec{\mu}')] \tag{2.46}$$

$$p(f|\vec{x}) = \mathcal{N}(f; \vec{\mu}, \mathbf{K}) \tag{2.47}$$

Mutual dependence between the random variables in $\vec{x}$ determines the function's shape and statistical properties.

2.3.4 Bayesian Optimization

Bayesian optimization is an approach that utilizes Bayes' Theorem (2.48) to direct a search to find either the minimum or maximum of an objective function. Optimization is accomplished by performing a regression of the low-sampled objective function with a Gaussian process and then using this surrogate function to direct the search. The suggested new point from the surrogate function is evaluated and added back into the Gaussian Process to redirect the search to a new plausible optimal value, chosen depending on the exploration method (probability of improvement, expected improvement, upper-confidence bound).[18]

[18] Roman Garnett. *Bayesian Optimization*. Cambridge University Press, 2022. in preparation

$$P(A|B) = \frac{P(B|A)\ P(A)}{P(B)} \tag{2.48}$$

The method used for this work is upper-confidence bound, which directs the search to the maximum of all the upper limit values from all confidence boundaries.

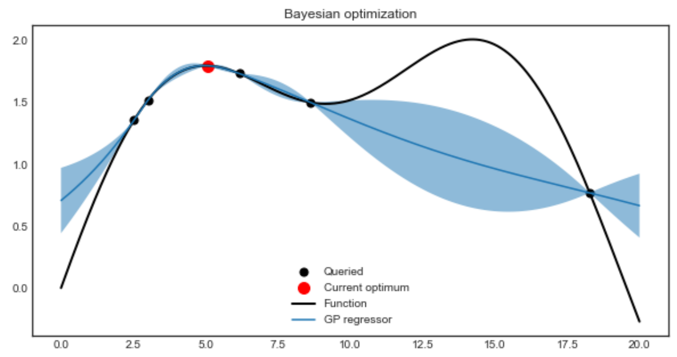


Figure 2.13: Bayesian Optimization - Confidence Interval Plot.

2.3.5 Classification Metrics

Metric is a performance indicator that measures the results obtained from something. In classification problems, you either get the correct prediction or you don't, whereas in regression problems, you can get a wrong value but still be close to the correct answer. We traverse classification errors with the help of the confusion matrix (Fig 2.14), which has the four possible outcomes of any decision. Below are some commonly used metrics for classification problems based on the confusion matrix. It's important to note that these metrics have an inherent bias; thus, a clear understanding of the problem is needed to select which to use. [19]

[19] David Powers. Evaluation: From precision, recall and f-measure to roc, informedness, markedness & correlation. In *Journal of Machine Learning Technologies*, volume 2, 2008

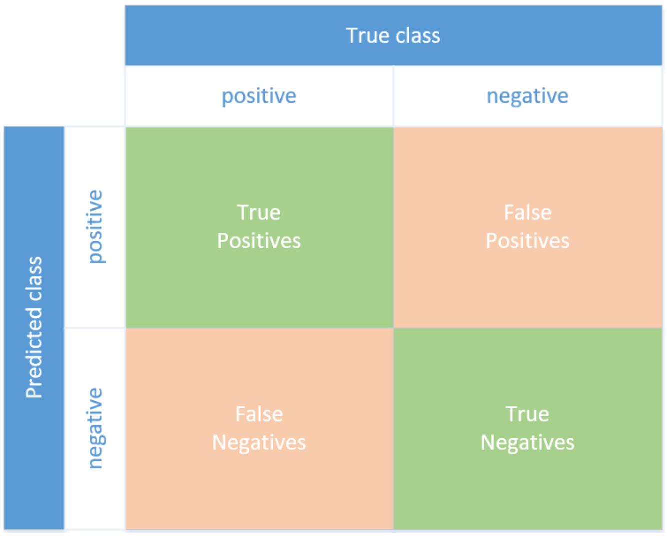


Figure 2.14: Confusion matrix.

- Accuracy: How many positives and negatives were correctly predicted?

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FN} + \text{FP} + \text{TN}} \tag{2.49}$$

- Precision: How many predicted positives were correct?

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2.50}$$

- Recall/Sensitivity: How many positives were correctly predicted out of them all?

$$\text{Recall/Sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2.51}$$

- Specificity: How many negatives were correctly predicted out of them all?

$$\text{Specificity} = \frac{\text{TN}}{\text{FP} + \text{TN}} \tag{2.52}$$

F1 Score

A specific case of the F_β score, giving equal weight to both Precision (2.50) and Recall (2.51). It is used when both False Positives and False Negatives are equally undesirable: Precision is affected by False Positives but not by False Negatives, and Recall is affected by False Negatives but not by False Positives. The F1 score combines these competing metrics through a harmonic mean.

$$F_1 = 2\frac{(\text{Precision})(\text{Recall})}{\text{Precision} + \text{Recall}} \tag{2.53}$$

As seen from equation 2.53, F1 score values will range from 0 to 1:

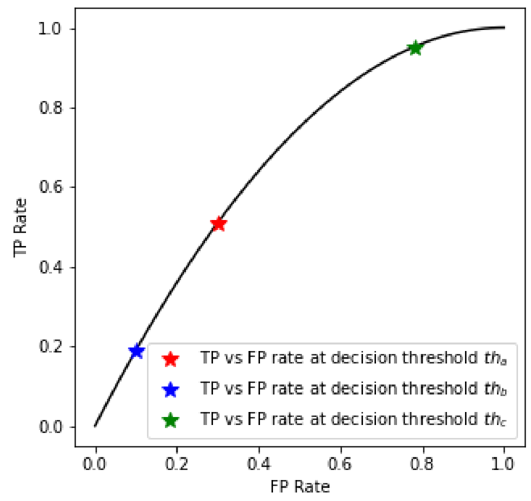
- It will only be 1 if both Precision and Recall are 1.

- It will be 0 if any of Precision or Recall are 0.

ROC and AUC

A Receiver Operating Characteristic curve plots the false positive rate ($1 -$ Specifity 2.52) vs. the true positive rate (Recall/Sensitivity 2.51) at different points, varying the classification threshold (Fig. 2.15).

Figure 2.15: (1-Specificity) vs Sensitivity. (ROC)



AUC stands for Area Under the ROC Curve; it gives insight into the probability of the model classifying a positive sample as more likely than a negative sample (Fig. 2.16). AUC has two important characteristics[20]:

[20] Tom Fawcett. Introduction to roc analysis. In *Pattern Recognition Letters*, volume 27, pages 861–874, 06 2006. DOI: 10.1016/j.patrec.2005.10.010

1. Scale-invariant: Does not care about absolute quantities/values of TP, TN, FP, or FN, but rather measures how well the samples are ranked.
2. Classification-threshold invariant: Because all possible thresholds are considered, AUC will measure the quality of the model itself regardless of what threshold is chosen.

