



AUTOMATIC CHANGE DETECTION IN MOVING CAMERA VIDEOS

Rafael Padilla

Tese de Doutorado apresentada ao Programa de Pós-graduação em Engenharia Elétrica, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Engenharia Elétrica.

Orientadores: Eduardo Antônio Barros da
Silva
Sergio Lima Netto

Rio de Janeiro
Outubro de 2021

AUTOMATIC CHANGE DETECTION IN MOVING CAMERA VIDEOS

Rafael Padilla

TESE SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA ELÉTRICA.

Orientadores: Eduardo Antônio Barros da Silva
Sergio Lima Netto

Aprovada por: Prof. Eduardo Antônio Barros da Silva
Prof. Sergio Lima Netto
Prof. José Gabriel Rodriguez Carneiro Gomes
Prof. Lisandro Lovisolo
Prof. Bruno Luigi Macchiavello Espinoza

RIO DE JANEIRO, RJ – BRASIL
OUTUBRO DE 2021

Padilla, Rafael

Automatic change detection in moving camera videos/Rafael Padilla. – Rio de Janeiro: UFRJ/COPPE, 2021.

XX, 177 p.: il.; 29, 7cm.

Orientadores: Eduardo Antônio Barros da Silva

Sergio Lima Netto

Tese (doutorado) – UFRJ/COPPE/Programa de Engenharia Elétrica, 2021.

Referências Bibliográficas: p. 156 – 177.

1. Anomaly Detection. 2. Video Processing. 3. Signal Processing. 4. Moving Camera. I. Antônio Barros da Silva, Eduardo *et al.* II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Elétrica. III. Título.

*Aos meus pais, irmãos e avós,
que exemplarmente repassaram a
importância da família, da
honestidade, do trabalho e da
persistência.*

*À Lena e à Jujuba, por
abraçarem comigo este sonho,
pela confiança e por acreditarem
que NÓS conseguiríamos.*

Agradecimentos

Quando as coisas estão difíceis, o suor, a luta, o trabalho e a persistência que muitos precisaram ter para que eu possa desfrutar todas oportunidades, são as maiores razões pelas quais eu continuarei sempre resistindo. Por isso, agradeço aos meus avós Vô Chico, Vô Maria, Vô Osmar e Vô Tereza, e aos meus pais Roberto e Suzan, por todo o esforço e renúncias que precisaram fazer para que eu e meus irmãos recebêssemos uma boa educação e bons valores. Jamais os decepcionarei.

Aos meus irmãos Vinicius e Rodrigo, por serem minha melhor lembrança e conexão com a melhor época de nossas vidas.

Agradeço à Lena, minha querida e amada esposa, pela enorme paciência e, principalmente pelo companheirismo e amor. O seu apoio, compreensão e confiança foram fundamentais para que eu realizasse este trabalho. Esta é uma conquista sua.

À Jujuba pelo carinho sincero, brincadeiras, paciência e amizade que só cresce todos os dias.

Aos meus amigos Jorge, Mike, Veronica, Janica e Nick pela amizade que há anos mantemos, provando que a distância e o tempo não são capazes de esmaecer nosso companheirismo.

Ao meu grande amigo Zé Maria que "através da busca levada pelo físico, metafísico, o ilusório... e de volta faz a descoberta mais importante de sua carreira".

A todos professores que tive até hoje, pois sempre me proporcionaram o prazer do aprendizado e do questionamento.

Aos meus colegas e amigos do laboratório SMT pelo acolhimento, ajuda, ensinamentos e momentos de descontração que tivemos.

Aos professores Eduardo Antônio Barros da Silva e Sergio Lima Netto, meus queridos orientadores, por permitirem que eu faça parte do grupo dos seus orientandos, pelas oportunidades que me deram de participar de projetos que surgiram ao longo do caminho, pela paciência, pela humanidade, e principalmente por todo aprendizado que recebi durante o doutorado. "Se eu vi mais longe, foi por estar sobre ombros de gigantes". Meus sinceros e profundos agradecimentos. Tenho enorme admiração por vocês.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

DETECÇÃO AUTOMÁTICA DE MUDANÇAS EM VÍDEOS COM CÂMERAS MÓVEIS

Rafael Padilla

Outubro/2021

Orientadores: Eduardo Antônio Barros da Silva
Sergio Lima Netto

Programa: Engenharia Elétrica

Neste trabalho, investiga-se a aplicação de técnicas de visão computacional baseadas em aprendizado de máquina como soluções para o problema de detecção de anomalias em vídeos capturados com câmeras móveis. Este problema é visto pela perspectiva de uma base formada por vídeos de referência, sem a presença de anomalias, e vídeos alvo, onde anomalias estão presentes em alguns quadros. Neste trabalho o termo ‘anomalias’ representa mudanças nos quadros dos vídeos alvos pela inclusão de objetos inexistentes nos quadros dos vídeos de referência. Assim, as técnicas aqui apresentadas têm como objetivo a classificação dos quadros dos vídeos alvos como anômalos ou não anômalos através de características extraídas dos quadros previamente alinhados.

Inicialmente, uma técnica de alinhamento temporal dos vídeos é proposta. Utilizando um trabalho anterior como base, um algoritmo de classificação é retreinado, e seu conjunto de hiperparâmetros é encontrado através de uma busca direcionada utilizando otimização bayesiana. Para evitar contaminação dos dados, os vídeos são separados em grupos distintos para treino, validação e teste. Para melhorar o desempenho do algoritmo de classificação, técnicas que utilizam a análise de componentes principais (em inglês PCA) são aplicadas como critérios de seleção das características extraídas dos quadros.

Uma nova técnica de alinhamento geométrico dos quadros e uma rede separada em módulos são apresentadas. Nesta rede, um módulo de morfologia diferenciável apresenta originalidade em relação a outras abordagens existentes na literatura, permitindo a realização de operações morfológicas de forma diferenciável. Os resultados obtidos nesta proposta são superiores aos resultados de trabalhos anteriores.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

AUTOMATIC CHANGE DETECTION IN MOVING CAMERA VIDEOS

Rafael Padilla

October/2021

Advisors: Eduardo Antônio Barros da Silva
Sergio Lima Netto

Department: Electrical Engineering

In this work, we investigate the application of computer vision techniques based on machine learning as solutions to the anomaly detection problem in videos with moving cameras. This problem is analyzed through a dataset consisting of reference and target videos. In this work, the term ‘anomalies’ represent changes on frames of target videos by adding object that were not present in the reference videos. Reference videos do not contain anomalies, which are existent in some frames of the target videos. The techniques presented here aim to classify the target frames as anomalous or not anomalous through features extracted from previously aligned frames.

Initially, a temporal alignment technique is proposed. Based on a previous work, a classification algorithm is retrained, and its set of hyperparameters is optimized through a Bayesian optimization process. To avoid data contamination, the videos are split into folds containing distinct train, validation, and testing sets. As an attempt to improve the classification algorithm, techniques applying principal component analysis (PCA) are applied as selection criteria of the features extracted from the frames.

A novel technique composed of a geometrical alignment of the frames and modular network is presented. In this network, a differentiable morphology module is highlighted as it proposes a novel approach in computing morphological operations in a differentiable way. The results achieved with our proposed technique overcome results from previous works.

Contents

List of Figures	xi
List of Tables	xv
List of Symbols	xvii
List of Abbreviations	xix
1 Introduction	1
1.1 Objectives	2
1.2 Contributions	3
1.3 This Work	4
2 Anomaly Detection in Cluttered Environments	6
2.1 Anomaly Detection	6
2.2 Review of Anomaly Detection Works	7
2.3 Available Databases	10
2.3.1 The VDAO Database	11
2.4 Anomaly Detection with VDAO Database	15
2.4.1 Using Spatio-Temporal Codebooks	15
2.4.2 Sparse-Representation of the VDAO	17
2.4.3 Extracting Features from VDAO Videos with CNNs	18
2.4.4 Experimental Results with VDAO Database	19
2.5 Conclusion	21
3 Deep Learning-Based Anomaly Detection	23
3.1 Temporal Video Alignment	24
3.2 Extracting Features with Resnet-50	29
3.3 Random Forest for Image Classification	33
3.4 Hyperparameter Optimization	35
3.5 Database Subdivision	36
3.6 Performance Assessment	38

3.7	Experimental Results	41
3.7.1	Results with Approach A	41
3.7.2	Results with Approach B	43
3.8	Conclusions	46
4	Deep Features Selection with PCA and Classification with Random Forest	48
4.1	Feature Selection with PCA	49
4.1.1	Traditional PCA Transform	51
4.1.2	Separable PCA	53
4.1.3	Incremental PCA	55
4.2	Experimental Results with PCA	57
4.2.1	Experimental Results with Separable PCA	58
4.2.2	Experimental Results with Spatial PCA	63
4.2.3	Experimental Results with Incremental PCA in Blocks	64
4.3	Final Results and Conclusions	65
5	An End-to-End Differentiable Anomaly Detection Pipeline	67
5.1	Geometric Alignment	70
5.2	Matthews Correlation Coefficient (MCC)	77
5.2.1	Modifications Necessary to Use MCC as a Loss Function	79
5.3	Dissimilarity Module (DM)	82
5.4	Temporal Consistency Module (TCM)	88
5.5	Differentiable Morphology Module (MM)	89
5.6	Classification Module (CM)	107
5.7	Training, Validating and Testing	107
5.8	Results	114
5.9	Conclusions	119
6	A Comparative Analysis of Object Detection Metrics with an Open-Source Tool	121
6.1	Introduction	122
6.2	An Overview of Selected Works on Object Detection	123
6.3	Bounding Box Formats	126
6.4	Performance Metrics	130
6.4.1	Precision and Recall	132
6.4.2	Average Precision	133
6.4.3	Mean Average Precision	136
6.4.4	Average Recall	136
6.4.5	Mean Average Recall	137

6.5	A Numerical Example	138
6.6	Most Employed Metrics Based on AP and AR	143
6.6.1	AP with IOU Threshold $t = 0.5$	143
6.6.2	mAP with IOU Threshold $t = 0.5$	143
6.6.3	AP@.5 and AP@.75	143
6.6.4	AP@[.5:.05:.95]	144
6.6.5	AP _S , AP _M , and AP _L	144
6.6.6	AR ₁ , AR ₁₀ , and AR ₁₀₀	144
6.6.7	AR _S , AR _M and AR _L	144
6.6.8	F1-Score	145
6.6.9	Other Metrics	145
6.6.10	Comparisons among Metrics	145
6.7	An Open-Source Toolbox	146
6.8	Metrics Evaluation in a Practical Example	147
6.9	Conclusions	149
7	Conclusions	151
7.1	Future Work	152
7.1.1	Input Features	152
7.1.2	Retrain the Network	152
7.1.3	Apply Differentiable Morphology in Other Problems	153
7.1.4	Replace the Geometric Alignment by the Temporal Alignment	153
A	Published and Submitted Papers	154
A.1	Journal Papers	154
A.2	Conference Papers	154
	References	156

List of Figures

1.1	Base pipeline used in this work.	4
2.1	Samples of reference and target frames of the VDAO database [1]. . .	13
2.2	Objects used in videos with multiple objects.	14
2.3	Objects used in videos with single objects.	15
2.4	Main steps of STC-mc method.	16
2.5	Scheme proposed in [2].	19
2.6	Example of ROC curve of a mock classification example.	20
3.1	Pipeline proposed in Chapter 3.	23
3.2	Alignment of signals performed with DTW [3].	24
3.3	Tool developed to help the alignment frames of two videos.	25
3.4	Visual alignment of target frame T_0	26
3.5	Alignment of target frame T_1 considering a search window $W_{a,b}$	26
3.6	Example matching the target frame T_1 with reference frame R_{j-2} . . .	27
3.7	Residual block (adapted from [4]).	30
3.8	Original Resnet-50 architecture.	31
3.9	Modified Resnet-50 architecture proposed by [2].	32
3.10	Scheme representing the selection of the hyperparameters with cross-validation.	39
3.11	Best results obtained by the Bayesian optimization during the cross-validation process for all layers.	44
3.12	DIS values obtained along the iterations during the cross-validation of the 4th layer (residual 3) in approach A.	45
3.13	Results obtained by each layer in the validation and testing sets. . . .	45
4.1	Pipeline proposed in Chapter 4.	48
4.2	Feature tensor represented as a 3-dimensional block with three axes v , h and d	53
4.3	Tubes representing one-dimensional feature vectors in each direction. . .	54
4.4	Feature selection scheme used to select features using PCA.	58

4.5	DIS _{overall} results obtained during training/validation with 3,840 features with the highest energy transformed with the separable PCA.	60
4.6	DIS values obtained during training/validation with features transformed with separable PCA and Bayesian optimization process selecting the amount of principal components.	62
4.7	A 3-dimensional feature tensor represented as 256 2-dimensional feature maps and 256 one-dimensional vectors.	63
4.8	Representation of the feature tensor $V \times H \times D$ into 15 smaller blocks.	65
5.1	Pipeline proposed in Chapter 5.	68
5.2	Examples of output by each module proposed in Chapter 5	69
5.3	Example of reference and target frames temporally aligned and their absolute differences	71
5.4	RMSE between corresponding frames of the reference and target videos aligned with the temporal alignment approach.	71
5.5	Example of reference and target frames temporally aligned and their absolute differences	72
5.6	Example of reference and target frames temporally aligned and their absolute differences	72
5.7	Motion vectors obtained with optical flow between reference and target frames	73
5.8	Process to compute transformation matrices for a given pair of temporally aligned frames R_j and T_i	74
5.9	Observable black borders in a reference frame after rotation of 10 degrees.	74
5.10	Absolute differences of a pair of temporally aligned reference and target frames.	76
5.11	Quantitative results obtained by geometrical alignment with and without CLAHE	76
5.12	Comparative quantitative results obtained by geometrical and temporal alignments	77
5.13	Examples of a ground-truth image and an output image used to compute the MCC.	78
5.14	Comparative metrics: accuracy (ACC), RMSE and MCC.	78
5.15	Examples of images output by the modules DM, TCM, and MM and their respective metrics.	80
5.16	Examples of ground-truth \mathbf{x} and output images \mathbf{y} resulting in undefined MCC values	81
5.17	Scheme representing the operations performed by the DM	83

5.18	Sigmoid functions produced with different values of γ	85
5.19	Scheme representing operations performed in sequence by a function $g(x)$ and a sigmoid function $s(x)$	86
5.20	Example of a pair of aligned frames where no anomalous object is present in the target frame	87
5.21	Examples of pairs of aligned reference and target frames under different illuminations.	87
5.22	Examples of outputs produced by the DM given pairs of aligned frames under the same illumination.	88
5.23	Examples of six consecutive frames produced by DM.	89
5.24	Set operations applied to binary images.	90
5.25	Erosion of the foreground with a structuring element producing . . .	92
5.26	Examples of results obtained with four morphological operations. . .	93
5.27	Example of convolution $g(x) = f(x) * g(x)$	95
5.28	Erosion results obtained with convolution followed by thresholding with different ϵ values.	96
5.29	Example of convolution $g(x) = (1 - f(x)) * h(x)$	98
5.30	Dilation results obtained with convolution followed by thresholding with different ϵ values.	99
5.31	Rotation of the step function.	100
5.32	41×41 circular kernels.	102
5.33	All possible 41×41 circular structuring elements to be learned by the MM.	103
5.34	Illustration of convolutions applied to a binary image.	103
5.35	Approximated opening and closing operations performed by combining convolutions and thresholding operations.	104
5.36	Dilation followed by erosion of a foreground with structuring elements of different radii.	105
5.37	Examples of frames transformed with the differential morphology operations.	106
5.38	Parameters learned by DM during training of fold 5.	111
5.39	Parameters learned by MM during the training of fold 5.	112
5.40	Threshold values learned by CM during training of fold 5.	112
5.41	Losses obtained in all modules during training for each fold.	113
5.42	Comparative output images of our proposed MM TCM method and the second best approach.	118
6.1	Two different types of annotated images from OpenImage [5].	127
6.2	Illustration of the intersection over union (IOU).	131

6.3	Samples of 12 images from the PASCAL VOC 2012 dataset [6]. . . .	139
6.4	Particular cases showing detected bounding boxes overlapping multiple ground truths.	140
6.5	Precision \times Recall points calculated for Table 6.3 and Table 6.4. . . .	141
6.6	Results of different approaches to compute the AP metric.	142
6.7	Class distributions of ground-truth and detected bounding boxes. . .	148

List of Tables

2.1	Popular datasets exploring anomaly detections.	12
2.2	Number of videos and frames for each target object in the VDAO-200 database.	15
2.3	Classification results on VDAO frames applying CNN to extract features reported in [2].	21
2.4	Classification results on VDAO frames achieved by different works.	21
3.2	Dimensions of feature tensors in the format produced by different Resnet-50 layers.	33
3.3	Ranges of values used by the hyperparameter tuning.	35
3.4	Folds containing videos of each class.	37
3.5	Objects used in the videos to train and validate for the fold $F_{\text{white jar}}$	38
3.6	Comparative DIS values obtained in the testing sets with features output by each Resnet-50 layer and reduced with average pooling and max pooling operations.	42
3.7	Results obtained with approach A for all layers with average pooling.	43
3.8	Hyperparameters obtained with approach B for the best layer (residual 3), with features reduced with average pooling.	44
4.1	DIS values for each object class obtained during validation with 3,840 features.	60
4.2	Hyperparameters found by the Bayesian optimization during the training/validation with 3,840 features selected with average pooling and separable PCA.	61
4.3	DIS values obtained in the testing sets with 3,840 features selected with average pooling and separable PCA.	61
4.4	Comparative DIS results obtained with random forest classifiers, trained with features transformed with separable PCA and average pooling.	62
4.5	Testing results with DIS values for two classes of objects considering 3,840 features.	66

5.1	Separation of objects into folds used to train, validate and test the network presented in Chapter 5.	108
5.2	Number of training, validation and testing samples per fold.	108
5.3	Parameters and learning rates used to train the proposed network. . .	109
5.4	Comparative frame-level results of the proposed methods (TCM MM and MM TCM) with other works.	115
5.5	Comparative average DIS and DIS _{overall} frame-level results.	116
5.6	Comparative object-level results of the proposed methods (TCM MM and MM TCM) with other works.	117
5.7	Comparative DIS _{overall} object-level results.	118
6.1	Popular free annotation tools and their supported output formats. . .	127
6.2	Popular object detection methods along with the datasets and metrics used to report their results.	130
6.3	Precision and recall values for detections in Figure 6.3 considering IOU threshold $t = 0.75$	140
6.4	Precision and recall values for detections in Figure 6.3 considering IOU threshold $t = 0.75$	141
6.5	AP results obtained with different interpolation methods and IOU thresholds.	149
6.6	Values of AP and average recall (AR) variations for different object sizes and number of detections per image.	149

List of Symbols

D_h	height of a convolutional filter, p. 30
D_w	width of a convolutional filter, p. 30
F_{obj}	fold where the videos of the object obj is in the testing set, p. 36
H_{out}	height of a feature map output by a convolutional layer, p. 30
P	padding used in a convolutional layer, p. 31
$R_i(x, y)$	pixel at position (x, y) of the i th reference frame, p. 26
R_j	j th reference frame, p. 25
S_h	vertical stride, p. 31
S_w	horizontal stride, p. 31
$T_i(x, y)$	pixel at position (x, y) of the i th target frame, p. 26
T_j	j th target frame, p. 25
W_{out}	width of a feature map output by a convolutional layer, p. 30
$W_{a,b}$	search window limited by frames a and b , p. 25
H	transformation matrix representing a rotation and a translation, p. 73
P	transformation matrix, p. 51
S	covariance matrix, p. 51
$\bar{\mathbf{x}}$	mean vector, p. 51
\mathbf{v}_i	i th eigenvector, p. 51
\mathbf{x}	one-dimensional vector $\in \mathbb{R}^{d \times 1}$, p. 51

λ_i	i th eigenvalue, p. 51
τ	network confidence threshold, p. 133
FN_i	total number of false negative frames in the set i , p. 39
FP_i	total number of false positive frames in the set i , p. 39
TN_i	total number of true negative frames in the set i , p. 39
TP_i	total number of true positive frames in the set i , p. 39
b_i	i th learnable bias, p. 83
w_i	i th learnable weight, p. 83
w_{r_i}	learnable weight applied in the i th reference feature map, p. 83
w_{t_i}	learnable weight applied in the i th target feature map, p. 83