2.3.6 *Radial Basis Function kernel*

Kernels provide a framework to represent data through pairwise comparisons rather than through individual samples. The radial basis function kernel is given by equation (2.54) and yields an infinite summation of polynomials. The RBF kernel is a decreasing function of the Euclidean distance between samples, and can also be used as a measure of similarity. γ is a positive parameter for controlling the radius and was defined as equation (2.55) for this application. [21]

[21] Shigeo Abe. *Support Vector Machines for Pattern Classification, 2 Ed.* Springer-Verlag London, 2010

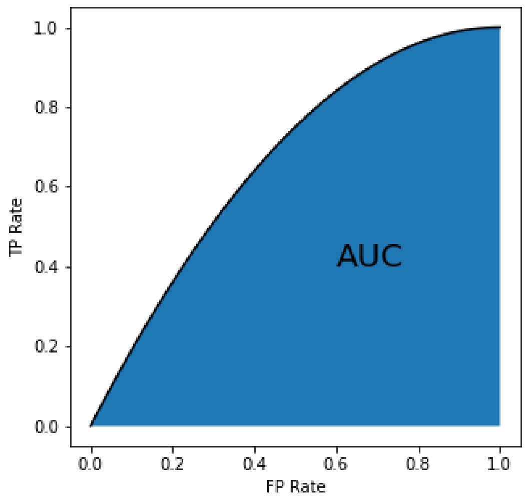


Figure 2.16: Area Under the ROC Curve. (AUC)

$$K(\mathbf{x},\mathbf{x}') = \exp\left(-\gamma\|\mathbf{x}-\mathbf{x}'\|^2\right) \tag{2.54}$$

$$\gamma = \frac{1}{(\text{n\_features})(\text{var}(\mathbf{x}))} \tag{2.55}$$

3 Implementation

Contents

To determine the viability of these automatic control methods as feature generators, we must test them against a benchmark. A transformer class was created for each of the different methods:

- Sobel
- PI Explicit/Implicit
- ST Explicit/Implicit

The chosen method to test these features was a Support Vector Classifier with a Radial Basis Function kernel.

Features	Hyperparameters
Sobel	c
PI	c, h, k_P, k_I
ST	c, h, L

This thesis contains two devised test cases: In the first one, 30 different score samples are compiled from testing the SVC with these features and no additional noise added; then, 60 further samples are collected by adding a high amplitude Gaussian/Uniform noise to the images before extracting the features. The second test case compiles score samples while continuously adding noise to the images. The following points were considered and applied to all tests:

- Data was split in two: 40% for training and 60% for testing. Each run has a distinct splitting to prevent bias from an advantageous/damaging split.

- Hyperparameters were optimized every run using Bayesian Optimization, with 30 loops to probe for the best hyperparameters assortment.

The steps for test one are the following:

Algorithm 3 Test one steps

```
$i \leftarrow 0$
while $i < 30$ do
    Split data set
    Bayesian Optimization for Sobel, PI and ST features.
    Add Uniform noise to the base images ($\pm 100$).
    Bayesian Optimization for Sobel, PI and ST features.
    Add Normal noise to the base images $100\mathcal{N}(\mu = 0, \sigma = 1)$.
    Bayesian optimization for Sobel, PI and ST features.
end while
```

The steps for test two are the following:

Algorithm 4 Test two steps

```
$A \leftarrow 2$
while $i \leq 100$ do
    Split data set
    Add Uniform noise to the base images ($\pm A$).
    Bayesian Optimization for Sobel, PI and ST features.
    Add Normal noise to the base images $A\mathcal{N}(\mu = 0, \sigma = 1)$.
    Bayesian optimization for Sobel, PI and ST features.
end while
```

Forewarning

The implicit PI features showed to be highly sensitive to hyperparameter tuning but are promising because when a sizable amount of time is spent adjusting them, they yield similar results to their explicit counterpart. The Bayesian Optimization module utilized the same range of values for the explicit/implicit hyperparameters to avoid bias, which does not favor these features. Future work can be done with a more finely tuned implicit PI transformer.

3.1 *Animals Dataset*

The dataset contains pictures of the faces of different animals (bears, cats, chickens, cows, deer, dogs, ducks, eagles, elephants, humans, lions, monkeys, mice, pandas, pigeons, pigs, rabbits, sheep, tigers, wolfs) and some unrelated pictures (landscapes). The following image sets were used:

- Wolf
- Tiger
- Sheep
- Cat
- Lion
- Deer
- Mouse
- Cow
- Duck

These images were rescaled to 80x80 pixels, and converted from RGB to grayscale.

$$\text{pixel} = (0.2989R + 0.5870G + 0.1140B)$$

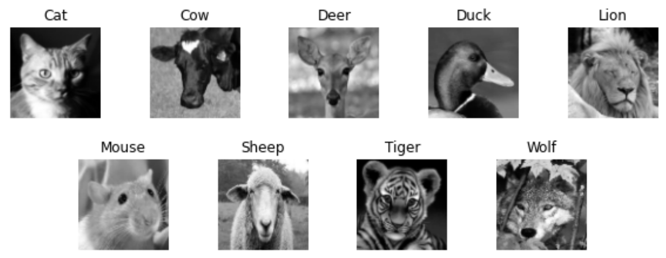


Figure 3.1: Sample of an image from each category.

In the end, the images were each an 80x80 pixels matrix, with values ranging from 0 to 255 (8 bits).

3.1.1 *Results with F1 Score*

With no noise added to the images, the explicit discretization PI features seem to perform better on average (Fig 3.2). A Mann-Whitney U test confirms that the distributions are statistically different, with a 95% confidence interval (Fig 3.3).

When Uniform noise is added, all features' average decreases. Explicit ST features overtake Sobel features (Fig 3.4). Hypothesis tests determine the distributions to have a statistically significant difference 3.5), which conveys that explicit PI and ST features had a better performance.

Gaussian noise results seem to repeat Uniform noise results, with both explicit PI and ST overtaking Sobel 3.6). Once again, hypothesis tests confirm a statistically significant difference in the distributions (3.7).

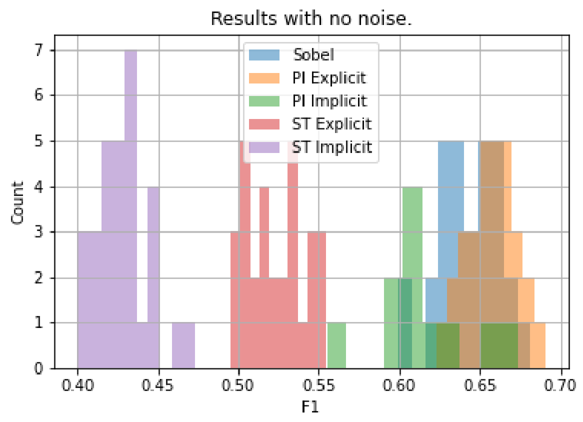


Figure 3.2: Animals DS: Histogram of F1 scores when no noise was added.

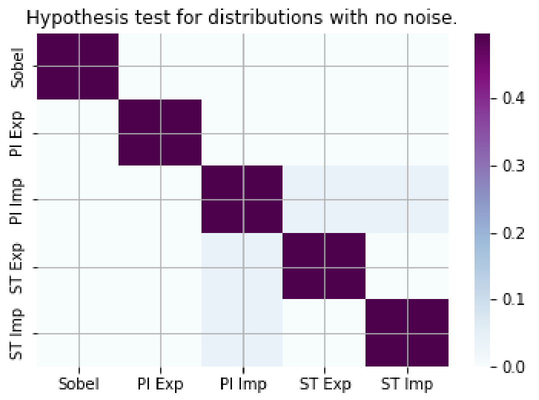


Figure 3.3: Animals DS: Hypothesis tests of F1 scores' distributions when no noise was added.

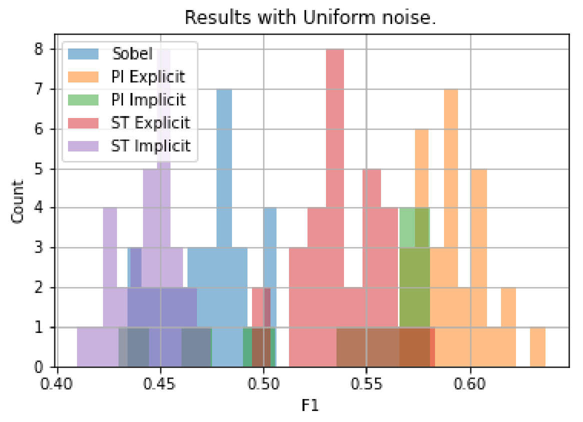


Figure 3.4: Animals DS: Histogram of F1 scores when Uniform noise was added.