List of Abbreviations

AP	average precision, p. 123
CM	classification module, p. 68
CNN	convolutional neural network, p. 3
COCO	common objects in context, p. 126
DAOMC	detection of abandoned objects with a moving camera, p. 18
DM	dissimilarity module, p. 67
DTW	dynamic time warping, p. 24
FC	fully-connected, p. 19
FN	false negative, p. 18
FPR	false positive rate, p. 19
FP	false positive, p. 18
IOU	intersection over union, p. 129
MCBS	moving-camera background subtraction, p. 18
MCC	Matthews correlation coefficient, p. 4
MM	mathematical morphology module, p. 68
PTZ	pan-tilt-zoom, p. 8
RF	random forest, p. 3
RMSD	root mean squared difference, p. 70
RMSE	root mean squared error, p. 70
ROC	receiver operating characteristic, p. 20

Resnet	residual network, p. 18
RoSuRe	robust subspace recovery, p. 17
SIFT	scale invariant feature transform, p. 124
STC-mc	spatio-temporal composition with moving camera, p. 16
STC	spatio-temporal composition, p. 15
SURF	speed up robust features, p. 124
SVM	support vector machines, p. 9
TCM	temporal consistency module, p. 68
TN	true negative, p. 39
TPR	true positive rate, p. 19
TP	true positive, p. 19
VDAO	video database of abandoned objects, p. 2
mAP	mean average precision, p. 123
mcDTSR	moving-camera domain-transformation sparse representation, p. 18
mcRoSuRe	moving camera robust subspace recovery, p. 17

Chapter 1

Introduction

In the last decade, the advances of Computer Vision methods have impacted applications in different areas: health care [7], entertainment [8–11], security [12, 13], transportation [14–17], medical diagnosis [18–20], manufacturing [21, 22], law [23], etc. Activities that could not be entirely performed by automated systems have become feasible with the evolution of the algorithms and hardware. Computer-aided diagnosis systems, for example, are now feasible due to the accuracy of image classification algorithms. Self-driven cars also exist due to improvements of object detection methods, so different classes of objects could be precisely identified.

Similarly, the need for robust and precise methods to detect undesired objects is also found in other critical applications including waste sorting, road inspection, surveillance of controlled areas such as shopping centers, stadiums, airports, industry and train stations [24]. Current object detection methods are object-addressed, which means only previously known classes of undesired objects can be identified [25–27]. Situations where the class of the object to be found is unknown prevent the application of the popular and classical object detection approaches. In those cases, the anomaly detection approach is adopted, in which a comparative analysis is performed to find objects which were not present in a predefined reference condition.

In industrial facilities, as offshore platforms, materials left in areas that may cause accidents, or items capable of producing flames are critical anomalies that could compromise the safety of the industrial plant and workers. Human visual inspection in such areas may be problematic due to the limitations of access in such hazardous environments, demanding more autonomous solutions. A typical solution to cover large areas remotely is the usage of surveillance systems with multiple cameras to provide visual coverage of the whole environment [28, 29]. An alternative to cover wide zones with reduced costs is the usage of moving cameras to monitor such large areas [28, 30]. Even though this approach may reduce costs on the acquisition of the images and videos, the process of detecting changes along the frames becomes even more complex. Problems caused by the movement of the

camera like jitters, illumination changes, occlusions, and shadows limit the success of techniques applied with static cameras [31, 32]. Such constraints hinder the comparison between target and reference videos affecting the quality of the results.

In this thesis, the anomaly detection problem using labeled training videos is investigated. By that, the anomaly detection is addressed by comparing a newly acquired video, known as the target video, to a reference video considered free of anomalies. In this way, an abandoned object, which can be associated to a video anomaly, is detected as the target and reference videos diverge considerably. For that purpose we will use the VDAO database (Video Database of Abandoned Objects) [33] and our results are compared to previous works that also reported results on this same database.

1.1 Objectives

This thesis covers the problem of anomaly detection in videos acquired with a moving camera in a cluttered industrial environment. The video database of abandoned objects in a cluttered industrial environment (VDAO) used in this work contains two groups of videos: reference and target videos. The former contains videos captured in the environment without the presence of certain objects, while in the latter group of videos, objects are inserted in the scene. Thus, in this work the term ‘anomaly detection’ is seen by the perspective of detecting changes in the target videos represented by objects added in the scene, characterizing the anomaly. As the VDAO database contains videos acquired in a real industrial scenario, the critical anomalies such as fire and smoke are represented by such abandoned objects of different classes.

The videos of the VDAO database were recorded in an industrial environment, with pipes, tubes, valves, chains and metal structures. Due to the movement of the robot and different light conditions, the reflection of light in such metal parts causes an extra disturb while comparing the same scene in different videos. The fact that the camera is moving and the structures in the scene are in different depths, the parallax effect makes the background and foreground appear to move at different speeds.

The problem this thesis proposes to solve is hampered due to the constraints presented in the VDAO database. Such difficulties are not only related to the characteristics of the videos such as camera jitters, occlusions and illumination changes, but also due to the fact that the testing database is made of chunks of all videos. As all frames in the testing sets are also seen in the training set, a careful process of division of training, validation and testing sets had to be done. So far, other works have been developed to solve the anomaly detection on the same database, but this

is the first thesis applying Deep Neural Networks.

Given a target video, the system proposed by this thesis aims to identify the presence of an anomalous object in each frame. The classes of such objects are unknown beforehand and are not considered by training a model of any type. For that, the target video is first aligned with the reference video. Then, deep features from frames of both videos (reference and target) are obtained with convolutional neural networks (CNNs) in a siamese-like structure. Different techniques were explored in this thesis. First, a random forest (RF) classifier was used to classify both reference and target features followed by a hyperparameter tuning with Bayesian optimization. In a comparative approach, a feature selection process was applied with principal component analysis (PCA) to reduce the space of features before it is classified by the random forest algorithm. By that, our results are equivalent to those obtained by previous works on the VDAO database. At last, we propose an end-to-end pipeline containing a module to enhance the differences between the target and reference frames, a differential morphology module to eliminate false positive regions of the target frame, and a classification layer responsible to predict if the given target frame contains any anomaly or not.

1.2 Contributions

The contributions of this thesis are mainly, but not limited to, are given by the following points:

- The benchmark videos of the VDAO database used for testing are available, but without labels. During this thesis, the testing videos were labeled, contributing to future works using the VDAO testing dataset.
- Differently from previous works using the VDAO database, the supervised approach used in this work requires a careful split of the data. Thus, a division of the videos into training, validation, and testing sets and different folds is proposed to avoid data contamination. This type of dataset separation has not been done in other works using VDAO and it has proved to be a good strategy to avoid having biased results.
- A real-time temporal alignment of the reference and target videos is proposed.
- A comparative approach using PCA to select deep features (features produced by a deep neural network) for the binary classification problem with random forest.
- A novel differential morphological operation module is proposed allowing the application of morphological operations in a differentiable way. By that, the

radii of structuring elements can be automatically adjusted during the learning process. This novel approach can be expanded to other tasks involving morphological operations.

- The Matthews correlation coefficient (MCC) was used as a metric to compute the pixel-wise assertiveness of modules of our proposed approach. Due to its limitation in some cases, we propose modifications in this method, expanding its application to critical cases.
- A full chapter is dedicated to review metrics used to evaluate object detection. This chapter was placed as the last chapter, and can be read independently from the other chapters. This review conducted the development of an open-source tool implementing different object detection metrics and compatible with different bounding box formats.

The next section presents the structure used in this thesis.

1.3 This Work

This thesis aims to provide a solution for the anomaly detection problem. For that, a database containing videos captured with a moving camera was used to train, validate and test the proposed models. The chapters and sections are organized to present the approaches in an incremental way. Thus, each chapter presents a solution following the structure of a common pipeline presented in Figure 1.1.

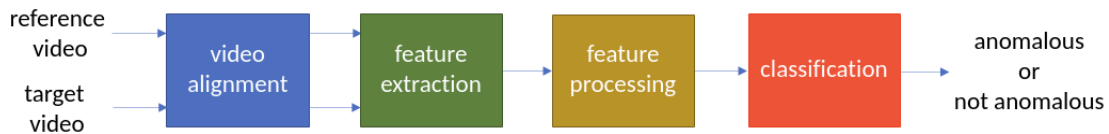


Figure 1.1: Base pipeline used in this work. Given a reference and target videos, the goal is to classify each frame of the target video as containing an anomaly (anomalous) or not (not anomalous).

The input of the proposed method is a pair of videos. The reference video represents the expected conditions, without anomalies. During the algorithm training phase, the anomaly is presented in the target video, and during the testing (or inference phase), the anomaly may or may not be presented in the target video. As both videos are not temporally synchronized, the first step, represented by the *video alignment* block, aims to align the frames of both reference and target videos. Then, *feature extraction* techniques will be explored to find a high-level representation of the aligned frames. All manipulation operations made in the features, such as feature reduction, feature selection, temporal consistency, morphology, etc. are

represented in the *feature processing* block. The last step, the *classification* block, outputs the class of each target frame, being called ‘anomalous’ if an anomaly is presented or ‘not anomalous’ otherwise. The anomaly is represented by an object added in the target frame which is not in the reference frame.

Along with the chapters, modifications in the pipeline are made and relevant works in the literature are reviewed accordingly. The reader is informed of the attempts that did not work as expected, decisions taken along the development of each phase, results compared with previous works in the same database, and conclusions. This work is divided as follows:

Chapter 2 defines the problem of anomaly detection, reviews related works and datasets.

In Chapter 3, a temporal alignment approach is proposed, a deep neural network is used to extract features from the frames, and a random forest classifier produces the frame classification. A Bayesian optimization technique is also applied to tune some hyperparameters of the classifier. The split of the database into folds containing training, validation, and testing is also covered in this chapter. In the end, the results of this approach and conclusions are also presented.

In Chapter 4, a feature selection module is added into the pipeline so that deep features have their dimensions reduced with principal component analysis (PCA). Different approaches to compute PCAs are applied and their results are compared and commented on.