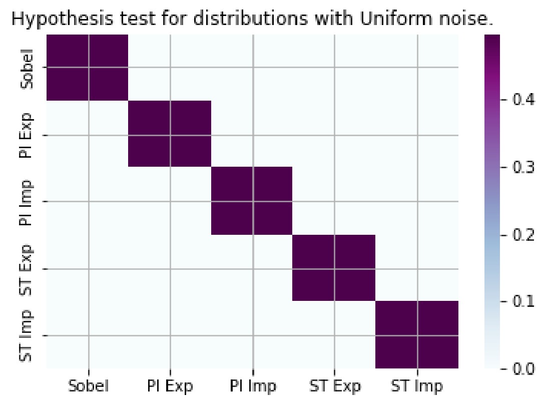


Figure 3.5: Animals DS: Hypothesis tests of F1 scores' distributions when Uniform was added.

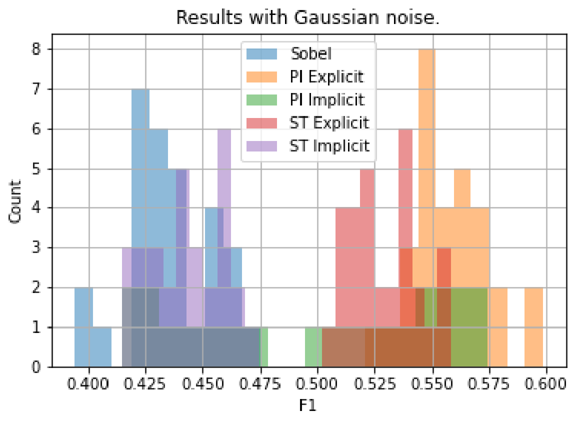


Figure 3.6: Animals DS: Histogram of F1 scores when Gaussian noise was added.

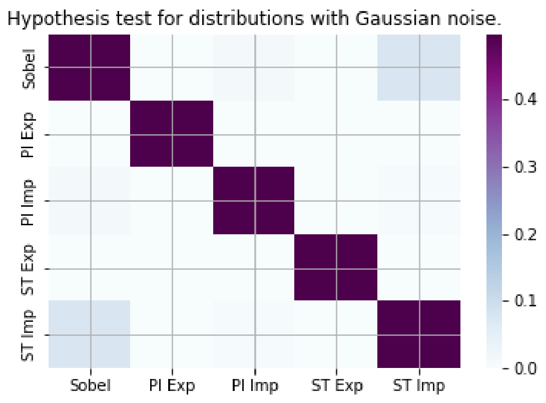


Figure 3.7: Animals DS: Hypothesis tests of F1 scores' distributions when Gaussian noise was added.

For the noises sweep, the ST features seem to be more resilient. Their slope is horizontal. Sobel's slope is steeper than the explicit PI features, and the implicit PI features have promising results occasionally, but its sensitivity to hyperparameter tuning is detrimental (Figures 3.8 & 3.9).

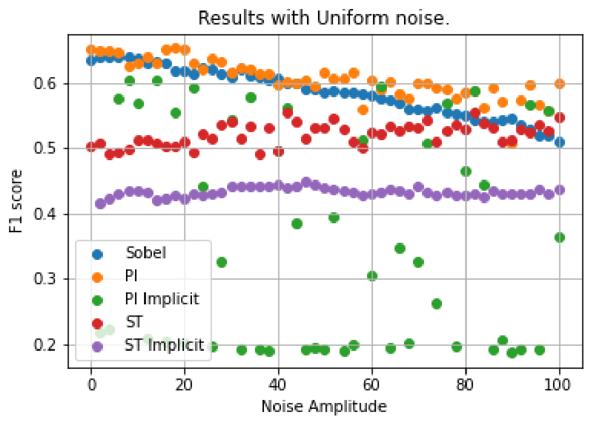


Figure 3.8: Animals DS: Scatter plot of F1 scores when sweeping through Uniform noise.

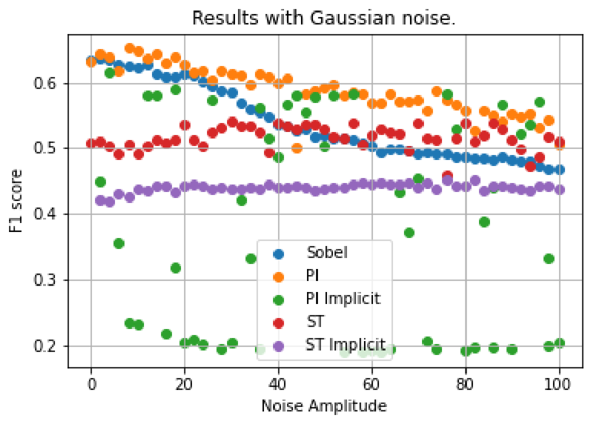


Figure 3.9: Animals DS: Scatter plot of F1 scores when sweeping through Gaussian noise.

3.1.2 Results with AUC Score

As with the F1 results, the explicit PI features surpass Sobel with AUC scoring (Fig 3.10). No statistically significant difference is found in Implicit PI and ST results. (Fig 3.11).

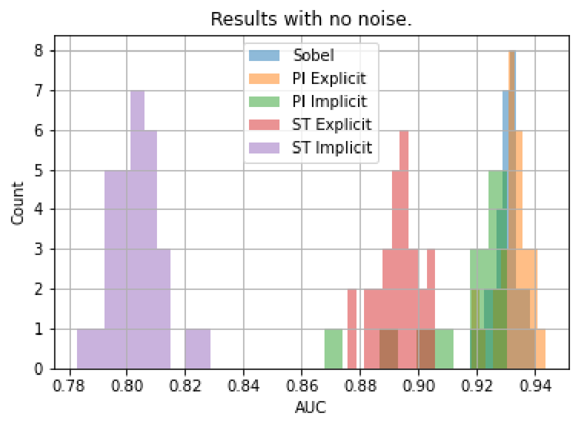


Figure 3.10: Animals DS: Histogram of AUC scores when no noise was added.

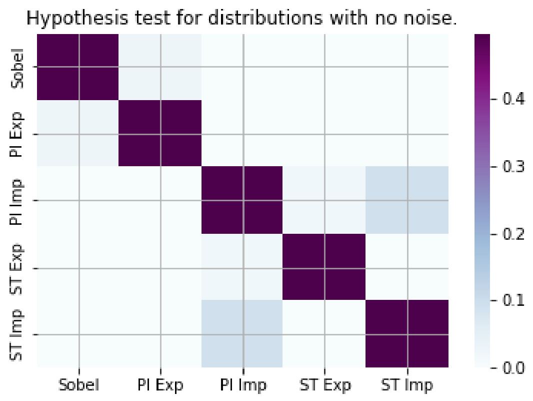


Figure 3.11: Animals DS: Hypothesis tests of AUC scores' distributions when no noise was added.

The distributions from the results when Uniform noise is added (Fig 3.12) produce similar results. Implicit PI results are not statistically different from both Sobel and implicit ST results, and explicit PI results are not statistically different from explicit ST results (Fig 3.13). Explicit PI and ST features lead to a better classification model.

Explicit ST features pull ahead when scored with AUC and added Gaussian noise (Fig 3.14). Results can be ranked: Explicit ST, Explicit PI, Sobel, Implicit ST, Implicit PI, as all distributions are found to be different with statistical significance (Fig 3.15).

Explicit and implicit ST features appear virtually impervious to the added noise sweep, with Sobel and explicit PI features' average decreasing every time the noise is increased (Figs 3.16 & 3.17).

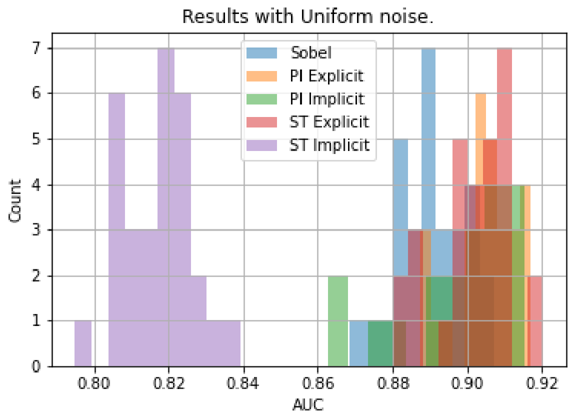


Figure 3.12: Animals DS: Histogram of AUC scores when Uniform noise was added.

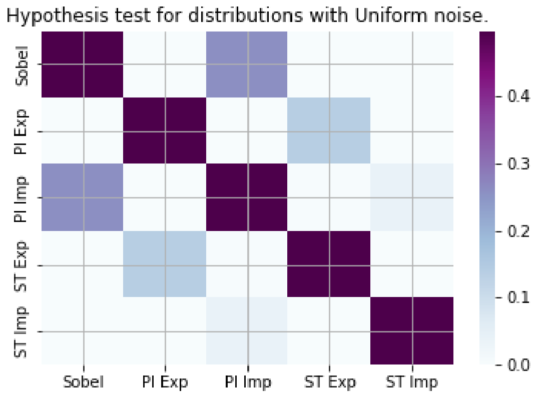


Figure 3.13: Animals DS: Hypothesis tests of AUC scores' distributions when Uniform noise was added.

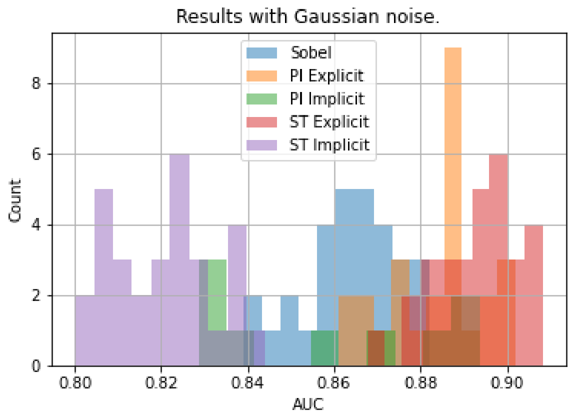


Figure 3.14: Animals DS: Histogram of AUC scores when Gaussian noise was added.

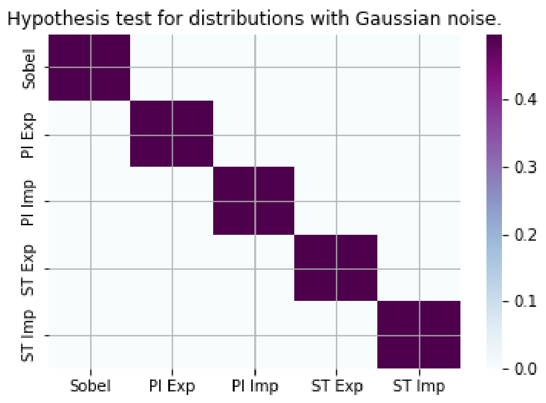


Figure 3.15: Animals DS: Hypothesis tests of AUC scores' distributions when Gaussian noise was added.

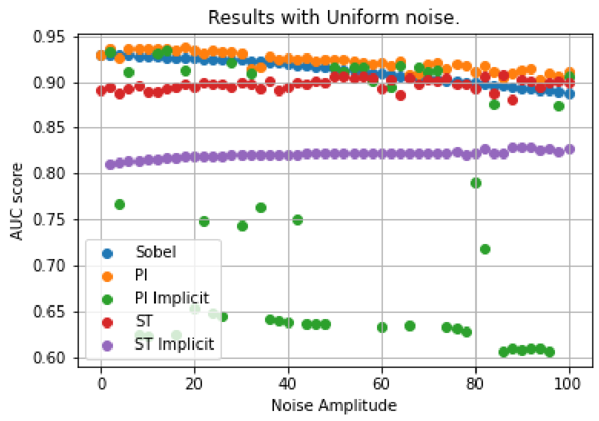


Figure 3.16: Animals DS: Scatter plot of AUC scores when sweeping through Uniform noise.

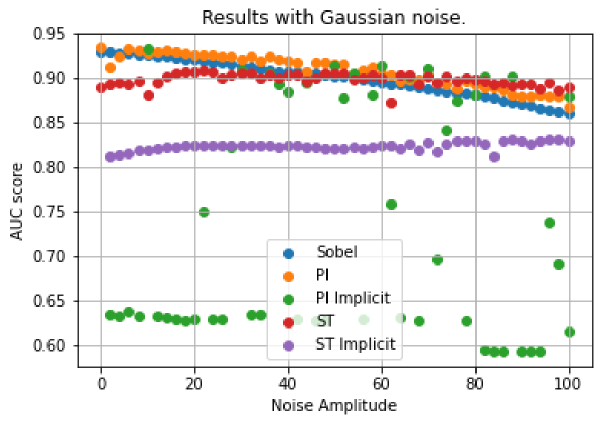


Figure 3.17: Animals DS: Scatter plot of AUC scores when sweeping through Gaussian noise.

3.2 Politicians Dataset

An extract from the Labeled Faces in the Wild people dataset commonly used for classification. The original dataset contains 5749 persons, with 13233 total samples. Each sample is a grayscale image with a size of 47x62 (base x height). We use a portion of the dataset with people for which it contains at least 60 different images. The result is 1348 samples with eight influential politicians from various countries:

- Ariel Sharon
- Colin Powell
- Donald Rumsfeld
- George W Bush
- Gerhard Schroeder
- Hugo Chavez
- Junichiro Koizumi
- Tony Blair

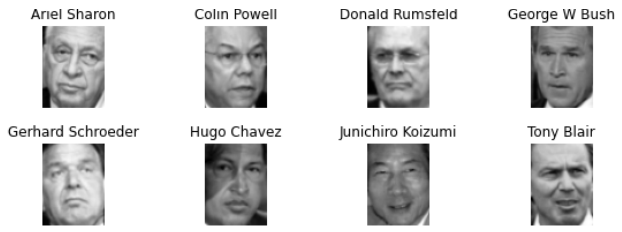


Figure 3.18: Politicians Dataset Samples.

3.2.1 Results with F1 Score

Sobel features pull ahead of explicit PI results with lower variance when no noise is added (Fig 3.19), possibly due to not having extra hyperparameters to adjust. Implicit PI features behave similarly to explicit and implicit ST features (Fig 3.20).

When Uniform noise is added, explicit and implicit PI features pull ahead, followed by implicit ST features and then by Sobel (Fig 3.21). Both distributions for PI features are similar, and implicit features behave similarly. Further work is required here, as it is noteworthy to check if the difference lies in the features themselves (an extra degree of tuning) or if it's just the hyperparameters' sensitivity affecting the distributions (Fig 3.22).