Chapter 5 includes new modules into the pipeline. A geometrical alignment is proposed to improve the quality of the registration between the reference and target frames. Before the classification module, a dissimilarity module fuses both reference and target frames producing a binary image highlighting the region where the anomaly is located. A temporal consistency module removes false positive pixels in consecutive frames and, before the classification, a novel differentiable morphology technique is proposed to eliminate the remaining false positive pixels within the frame level.

Chapter 6 is an independent chapter which is not directly connected to the previous chapters. In that chapter, a deep review on the subject of object detection metrics is provided. The reader may sense a clear disconnection from the main idea of this thesis, but due to the importance and impact this chapter brought to the scientific community, the authors found it relevant enough to be part of this thesis and, therefore, was placed as the last chapter before the final conclusions.

Conclusions and main ideas for future works are presented in Chapter 7.

Chapter 2

Anomaly Detection in Cluttered Environments

This chapter defines the concept of anomaly detection and reviews the main works and datasets used in this task. We start by defining the limits between methods of classical object detection and abandoned anomalous objects in Section 2.1 . Then, a review of works involving object anomaly detection is presented in Section 2.2. Next, available datasets covering the task of abandoned objects are analyzed in Section 2.3, including the VDAO, the best suited dataset for the purpose of this work. Works using the VDAO database are reviewed and their results are shown in Section 2.4. Finally, conclusions are presented in Section 2.5.

2.1 Anomaly Detection

The word *anomaly* comes from the Greek word *anomalos*, meaning *unevenness* or *irregularity* [34]. Depending on the scenario and circumstances, the representation of anomalies may vary. In general, anomaly is a non-conforming pattern often referred as outlier, exception, surprise, aberration, peculiarity, or contaminants [35].

Anomaly detection systems tackle the problem of finding patterns in data that do not comply with expected behaviors [35]. Within the computer vision context, in some applications objects placed or removed from a scene or unsought actions can be considered anomalies. Nevertheless, such anomalous cases depend on a certain context and are part of the system specifications. For instance, pedestrians walking on the sidewalk is not considered an uncommon situation, but it can be considered anomalous (and dangerous) by a system if pedestrians are detected crossing a busy highway [14]. Likewise, certain objects like guns and abandoned luggage can be considered anomalies in public crowded places [12, 36, 37]. In industrial facilities, as offshore platforms, materials left in areas that may cause accidents or items capable

of producing flames are critical anomalies that could lead to severe consequences if not detected in time [33]. There is a current need for surveillance systems designed to monitor certain areas and raise alerts if predefined anomalous situations or objects are observed.

The anomaly detection task in the context of this work is seen as detecting changes in a scene given two videos. Non structural changes such as in illuminations, small differences in sizes and positions of the objects caused by the movement of the robot, must be ignored by the system. Only objects that are in the scene in a particular video, which were not presented in a reference video must be detected. That is why the reader must be aware of the fact that anomaly is here understood as a representative change in the scene given the inclusion of a previous non existing object.

In view of critical tasks to be identified by surveillance systems, anomalous objects left in a scene can be considered very critical, specially if they can threaten personal safety or put operations in risk [36]. Detecting undesired objects is a need in many applications such as waste sorting, road inspection, supervision of controlled areas such as shopping centers, stadiums, airports, train stations and so on [24].

Supervised object detection methods are object-addressed, where only previously known classes of objects can be identified [25–27, 38–41]. Regarding surveillance systems, depending on the application, there is an enormous amount of object classes that could be classified as anomalies, making popular object detectors useless in such applications [13].

To overcome such problem, some anomaly detection systems are designed to compare a given input video to another reference video, which represents normal environmental conditions [42]. As an anomaly usually involves rare activities, designing a dataset with uncommon situations is very challenging. In addition, capturing undesired anomalies such as fire, flood, and smoke in real scenarios may represent risks. Each dataset characterizes anomalies in a different perspective. The VDAO dataset used in this thesis represents anomalies as abandoned objects. But before covering the datasets, works proposing solutions for the anomaly detection task are presented in the next section.

2.2 Review of Anomaly Detection Works

Given an input video, the goal of anomaly detection algorithms is to precisely recognize the moment or the position of an anomalous action or object. In this task, the videos are recorded with surveillance cameras, whose position and type determine the approach employed by different works. With respect to video capture, surveillance systems can be classified into two types: systems using static cameras and

systems using moving cameras. The first category includes non-moving cameras, which can be static cameras or PTZ (Pan-Tilt-Zoom) cameras. Static cameras do not rotate and are limited to a much narrower view of the scene. PTZ cameras allow zooming and rotations, expanding the view of the region. The second category, systems using moving cameras, provides a wider visual coverage of the environment by applying a reduced number of cameras.

In videos with static cameras, background subtraction has been widely applied to detect anomalies [43–45]. In [46], the authors consider frequent local motions in the background such as waving trees and grass and apply a visual attention mechanism to infer a complete background. A framework was proposed to detect abandoned objects and identify their owners in [47]. Some works apply static cameras to detect moving objects, but allow a small amount of movement usually due to some kind of camera jitters [48–50].

On the other hand, in a more challenging scenario, works such as [28, 30, 51] explore the usage of moving cameras to supervise large areas. Even though moving cameras may cover larger areas and reduce costs, detecting changes along the frames becomes a more complex task. The camera movements and speed may vary along the route and, as the camera is moving, shadows, occlusions, and light conditions are frequently changing, which limits the success of techniques used by static cameras [31, 32, 52]. The work [28] was able to detect abandoned objects along a road by mounting a camera on a moving car. By that, the reference video without abandoned objects was recorded along the path. Then, target videos recorded along the same path were processed to find such objects. Global positioning system (GPS) coordinates were used to align the reference and target frames and a homography was used to estimate the affine transformations between the corresponding frames. In the end, the frames are compared with a normalized cross-correlation image. Similarly, [30] performed the detection of anomalies in train tracks by recording videos with a camera mounted on a train. After aligning the frames, different similarity measurements were applied to detect the presence of anomalies in the scenes. In both works, a reference video with no suspicious object in the scene is compared to a target video, with possible anomalous objects.

Considering that anomalous events are uncommon situations, ordinary (non-anomalous) events in videos are more likely to be reconstructible based on training data. Thereby, a common approach is to build a model using only non-anomalous events from the training set of videos. Sparse linear combinations of similar patterns are considered to be a good representation of the expected non-anomalous activities with a low reconstruction error. In contrast, as the anomalous events are not presented in the training data, these events will produce high reconstruction errors [53]. The reconstruction of frames using previously learned coefficients is the essence of

approaches using sparse coding to detect anomalies in videos. Constrained by a dictionary whose columns are basis vectors for reconstructing non-anomalous events, the work [54] applies a spatio-temporal sliding window to scan videos. Salient points detected in regions of successive frames determine cuboids of events. By projecting a cuboid onto the set of the sparse coding basis vectors, a reconstruction vector is obtained and its reconstruction error is measured, thus determining if the input event is anomalous or not. In this approach, the basis vector dictionaries are updated in a non-supervised way, thus it does not require prior assumptions of the anomalous events.

The work in [55] affirms that the optimization of sparse coefficients is extremely time consuming, becoming the bottleneck of dictionary learning based approaches. An architecture similar to stacked recurrent neural networks (sRNN) is then proposed to update the sparse coefficients reducing computational costs. Instead of using a predetermined similarity measurement of consecutive frames, a data-dependent similarity measurement is used. In this work, multiple datasets with static cameras were used.

Surveillance videos were used to monitor abnormal activities of crowds with a static camera in [56]. A feature space is represented by a histogram built with motion vectors from background subtraction of non-anomalous frames. A dictionary with low rank coefficients is learned during the training phase by optimizing the reconstruction error. In the testing stage, the error of the reconstructed coefficient vectors is expected to be low for non-anomalous samples and, therefore, determines the presence of anomalous activities.

Many works take the advantage of deep learning-based methods to model the spatio-temporal relation in videos to detect anomalies. In [57], a spatio-temporal CNN was trained to extract crowd motion patterns and classify anomalous and non-anomalous events. Also to identify unusual crowd activities, a deep Gaussian mixture model was used in [58] to represent normal activities in videos captured with static cameras. The work [59] combines a U-Net with a long short-term memory (LSTM) network to represent spatial and temporal information to identify abnormal events in videos with static cameras. A residual spatio-temporal autoencoder was trained in [60] using non-anomalous videos. As normal frames are reconstructed with lower errors than the anomalous frames, a threshold is set to identify the anomalies.

In [61], the so popular generative adversarial networks (GANs) were used to detect anomalies in surveillance videos. The work in [62] applies three denoising autoencoders to learn individual and mixed motion features from videos. The appearance, motion and joint representations of sequential frames are individually classified by one-class support vector machines (SVMs). The results are fused by a single decision module that determines the presence of the anomalies.

As anomalies are usually sporadic and specific events, most databases used by the aforementioned works were specially designed with this purpose. The realism and spontaneity of some actions do not always correspond to the reality, specially when actions are performed by individuals. The realistic scenarios must also be faithfully represented in these videos, so that real surveillance systems can benefit from models trained on other datasets. It is also not hard to find databases which represent anomalous situations in a very realistic way, encompassing constraints found in real conditions. Nevertheless, in their majority, reference videos to represent expected normal conditions are not included in the databases. Thus, next section is dedicated to overview different databases used as benchmark by most anomaly detection systems.

2.3 Available Databases

As previously stated, the definition of anomaly varies according to the target problem. For a system to be able to identify correctly in which frame or position the anomaly is, it is necessary to use a video database representing such events in an accurate and realistic way. Public available datasets to attend the problem of anomaly detection are created for specific purposes with limited classes of anomalies and scenarios. The most popular ones are described below:

- The UCF-Crime [63] consists of videos captured by still cameras representing 13 anomalies such as abuse, fighting, robbery, burglary, etc.
- The UMN: Detection of Unusual Crowd Activity dataset [64] presents videos of people walking around and the anomaly is characterized by only running action.
- The UCSD: Pedestrian Anomaly Dataset [65] gathers surveillance videos recorded in a single place. They characterize anomalies as activities performed in determined sites, e.g. a group of people riding bicycles in the walkways.
- The Avenue dataset [66] has short videos with some unrealistic anomalies, such as paper throwing, loitering, and running.
- The Subway dataset [29] consists of cameras positioned in the entrance and exit of a subway station. The anomalies are labeled as people walking in wrong directions, loitering near the exit, and skipping payment.
- The work [67] presents the CDNET, a popular database designed for motion and change detection in the background. It contains 53 video sequences, split into 11 categories, covering various movements of the background with a static

camera. Until now, none of the mentioned databases represents the anomalies as abandoned objects.