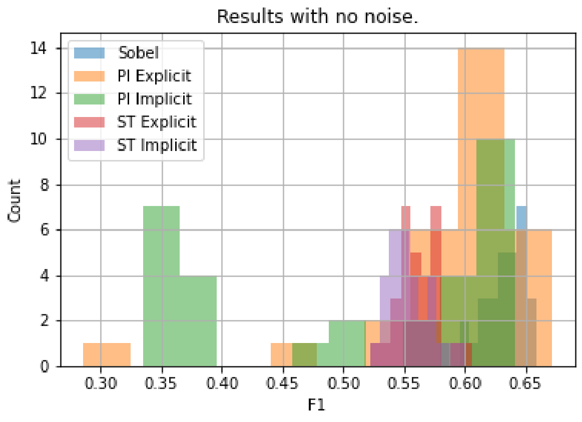


Figure 3.19: Politicians DS: Histogram of F1 scores when no noise was added.

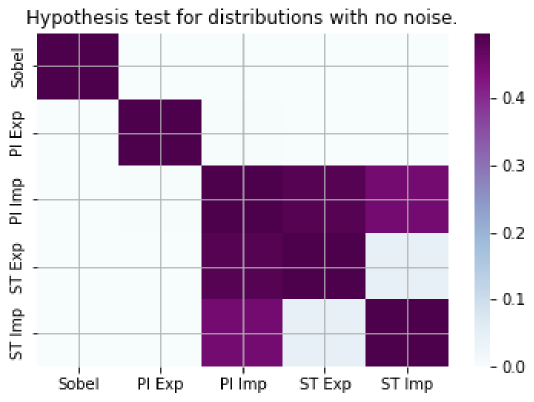


Figure 3.20: Politicians DS: Hypothesis tests of F1 scores' distributions when no noise was added.

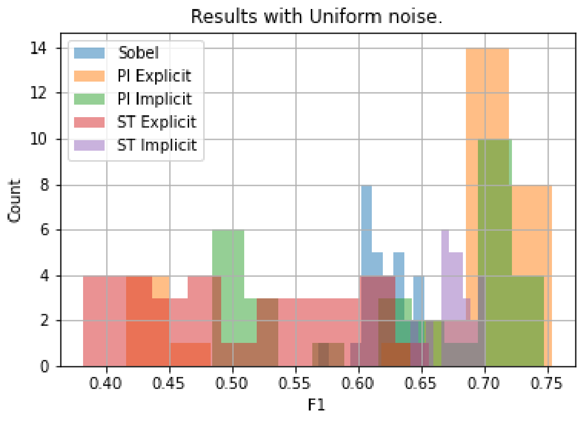


Figure 3.21: Politicians DS: Histogram of F1 scores when Uniform noise was added.

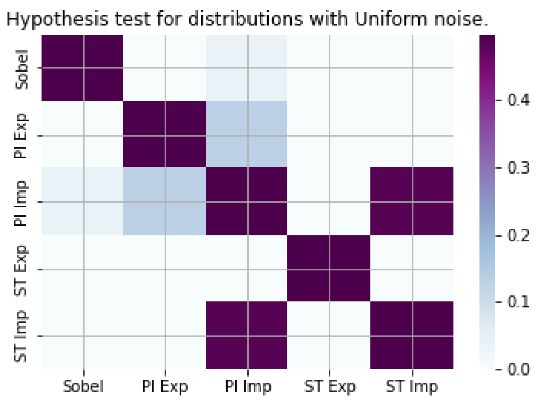


Figure 3.22: Politicians DS: Hypothesis tests of F1 scores' distributions when Uniform noise was added.

Regarding Gaussian noise results, implicit PI features appear to have the potential to yield better results than all the others (Fig 3.23). Explicit PI and Sobel results are not as sensitive as having a hidden combination of hyperparameters that could provide a higher score. Implicit PI features behave similarly to explicit PI and implicit ST features (Fig 3.24).

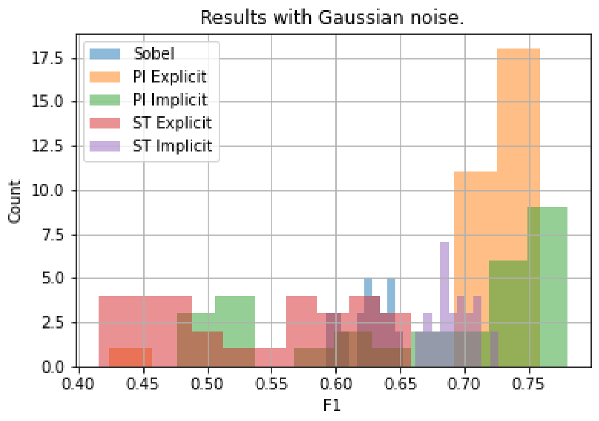


Figure 3.23: Politicians DS: Histogram of F1 scores when Gaussian noise was added.

For these images, the noise acts as a regularization, enabling the SVC able to generalize better, which causes a couple of features to have a positive slope for the noise sweep tests (Figs 3.25 & 3.26).

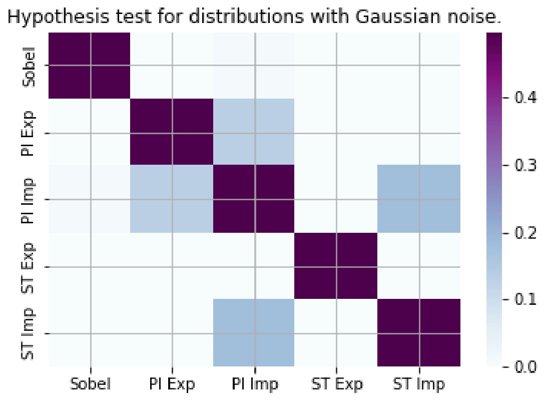


Figure 3.24: Politicians DS: Hypothesis tests of F1 scores' distributions when Gaussian noise was added.

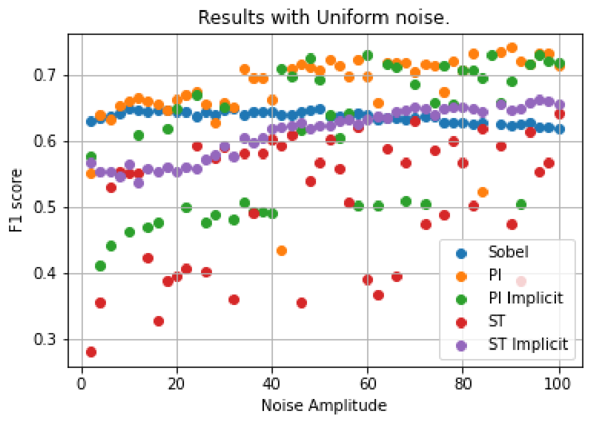


Figure 3.25: Politicians DS: Scatter plot of F1 scores when sweeping through Uniform noise.

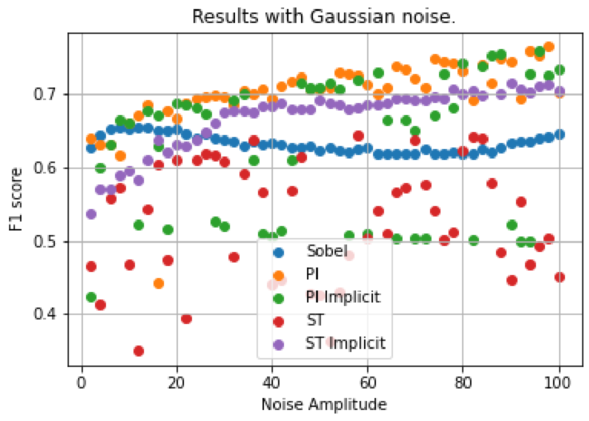


Figure 3.26: Politicians DS: Scatter plot of F1 scores when sweeping through Gaussian noise.