Among datasets to represent abandoned/lost objects, we can mention [68], which contains videos recorded with four different static cameras capturing left-luggage of different types such as briefcase, suitcase, backpack, etc. The i-LIDS dataset [69] covers 6 different real world footage including abandoned baggage detection and doorway surveillance. These videos were also recorded with static CCTV cameras and not all videos are labeled. Table 2.1 summarizes some of the most widely used datasets for the task of anomaly detection.

So far, to the best of our knowledge, the only dataset that addresses the anomaly detection problem as abandoned objects with a moving camera in industrial environments and provides reference videos free of anomalies is the video database of abandoned objects, here simply referred to as VDAO. The VDAO database is described in details in the next subsection.

2.3.1 The VDAO Database

Our proposal is within the context of monitoring offshore platforms, aiming to alert threatening objects that may cause risk to the safety of workers, operations, and wildlife presented in such areas. The VDAO database [33] fits perfectly in the context of anomaly detection with moving cameras in an industrial environment. After a strong research on other datasets used to detect anomalies, we found that the VDAO is the best suited one for this kind of problem. As the videos in the VDAO database were recorded in a real industrial plant of an oil and gas company, the scenes and structures seen in the videos are real and very common in this kind of environment. The main reasons that lead us to choose the VDAO among the other datasets are:

- The scenes are real industrial environments, containing pipes, valves, metallic objects and structures found in oil and gas industries.
- Anomalies are represented as objects of different classes placed in the scene. Most of the other datasets represent anomalies as actions.
- Videos are recorded under different illuminations, which not only makes this dataset more realistic but also increases the difficulty to detect the objects.
- The camera is moving, being an alternative to cover large areas. Most of other datasets use static cameras.

The VDAO is one of the most challenging databases covering anomaly detection in large areas. The VDAO, presented in [33] and available at [1], is a collection

Table 2.1: Popular datasets exploring anomaly detections.

Dataset	Number of videos	Number of anomalies (unusual events)	Examples of anomalies	Types of Camera(s)
UCF-Crime [63]	1900	13	Abuse, Arrest, Assault, Accident, Vandalism, etc	stationary
UMN [64]	1 with 11 different shots	1	People running	stationary
UCSD [65]	98	5	non-pedestrian entities in the walkways (bikers, skaters, etc), anomalous pedestrian motion patterns	stationary
Avenue [66]	37	14	people running, throwing objects, loitering, etc	stationary
The Subway (entrance and exit) [29]	2	19	people walking in wrong directions, no payment, loitering, etc	stationary
PETS 2006 [68]	not informed	1	luggages left unattended	stationary
i-LIDS [69]	not informed	6	abandoned baggage, parked vehicle, people, etc	stationary
CDNet [67]	31	does not apply	does not apply	stationary, PTZ and thermal cameras)
VDAO [1]	77	1	abandoned objects	moving camera

of videos recorded in an industrial environment surrounded by pipes, valves, and illumination changes. The 77 videos found in the VDAO database were recorded by a camera mounted on an *iRobot Roomba*, going through a 6-meter path in a back-and-forth movement on a predefined trajectory. Each video has a duration of approximately 8 minutes. Two different cameras, Axis P1346 and Dlink DCS-3717, were used to record the videos, both with resolution 1280 x 720 pixels and 24 frames per second. A total of 24 objects were placed in the scenario, simulating objects that do not belong to the environment in normal expected conditions. Two groups of videos were obtained: *reference* and *target* videos. Reference videos are those in which abandoned objects are not present and target videos contain one or more abandoned objects, which could be partially occluded as shown in Figure 2.1. Among the target videos, 15 distinct objects were added in the scene, resulting in a video with multiple objects (Figure 2.2). The other 9 objects were placed in

the scenario producing videos with a single object (Figure 2.3). Each video with multiple objects is approximately 18-minute long, and videos containing a single object have approximately the duration of 6 minutes. Available annotation files contain bounding boxes representing the position of the objects in every frame. The partial occlusions of the objects and the variation of illumination in VDAO database hamper the task of object detection. Shadows of the objects, reflections of light on metal parts of the scenario, camera vibrations, and speed difference of the robot during the videos acquisition are some complications that prevent the perfect alignment of the reference and target frames. The videos are organized into 10 tables, each indicating the illumination type used during the acquisition and if the target videos contain a single or multiple objects.

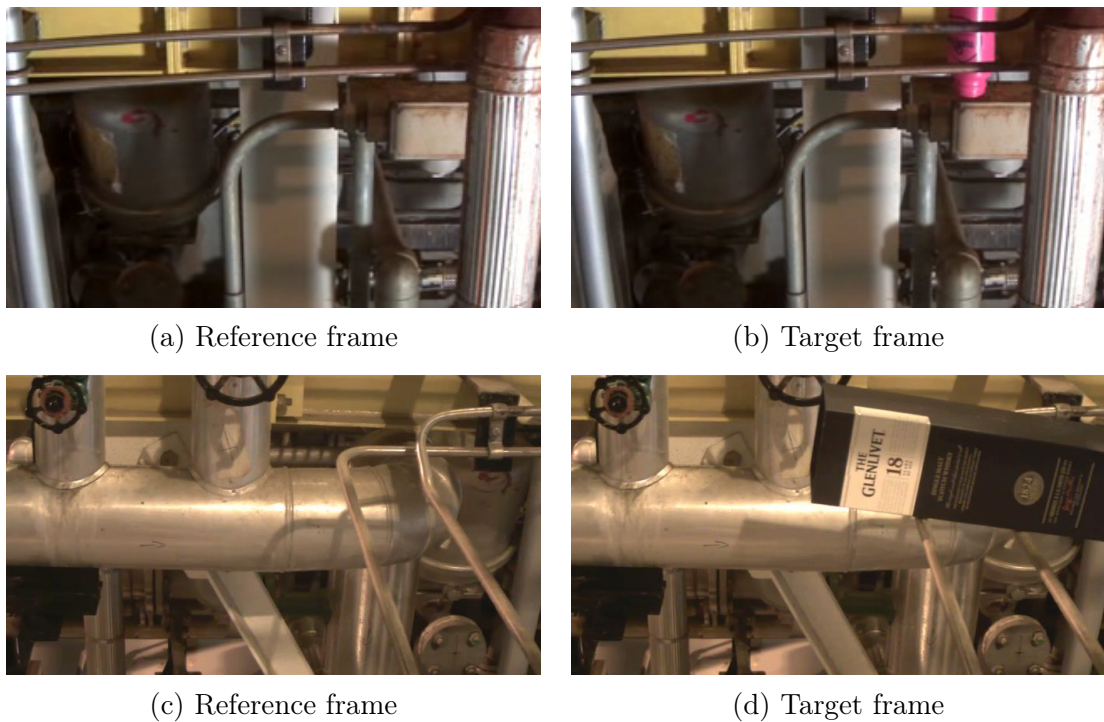


Figure 2.1: Samples of reference and target frames of the VDAO database [1]. Reference frames (a) its their corresponding target frames (b) with an abandoned object: a pink bottle on the top right corner; (c) Reference frame and its corresponding target frame (d) with an abandoned object: green box on the right side of the image.

VDAO-200: The Testing Database

The first works developed with the VDAO database used a 200-frame long subset for benchmark [70–72]. This auxiliary testing database is available at [73], and contains 59 single-object videos with 9 different objects in different positions under 2 illumination conditions. The classes of objects included in this set are: *black*

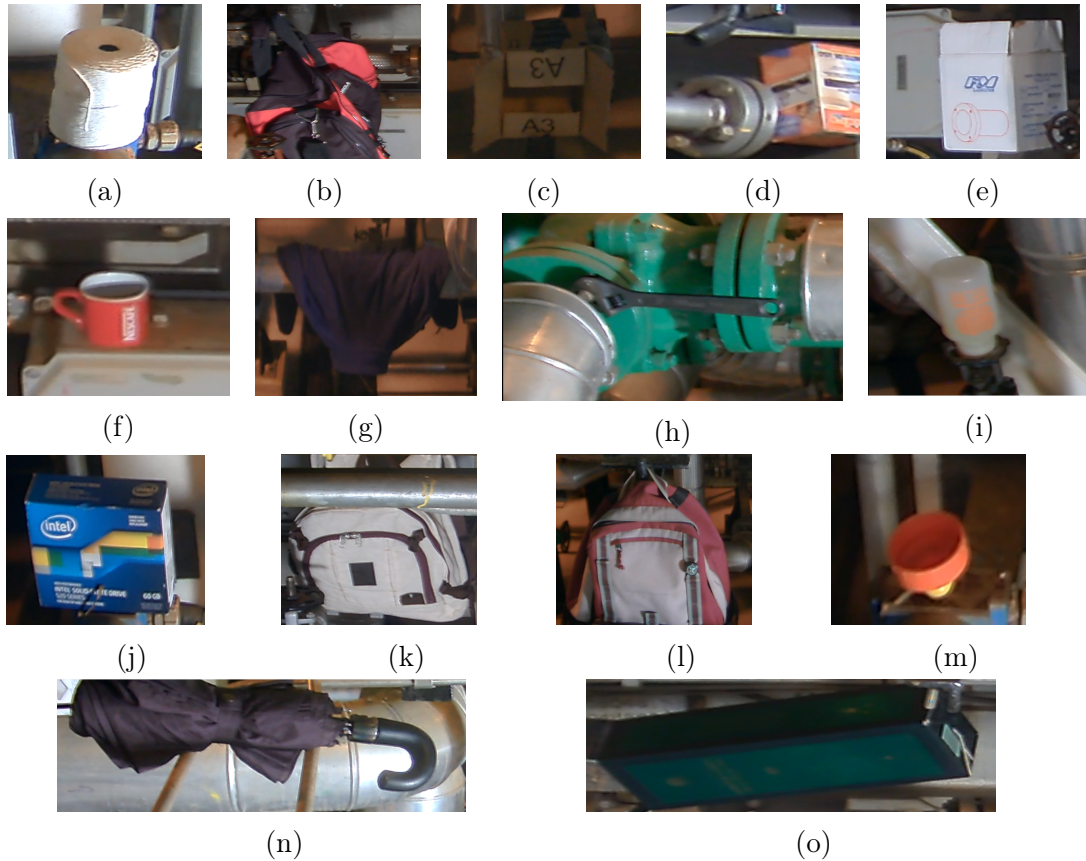


Figure 2.2: Objects used in videos with multiple objects (the scales of the images were changed for a better representation): (a) string roll; (b) bag; (c) white box; (d) lamp bulb box; (e) spotlight box; (f) mug; (g) blue coat; (h) wrench; (i) bottle; (j) blue box; (k) backpack; (l) pink backpack; (m) bottle cap; (n) umbrella; (o) green box.

backpack, black coat, brown box, camera box, dark-blue box, pink bottle, shoe, towel, white jar. Table 2.2 presents the number of videos for each object class.

All 59 videos of the testing set are short-duration patches of the VDAO videos containing single objects. Among all 60 single-object videos of the original VDAO, only 1 is not present in the VDAO-200 database. This makes a fair split of videos between training and testing sets without intersection a difficult task. VDAO is not only a challenging dataset due to the constraints found in the videos (jitters, illumination, shadows, etc.), but also because the division of the videos into training, validation and testing sets without having the same video in more than one set is also challenging. A solution to this problem was used by [2], where their training sets are split into folds, so the frames in the testing set were never seen during the training. Section 3.5 is dedicated to cover the approach used by our work as an attempt to overcome such problem.

In the next section, the works that applied different techniques in the VDAO database are discussed.

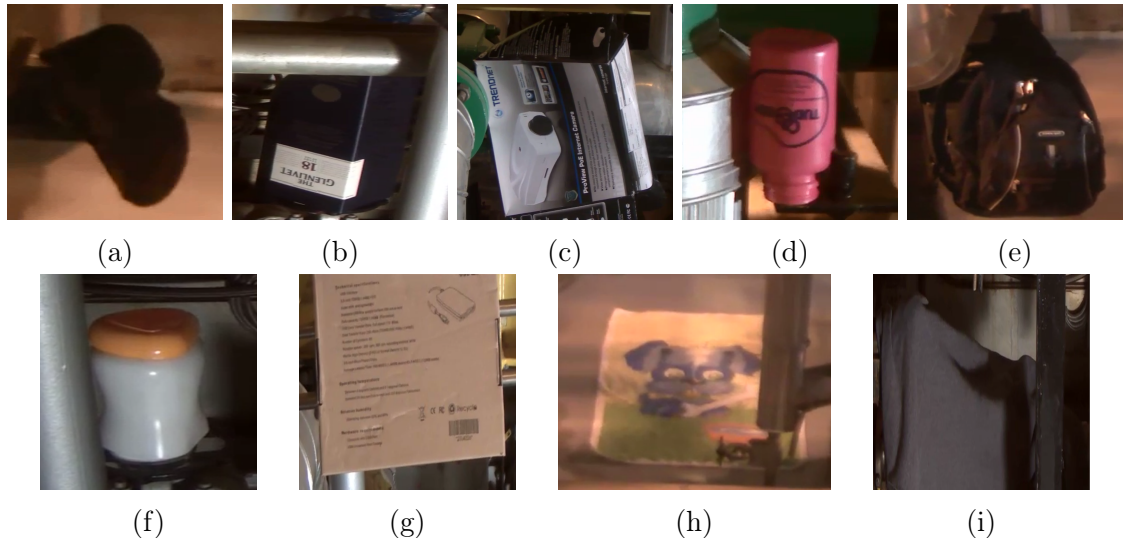


Figure 2.3: Objects used in videos with single objects (the scales of the images were changed for a better representation): (a) shoe; (b) dark-blue box; (c) camera box; (d) pink bottle; (e) black backpack; (f) white jar; (g) brown box; (h) towel; (i) black coat.

Table 2.2: Number of videos and frames for each target object in the VDAO-200 database.

objects	videos	frames
dark-blue box	6	1200
black backpack	10	2000
black coat	6	1200
brown box	6	1200
camera box	6	1200
pink bottle	7	1400
shoe	6	1200
towel	6	1200
white jar	6	1200

2.4 Anomaly Detection with VDAO Database

Concerning the anomaly detection in the VDAO database, different works have attempted to identify anomalies within the frame level, being able to classify frames as anomalous or non-anomalous. Next subsections are dedicated to describe the three types of approaches applied in the VDAO considering anomaly detection within the frame level. In the end, their results are compared.

2.4.1 Using Spatio-Temporal Codebooks

Traditional spatio-temporal composition (STC) [74] methods are used to break videos into small 3D volumes and to calculate the probability of spatio-temporal patterns. However, if applied to videos recorded with a moving camera, a lot of false detections may occur. The work in [70] proposes a modified STC algorithm to

overcome such problem and applies its solution on the VDAO and UCSD databases. In their work the authors suggest a two-step dictionary to improve the anomaly detections. Their main contribution is a spatio-temporal feature extraction process obtained by filtering the video sequence.

The modified STC method, namely STC-mc (STC-moving camera) breaks down the videos into small volumes that are represented by codewords from a codebook. Probabilities of occurrence of spatio-temporal compositions of the codewords are then calculated. The compositions with the lowest probabilities are candidates to be anomalous. Three parameters were defined based on experimental tests: standard deviation σ for the Gaussian temporal smoothing filter; weight of the time derivative λ ; and the threshold limiting the maximum distance ϵ_1 to consider two different codewords. Figure 2.4 represents this workflow.

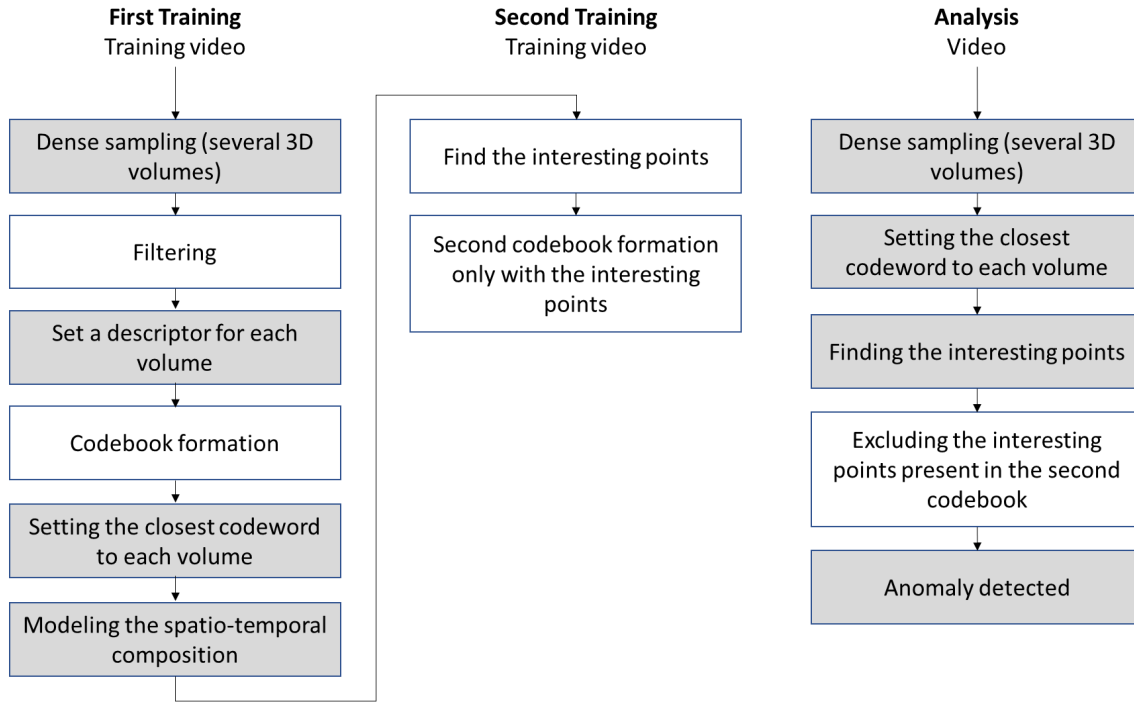


Figure 2.4: Main steps of STC-mc method. The blocks in gray are the original steps of the STC method. The white blocks represent their contributions. (Image was adapted from [70]).

In their initial step, the frame sequences are split into 3D volumes. A descriptor for each volume is obtained by applying a time derivative of each pixel in the volume. This descriptor is sensitive to the camera movement, so a Gaussian filter with kernel size 5 is used to temporally smooth the derivatives variations. The standard deviation σ of the filter is a hyperparameter found experimentally. The descriptor for each volume is a vector of dimension 1×245 . To reduce the redundancy present in the spatio-temporal volumes, similar volumes are clustered and, for each group, a codeword is generated to represent these volumes. The set of codewords forms the

codebook. Based on the Euclidean distance, each volume is related to a codeword with a weight. The probability of the volumes within a larger region is calculated. By applying a threshold on the probability, it is possible to measure how likely an anomaly is in each volume. For the VDAO database, the threshold was not enough to detect the anomalous events, so an additional codebook was introduced.