3.2.2 *Results with AUC Score*

Scoring with AUC and not adding noise results in almost all features behaving similarly (Fig 3.27). Explicit PI & ST and implicit PI features behave comparably, and only Sobel and implicit ST features maintain diverse behaviors (Fig 3.28).

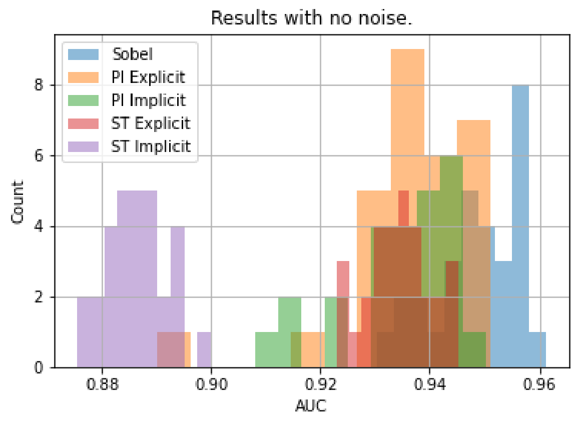


Figure 3.27: Politicians DS: Histogram of AUC scores when no noise was added.

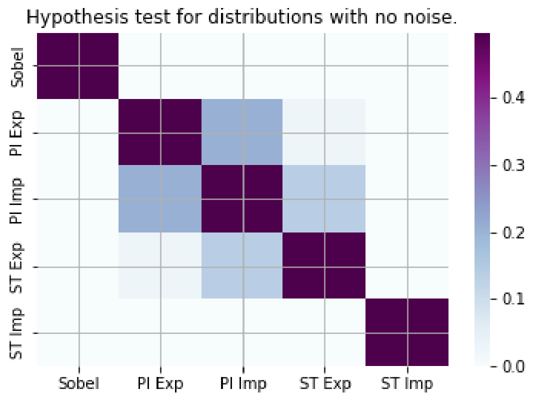


Figure 3.28: Politicians DS: Hypothesis tests of AUC scores' distributions when no noise was added.

When Uniform noise is added to this dataset, Sobel maintains its score, but the noise has a positive result on the other features (Fig 3.29). The PI features behave similarly, as do the ST features (Fig 3.30).

Results with Gaussian noise favor explicit PI results, followed by implicit PI and Sobel results (Fig 3.31). Implicit PI results seem statistically similar to most others (Fig 3.32).

With this dataset, the noise resulted in a better generalization of the SVC for AUC scores (Figs 3.33 & 3.34).

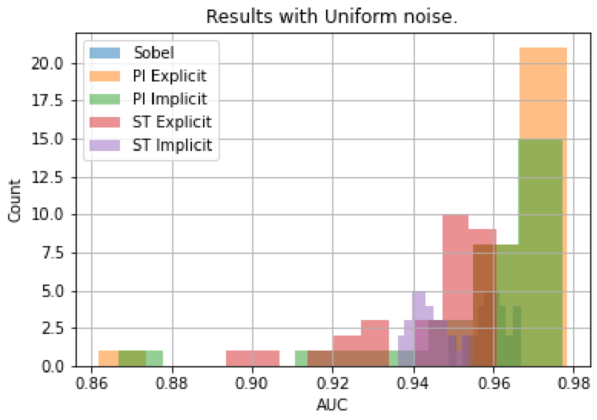


Figure 3.29: Politicians DS: Histogram of AUC scores when Uniform noise was added.

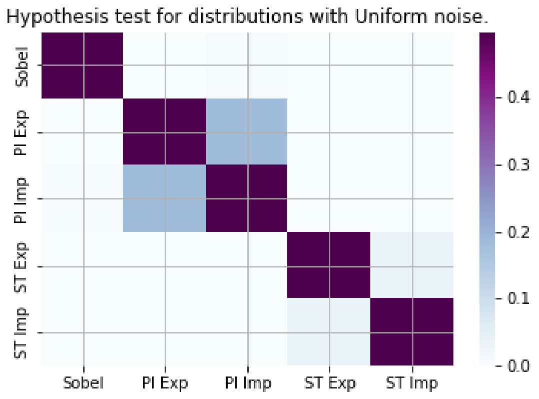


Figure 3.30: Politicians DS: Hypothesis tests of AUC scores' distributions when Uniform noise was added.

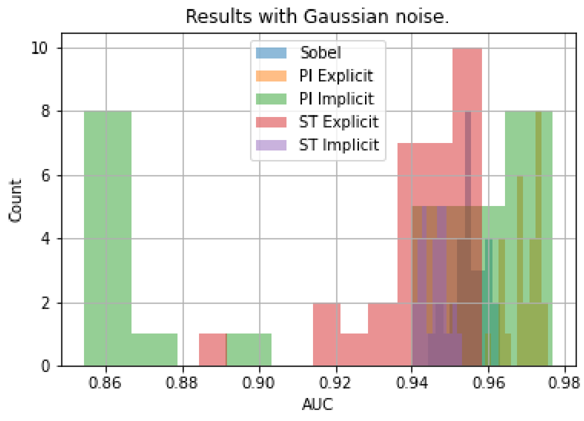


Figure 3.31: Politicians DS: Histogram of AUC scores when Gaussian noise was added.

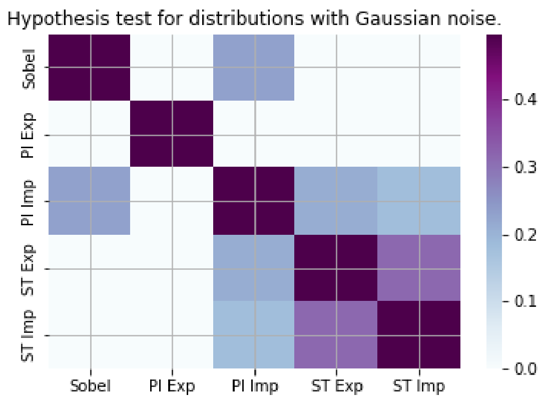


Figure 3.32: Politicians DS: Hypothesis tests of AUC scores' distributions when Gaussian noise was added.

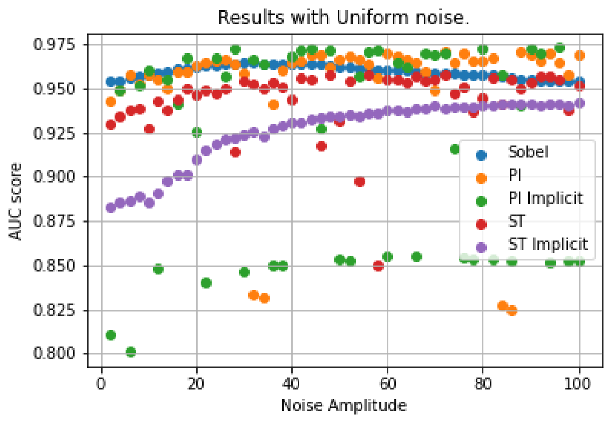


Figure 3.33: Politicians DS: Scatter plot of AUC scores when sweeping through Uniform noise.

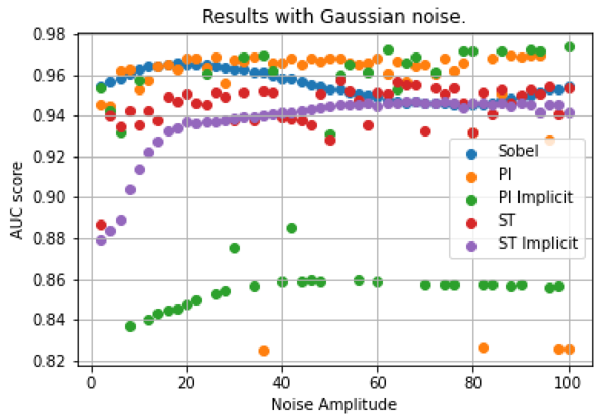


Figure 3.34: Politicians DS: Scatter plot of AUC scores when sweeping through Gaussian noise.

4 Results, Conclusions and Future Work

Contents

This section will contain insight from the results from section 3, the conclusions drawn concerning the questions asked in this thesis' problem statement, and what future work can be done to advance this investigation further.

4.1 Results

The question this work was centered on was "Are Automatic Control Algorithms viable to extract features from images to feed machine learning algorithms?". The answer obtained from the results is yes. Not only do some of these features behave comparably to those obtained from the Sobel kernel, but they occasionally surpass it. The distribution plots are a good visualization tool but are hard to interpret, so table representations for all results are shown.

First, with the exception of a 19.79% higher standard deviation in AUC results with no noise (Table 4.4) and a 0.82% higher standard deviation in AUC results with Gaussian noise (Table 4.6), the explicit PI features outperformed Sobel with higher mean, minimum, quartiles and maximum, and lower standard deviation in all of the sampling tests with the animal dataset. There's a statistically significant difference between these distributions per the hypothesis tests in section 3.1.2.

Explicit ST features outperformed the Sobel features when noise was added, with the exception of 0.74% higher standard deviation in F1 results with Uniform noise (Table 4.2) and 12.34% higher standard deviation in AUC results with Uniform noise (Table 4.5). There's a statistically significant difference between these distributions per the hypothesis tests in sections 3.1.1 and 3.1.2.

Sobel remains unchanging in all the sampling tests on the politicians' dataset, with a very low standard deviation. The automatic control algorithms had a more challenging time getting stable results and had a more comprehensive range of scores, with the exception of implicit ST features. PI features always had a higher maximum obtained value than Sobel for tests when noise was added. ST features had a higher maximum obtained value for tests with F1 scores and added noise.

Table 4.1: Animals Dataset: Compilation of F1 results with no noise

	Sobel	PI Exp	PI Imp	ST Exp	ST Imp
count	30	30	30	30	30
mean	0.642107	0.657804	0.358367	0.523504	0.428740
std	0.019519	0.015514	0.201677	0.017582	0.016628
min	0.599568	0.623143	0.192736	0.495378	0.400048
25%	0.626935	0.649519	0.199480	0.508153	0.417775
50%	0.643380	0.660579	0.212891	0.521776	0.428530
75%	0.657345	0.666159	0.602748	0.534524	0.436239
max	0.681353	0.690407	0.673339	0.554765	0.473129

Table 4.2: Animals Dataset: Compilation of F1 results with Uniform noise

	Sobel	PI Exp	PI Imp	ST Exp	ST Imp
count	30	30	30	30	30
mean	0.473029	0.591369	0.316817	0.539295	0.445350
std	0.020600	0.016167	0.152751	0.020753	0.014599
min	0.434242	0.566353	0.180564	0.494719	0.410158
25%	0.460832	0.578070	0.201055	0.528117	0.436222
50%	0.477540	0.589666	0.218071	0.537054	0.447127
75%	0.485813	0.601452	0.463261	0.555430	0.454649
max	0.506889	0.636355	0.580973	0.583408	0.474330

Table 4.3: Animals Dataset: Compilation of F1 results with Gaussian noise

	Sobel	PI Exp	PI Imp	ST Exp	ST Imp
count	30	30	30	30	30
mean	0.435821	0.558269	0.335545	0.529536	0.443108
std	0.020031	0.017456	0.150237	0.015052	0.016715
min	0.394162	0.521004	0.178326	0.502241	0.414837
25%	0.422339	0.547027	0.190862	0.516163	0.428140
50%	0.434335	0.556163	0.308389	0.525250	0.443026
75%	0.452158	0.570624	0.468066	0.540455	0.457204
max	0.475329	0.598312	0.574553	0.558055	0.474171

	Sobel	PI Exp	PI Imp	ST Exp	ST Imp
count	30	30	30	30	30
mean	0.930166	0.932372	0.745026	0.892874	0.803584
std	0.004891	0.005859	0.144050	0.007551	0.009234
min	0.917780	0.918222	0.601326	0.875632	0.783010
25%	0.928064	0.930195	0.619482	0.888999	0.799262
50%	0.931127	0.932661	0.642338	0.893608	0.803548
75%	0.932595	0.935760	0.917016	0.897249	0.807513
max	0.940424	0.943385	0.930669	0.905920	0.828811

Table 4.4: Animals Dataset: Compilation of AUC results with no noise

	Sobel	PI Exp	PI Imp	ST Exp	ST Imp
count	30	30	30	30	30
mean	0.890085	0.905066	0.803633	0.901884	0.816900
std	0.009134	0.008095	0.129807	0.010262	0.009571
min	0.868521	0.887279	0.606125	0.880141	0.794940
25%	0.883268	0.901902	0.646317	0.897241	0.811495
50%	0.890046	0.906110	0.885124	0.903961	0.818380
75%	0.897499	0.909877	0.904811	0.908511	0.822141
max	0.907085	0.916834	0.915307	0.920065	0.839250

Table 4.5: Animals Dataset: Compilation of AUC results with Uniform noise

	Sobel	PI Exp	PI Imp	ST Exp	ST Imp
count	30	30	30	30	30
mean	0.863796	0.883641	0.698364	0.893077	0.820378
std	0.010827	0.010916	0.116385	0.009523	0.011751
min	0.838941	0.861075	0.589132	0.867689	0.800007
25%	0.858589	0.876447	0.604661	0.886230	0.811819
50%	0.864579	0.886615	0.627801	0.894088	0.821740
75%	0.871638	0.888778	0.832762	0.899952	0.829327
max	0.881989	0.901776	0.893633	0.908173	0.844165

Table 4.6: Animals Dataset: Compilation of AUC results with Gaussian noise

	Sobel	PI Exp	PI Imp	ST Exp	ST Imp
count	30	30	30	30	30
mean	0.632175	0.594158	0.507467	0.561445	0.554639
std	0.018527	0.072307	0.122441	0.017336	0.018410
min	0.579699	0.285882	0.334742	0.522350	0.522179
25%	0.621519	0.581588	0.365830	0.550276	0.540679
50%	0.635879	0.615826	0.568082	0.558491	0.551816
75%	0.645784	0.629679	0.614919	0.573638	0.566590
max	0.659052	0.671475	0.641180	0.605608	0.601200

Table 4.7: Politicians Dataset: Compilation of F1 results with no noise

	Sobel	PI Exp	PI Imp	ST Exp	ST Imp
count	30	30	30	30	30
mean	0.620200	0.665734	0.635351	0.504582	0.675453
std	0.018464	0.105985	0.096182	0.081249	0.015363
min	0.568133	0.415946	0.483839	0.381916	0.641394
25%	0.606947	0.685424	0.527701	0.435827	0.668287
50%	0.618557	0.702830	0.674089	0.489354	0.674431
75%	0.631457	0.727642	0.714594	0.574440	0.684833
max	0.652962	0.753132	0.747902	0.656394	0.701253