2.4.2 Sparse-Representation of the VDAO

Inspired by the low-rank subspace decomposition, named Robust Subspace Recovery (RoSuRe), [72] represents a video by a data matrix given by:

$$X = LW + E, \quad (2.1)$$

where matrix L represents the union of subspaces and each column L_j represents a sampling matrix of a subspace S_j , matrix W is a sparse block-diagonal matrix, so that $LW = L$ with $W_{ii} = 0$, and matrix E is a matrix of perturbations. By assuming that W and E are sparse, Equation 2.1 can be solved by the following minimization problem:

$$\min_{W,E} \|W\|_1 + \lambda \|E\|_1 \quad \text{s.t.} \begin{cases} X = L + E \\ LW = L \\ W_{ii} = 0 \end{cases}. \quad (2.2)$$

Given a reference video, its data matrix is represented by $X_r = L_r W_r + E_r$, whose columns represent a frame of the reference video. The low-rank linear part of the reference video is L_r and E_r is the sparse error matrix. The data matrix representation of a target video is $X_t = L_t W_t + E_t$. The authors consider the low-rank part of the target video (L_t) equals to the low-rank part of the reference video (L_r), so that $X_t = L_r W_t + E_t$. The sparse error matrix E_t is said to contain the anomalies. Therefore, Equation 2.1 is then modified to a new version namely mcRoSuRe (moving camera Robust Subspace Recovery) given by

$$\min_{W,E} \|W\|_1 + \lambda \|E\|_1 \quad \text{s.t.} L_r W = X - E. \quad (2.3)$$

In the end, this procedure results into Equation 2.4, whose sparse matrix E , representing the remaining error, is expected to concentrate just the abandoned objects in X_t that are not present in X_r , that is

$$E_t = E_r W + E. \quad (2.4)$$

Another work that explores low-rank representation of target videos in the VDAO

database is [71]. The authors improved the mcRoSuRe developing an accelerated version of the algorithm, referred to as mcRoSuRe-A. The main idea to reduce the computational cost is to replace Xr by a much smaller matrix X'_r . Besides the works covered in this section, the results of the mcRoSuRe-A are compared to other two works: detection of abandoned objects with a moving camera (DAOMC) [28] and the moving-camera background subtraction (MCBS) [30]. Also, a detailed computational analysis shows that the mcRoSuRe-A runs 100 times faster than the original mcRoSuRe.

More recently, the mcRoSuRe algorithm was modified to a newer version called moving-camera domain-transformation sparse representation (mcDTSR), with great improvements of its results [52]. In this work, the transformed-domain optimization problem was turned into a two-stage iterative process. First, an inner loop estimates the best geometric transformation between the frames, then, the best matrix factorization is estimated. We revisit all these works and compare their results in Subsection 2.4.4.

2.4.3 Extracting Features from VDAO Videos with CNNs

The first work to explore deep convolutional neural networks (CNNs) in the VDAO is [2]. Initially, reference and target frames were temporally aligned using DTW (dynamic time warping) [3, 75], a popular algorithm to measure similarities between two temporal sequences. Then, convolutional layers of the residual network-50 (Resnet-50) [4] were used to extract features of both reference and target frames. Layers of 17 depths of the network were chosen to extract the features. An average pooling layer was inserted at the end of every layer to reduce the number of parameters. For each layer $L = \{1, 2, 3, \dots, 17\}$, this process resulted in n pairs of reference and target feature maps. Each feature map obtained from the target frame was subtracted from the feature map obtained from its corresponding reference frame, resulting in a single tensor of features. In the end, a random forest classifier [76] was used to classify each each frame as anomalous (containing an abandoned object) and not anomalous (without an abandoned object). The goal of their work was to identify which convolutional layers obtain the best discriminant features to be fed to a random forest classifier. Figure 2.5 shows the scheme proposed in [2]. Note the average pooling used to reduce the number of features is not presented in this scheme.

After predicting the class of every frame, the authors use a majority voting window to minimize incorrect predictions. Along the sequence of frames, some were misclassified either as false positive (FP) or False Negative (FN) and, thus, the predictions of neighboring frames were used to improve the results. For example, to correct the classification of the i^{th} frame (F_i), the prevailing classification of frames

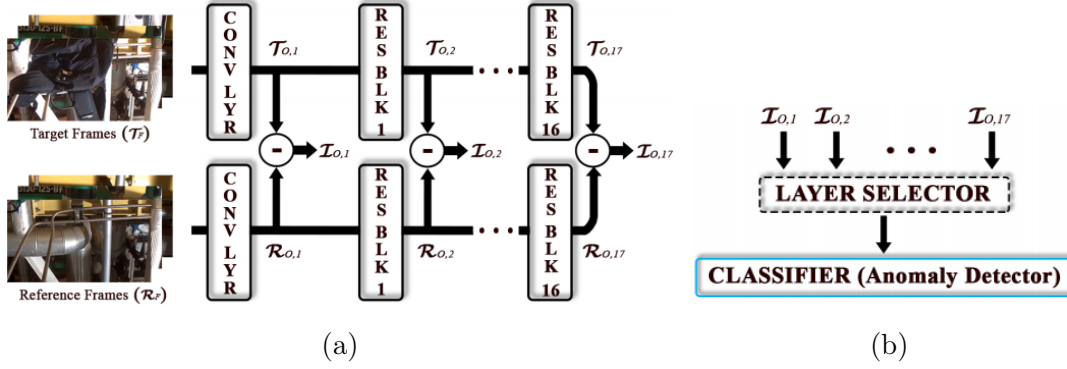


Figure 2.5: Scheme proposed in [2]. (a) Two aligned frames, target \mathcal{T}_F and reference \mathcal{R}_F , are passed through the convolutional layers of Resnet-50. Their feature maps obtained from different layers are subtracted. (b) The layer selector is a term used by the authors to say that features originated from the same layer are fed to a classifier and the layer with the best result is selected. (Images were adapted from [2])

ranging from $i - 2$ to $i + 2$ was assigned to frame F_i .

Their experiments were also performed replacing the random forest classifier with a fully-connected (FC) layer (2-layer perceptron). The best results were obtained by the random forest classifier with features extracted from Layer 4, the 3rd residual layer.

2.4.4 Experimental Results with VDAO Database

Within the scope of the problem covered in this work, given the classification of an input frame, a true positive (TP) it is considered if the input frame has an abandoned object in the scene and the classifier correctly labeled it as anomalous. If the classifier could not identify an existing anomaly in the input frame, it is considered a FN. Thus, the desired result is obtained when the model considers all anomalous frames as *positives* and all non-anomalous frames as *negatives*. Such evaluations can be measured by means of the true positive rate (TPR) and false positive rate (FPR), expressed as

$$\text{TPR} = \frac{\text{TP}}{\text{number of positive ground-truth samples}} \quad (2.5)$$

and

$$\text{FPR} = \frac{\text{FP}}{\text{number of negative ground-truth samples}}. \quad (2.6)$$

TPR is the proportion of TP frames among all frames that contain the anomalies in a given set. FPR is the proportion of FP frames among all frames that do not contain anomalies in a given set.

The trade-off between TRP and FPR can be measured with the receiver operating characteristic (ROC) curve as exemplified in Figure 2.6. Each point in the ROC curve represents an operating point, whose best value is obtained when TPR=1 and FPR=0, representing the best possible classification.

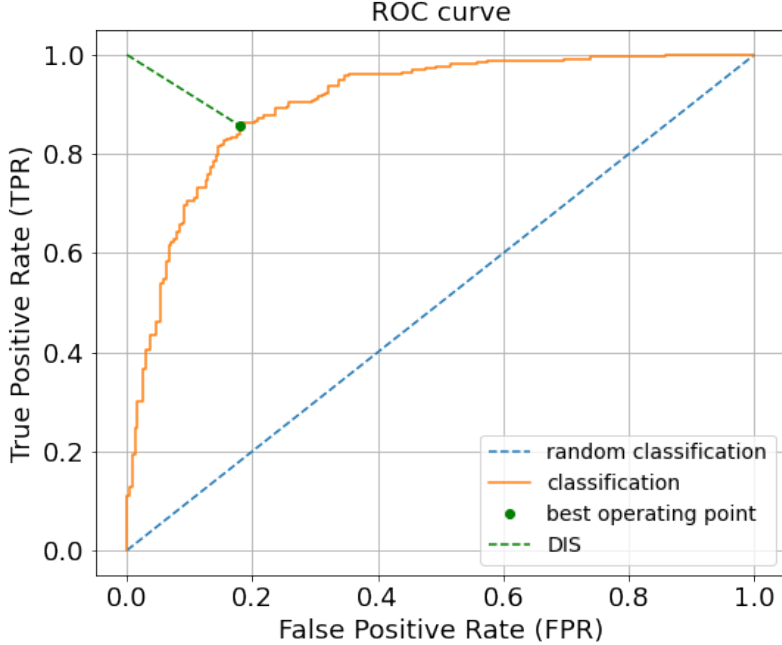


Figure 2.6: Example of ROC curve of a mock classification example. The distance between the ideal point (TPR=1, FPR=0) to the best operating point (in green) is called DIS, whose ideal value is 0.

The benchmark metric used by the works in Table 2.4 to evaluate the classifications of target frames in the VDAO-200 database is the DIS, which is the minimum distance of an operating point to the point of ideal performance as shown in Figure 2.6. The DIS metric is computed by

$$\text{DIS} = \sqrt{(1 - \text{TPR})^2 + \text{FPR}^2}. \quad (2.7)$$

Table 2.3 compares the results obtained by [2] using different convolutional layers as feature extractors. Layer 4, the 3rd residual layer, obtained the lowest DIS, therefore, the best result.

Now, considering all the works that used the VDAO database as benchmarking for the anomaly detection problem, Table 2.4 summarizes True Positive Rate (TPR), False Positive Rate (FPR) and DIS. The best results are highlighted.

Even though the MCBS method [30] obtained the highest TPR, their FPR was the worst. The lowest FPR was obtained by [2], but with lower TPR than [52], which reached the best DIS.

The work developed in [2] has a potential for improvement and was used as

Table 2.3: Classification results on VDAO frames applying CNN to extract features reported in [2]. TPR, FPR and DIS values are calculated using Equations (2.5), (2.6) and (2.7) respectively.

CNN + RF			
Layer	TPR	FPR	DIS
1	0.68	0.32	0.45
2	0.73	0.30	0.40
3	0.69	0.24	0.39
4	0.74	0.25	0.36
5	0.68	0.27	0.42
6	0.68	0.27	0.42

Table 2.4: Classification results on VDAO frames achieved by different works. TPR, FPR and DIS values are calculated using Equations (2.5), (2.6) and (2.7) respectively.