Table 4.8: Politicians Dataset: Compilation of F1 results with Uniform noise

	Sobel	PI Exp	PI Imp	ST Exp	ST Imp
count	30	30	30	30	30
mean	0.624911	0.721796	0.667160	0.533802	0.691709
std	0.017913	0.058879	0.104365	0.078451	0.015950
min	0.592068	0.423808	0.476718	0.415903	0.661450
25%	0.611451	0.717235	0.595115	0.459854	0.682070
50%	0.626573	0.732711	0.717484	0.526450	0.690532
75%	0.637495	0.743629	0.753589	0.595673	0.702135
max	0.652522	0.758982	0.779504	0.658261	0.726534

Table 4.9: Politicians Dataset: Compilation of F1 results with Gaussian noise

	Sobel	PI Exp	PI Imp	ST Exp	ST Imp
count	30	30	30	30	30
mean	0.948755	0.936795	0.934812	0.934348	0.886408
std	0.007678	0.011913	0.011072	0.006076	0.005839
min	0.930242	0.890063	0.908227	0.923038	0.875525
25%	0.944769	0.933290	0.931052	0.930639	0.882247
50%	0.949990	0.938284	0.937352	0.935084	0.885835
75%	0.955597	0.944810	0.942425	0.938195	0.890083
max	0.961130	0.951160	0.950222	0.945205	0.899924

Table 4.10: Politicians Dataset: Compilation of AUC results with no noise

	Sobel	PI Exp	PI Imp	ST Exp	ST Imp
count	30	30	30	30	30
mean	0.959473	0.964839	0.959148	0.943571	0.943912
std	0.004511	0.019908	0.022742	0.017375	0.004436
min	0.948819	0.861910	0.866843	0.893433	0.936168
25%	0.956749	0.966072	0.957608	0.934968	0.941007
50%	0.959653	0.969049	0.966288	0.949365	0.943431
75%	0.962778	0.970423	0.971558	0.956037	0.946742
max	0.966712	0.978124	0.976960	0.960747	0.953799

Table 4.11: Politicians Dataset: Compilation of AUC results with Uniform noise

Table 4.12: Politicians Dataset: Compilation of AUC results with Gaussian noise

	Sobel	PI Exp	PI Imp	ST Exp	ST Imp
count	30	30	30	30	30
mean	0.954950	0.969697	0.926631	0.941808	0.945971
std	0.004471	0.004213	0.048996	0.014755	0.003391
min	0.944428	0.959288	0.854462	0.884453	0.940118
25%	0.953055	0.967710	0.862288	0.938856	0.943157
50%	0.954988	0.970767	0.951159	0.945471	0.946030
75%	0.957789	0.973078	0.964964	0.952028	0.948235
max	0.963004	0.975641	0.976612	0.958268	0.953173

4.2 Conclusions

- Automatic control structures such as Proportional-Integral and Super-Twisting filters, with both explicit and implicit discretization, are viable feature generators.
- The features generated with these filters behave similarly or better than a benchmark kernel commonly used to generate edge information.
- These features require additional hyperparameters to be tuned, and the implicit discretization methods are much more sensitive to any difference than their explicit discretization counterparts.
- Like tools in a toolbox, sometimes a screwdriver will outperform a hammer, which doesn't mean the hammer is useless. Some features behaved better with the animals' dataset and others with the politicians' dataset.
- The physical limitations of the automatic control algorithms filtered added noise and provided steady and stable results, which means these features could be better suited for applications where characterized noise is expected.

4.3 Future Work

Knowing that these features are viable for machine learning algorithms and applications, many paths open from the available controllers/observers in automatic control theory and machine learning algorithms and applications.

- Further investigation regarding the sensitivity of implicit methods is required. Could implicit methods work all-around better than explicit methods? Are they just different tools to be used on different occasions?
- There are more ways to discretize differential equations. Would these be different, better features than the ones obtained from using the Forward/Backward Euler Method?
- All these features were trained on a convex model. Non-convex methods such as Neural Networks create their features in hidden layers, but they are still susceptible to the input. Could these features yield consistently better results in noisy situations with non-convex machine learning methods?
- The current implementation of the automatic control features uses one row/column at a time, which means diagonal edges may be

improved. Sobel, for instance, checks edges three rows/columns at a time. Could these algorithms be modified so that the number of rows/columns used was an additional hyperparameter?

- These features can be used with the Deep Image prior method, which consists of enhancing and removing noise from an image given an untrained neural network.
- The step-size hyperparameter could allow us to focus the image differently, like a camera's lens. Would a more appropriate use of these features be object identification in images where much is happening?

Bibliography

Shigeo Abe. *Support Vector Machines for Pattern Classification, 2 Ed.* Springer-Verlag London, 2010.

S. Bennett. A brief history of automatic control. In *IEEE Control Systems Magazine Vol. 16*, number 3, pages 17–25, 1996. DOI: 10.1109/37.506394.

Denis Efimov Bernard Brogliato, Andrey Polyakov. The implicit discretization of the super-twisting sliding-mode control algorithm. In *15th International Workshop on Variable Structure Systems*, pages 349–353, july 2018.

C.K.I. Williams C.E. Rasmussen. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

Jorge Lira Chávez. *Tratamiento digital de imágenes multiespectrales*. UNAM, 2010. ISBN 978-607-00-3403-9.

E. Hildreth D. Marr. *Theory of edge detection*. Proc. R. Soc. Lond., 1980. DOI: 10.1098/rspb.1980.0020.

Jorge Rivera et al. *Sliding Mode Control*. IntechOpen, 2011. ISBN 978-953-51-6002-1.

Kendall E. Atkinson et al. *Numerical solution of ordinary differential equations*. John Wiley & Sons, Inc., 2009. ISBN 978-111-81-6449-5.

Benjamin C. Kuo Farid Golnaraghi. *Automatic Control Systems*. John Wiley & Sons INC, 2010. ISBN 978-0470-04896-2.

Tom Fawcett. Introduction to roc analysis. In *Pattern Recognition Letters*, volume 27, pages 861–874, 06 2006. DOI: 10.1016/j.patrec.2005.10.010.

Roman Garnett. *Bayesian Optimization*. Cambridge University Press, 2022. in preparation.

G. Feldman I. Sobel. A 3x3 isotropic gradient operator for image processing. pages 271–272, 1968.

Tore Hagglund Karl J. Astrom. *PID Controllers*. Instrument Society of America, 1995. ISBN 1-55617-516-7.

A. Levant. Sliding order and sliding accuracy in sliding mode control. In *International Journal of Control Vol. 58*, pages 1247–1263, 1993.

A. Levant. Higher-order sliding modes, differentiation and output-feedback control. In *International Journal of Control Vol. 76*, number 6, pages 924–941, 2003.

George Stockman Linda Shapiro. *Computer Vision*. Prentice Hall, 2001. ISBN 978-0130307965.

J.C. Maxwell. On governors. In *Proceedings of The Royal Society of London*, number 10, pages 270–283, 1868.

C.S. Ong M.P. Deisenroth, A.A. Faisal. *Mathematics for Machine Learning*. Cambridge University Press, 2020. ISBN 9781108470049.

David Powers. Evaluation: From precision, recall and f-measure to roc, informedness, markedness & correlation. In *Journal of Machine Learning Technologies*, volume 2, 2008.

Richard E. Woods Rafael C. Gonzalez. *Digital Image Processing*. Pearson Education Limited, 2018. ISBN 978-1-292-22304-9.

Scott E. Umbaugh. *Computer imaging: digital image analysis and processing*. CRC Press, 2005. ISBN 0-8493-2919-1.

Mark Burge Wilhelm Burger. *Principles of Digital Image Processing: Fundamental Techniques*. Springer-Verlag London, 2009. ISBN 978-1-84800-190-9.