Method	TPR	FPR	DIS
STC-mc [70]	0.48	0.41	0.66
DAOMC [28]	0.89	0.46	0.47
MCBS [30]	0.99	0.98	0.98
mcRoSuRe-A [71]	0.95	0.37	0.37
CNN+MLP [2]	0.66	0.28	0.44
CNN+RF [2]	0.74	0.25	0.36
mcDTSR [52]	0.88	0.26	0.29

starting point for this thesis. In [2], the features extracted with the first convolutional layers of the Resnet-50 were used to train a random forest classifier, whose hyperparameters were not totally optimized. The number of trees in the forest was limited to 100 without limiting their depths. In order to avoid misclassification of individual frames, a voting window with fixed size of 5 was used, which means the final classification of each frame considered the results of two previous and two following frames. They did not evaluate other voting window sizes. The threshold of the probabilities of the class was the only parameter optimized by [2], leaving a gap to be explored. For that, their results were first checked with an independent implementation, then, some parameters were explored with a Bayesian optimizer algorithm to help finding the best combination of trees, depth, voting window, and threshold.

2.5 Conclusion

This chapter reviews the concept of anomaly detection and how it differs from the object detection task. Different approaches to detect anomalies in videos were reviewed, including popular databases used to train and evaluate such systems. The VDAO database and its testing set, the VDAO-200, both adopted by our work, were presented. The evaluation metric DIS, used to benchmark results on the VDAO-200,

was explained, and the approaches using the VDAO and their results were described.

Based on the work [2], the scheme shown in Figure 1.1 was modified resulting in a new pipeline and is described in the next chapter. In the proposed pipeline, a temporal alignment block is introduced to associate a reference frame with a target one. A Bayesian optimizer block was also inserted into the pipeline so the best hyperparameters can be used to train a random forest classifier.

Chapter 3

Deep Learning-Based Anomaly Detection

In this chapter a novel strategy used to extract, classify, and find the best parameters to identify anomalous frames is described. Based on a reference video, without anomalies, the process presented in this chapter is able to classify each frame of an unseen video determining the presence or absence of anomalies. The first work exploring features extraction with CNNs [2] in the VDAO database, described in Chapter 2, was improved. With a similar structure, our results surpass the ones of the aforementioned work, having the advantage of operating in real time. Figure 3.1 shows the processing stages covered in this chapter.

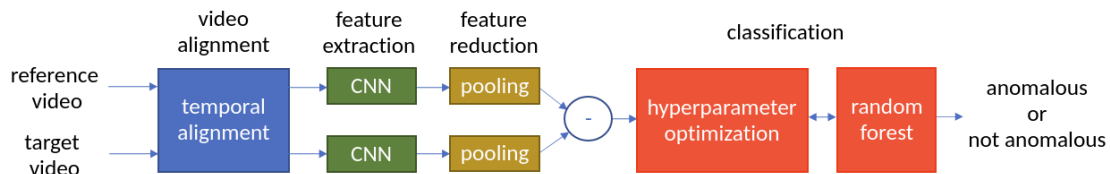


Figure 3.1: Pipeline proposed in this chapter. A pooling layer is used to reduce the feature dimensions, and a hyperparameter optimization block is introduced to find the best classifier configuration.

Initially, both target and reference videos are synchronized with a video alignment technique, which is presented in Section 3.1. Each pair of synchronized reference and target frames is then passed by a pre-trained feature extraction deep neural network to produce a pair of deep-feature tensors. As proposed by [2], the deep features are spatially reduced by a pooling operation. Then, these features are subtracted producing a single tensor of features. The feature extraction module is discussed in Section 3.2. In the classification phase, a random forest classifier is trained to categorize the tensor of features as anomalous or not anomalous. The hyperparameters of the classifier are found through an iterative hyperparameter optimization process. The classifier and the hyperparameter optimization blocks are

covered in Section 3.3 and Section 3.4, respectively. To avoid data contamination, a split of the videos into folds is necessary as described in Section 3.5. Two approaches are used to optimize the hyperparameters. First, the same set of hyperparameters is used by all folds. The other strategy is to obtain an optimized set of hyperparameters for each fold. In the end, the results and conclusions are presented in Sections 3.7 and 3.8 respectively.

3.1 Temporal Video Alignment

Dynamic time warping (DTW) [3] is a very popular algorithm used to align two temporal sequences. DTW has been widely applied to explore similarities in videos [77] and audio signals [10]. Although DTW obtains a global optimal solution, it does not necessarily obtain the best alignment for each individual sample [78]. Different works suggest modifications in DTW in order to support specific applications. The work in [79], for instance, proposes changes in the original DTW to correct the so-called *singularities*, which occur when an undesirable behavior shows up during the alignment. In some works, a singularity can appear when a single point on a time series maps onto a large section of the other time series, as shown in Figure 3.2.

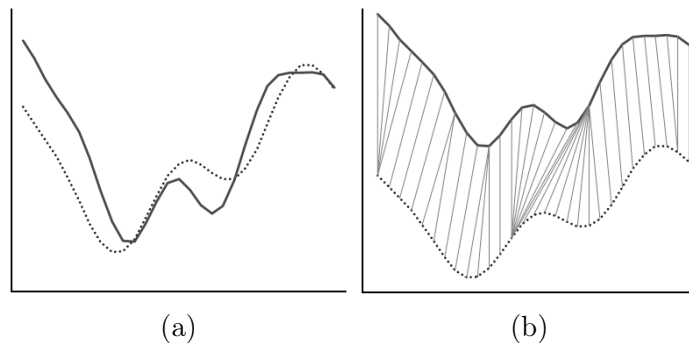


Figure 3.2: Alignment of signals performed with DTW [3]. (a) signals to be aligned. (b) alignment produced by DTW. Note that a short region of the continuous line matches a large region of the dotted line. Images adapted from [79].

Not all works using the VDAO database applied DTW to align the frames. The work in [80], for example, uses a maximum-likelihood approach based on the video motion data to estimate when the robot reaches the end of the rectilinear track. In that work, the authors consider that the robot is moving with a constant speed along the track, until it reaches the end of its course. By knowing the initial and final position of the robot, the frames in-between are properly aligned.

As our work considers [2] as baseline, it would be natural to use DTW as the temporal alignment method. Nonetheless, a drawback of this method is that it prevents the alignment from being performed in real time, once both signals have to

be fully acquired beforehand. In practical cases, the anomaly must be detected as soon as possible, and waiting for the whole target video to be acquired may preclude real time applications.

To address this issue, our work proposes a very simple temporal alignment based on a fixed-length sliding window that moves on the reference signal. By receiving a target frame, our approach finds the most similar reference frame within a limited region in the reference video.

Similarly to the standard DTW, we also consider that the first frame of the target video is aligned with the reference video. For that, we developed a tool to visually align the first frame of all target videos of the VDAO database with their respective reference frame, as shown in Figure 3.3.

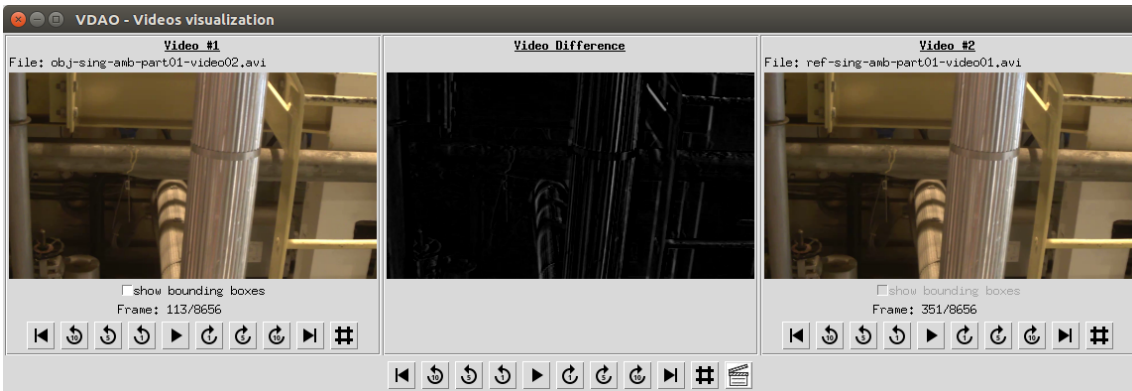


Figure 3.3: Tool developed to help the alignment frames of two videos. On the left, the first frame of the trajectory of the robot regarding the target video *obj-sing-amb-part01-video02.avi* is selected. On the most-right, a frame of the reference video *ref-sing-amb-part01-video01.avi* is shown. In the center, the difference between the target and reference frames is displayed. By navigating through the reference frames and observing their difference images, it is possible to find the reference frame that best matches the target frame.

Given a target video T , we visually prealign its initial frame T_0 to its best frame correspondence of the reference video R . Let us suppose that the best visual correspondence of the target frame T_0 is the j^{th} frame of the reference video, so that $T_0 \rightarrow R_j$ as in Figure 3.4. The correspondence of the next target frame T_1 in R is found by measuring the distance of T_1 with all frames within a window limited by R_{j-a} and R_{j+b} , as detailed in Figure 3.5. The search window $W_{a,b}$ limits the search range for the best match. In summary, the alignment process of the videos can be interpreted as finding the correspondence of a target frame T_i in R within window centered in R_j with size $(b + a + 1)$, such that $T_{i-1} \rightarrow R_{j-1}$.

By limiting the size of the window $W_{a,b} = W_{5,5}$ ($a = 5$ and $b = 5$) to 11, we set the search in a fixed region of 11 frames in the reference video. This way we avoid the need to measure similarities in temporally distant frames. The window

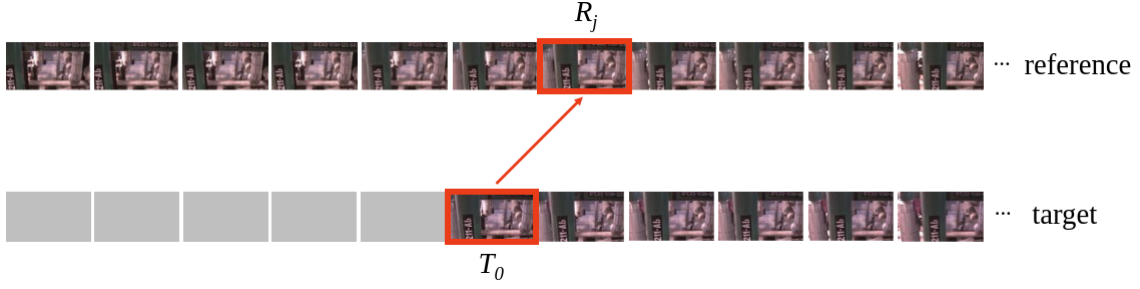


Figure 3.4: Initially, the target frame T_0 is visually aligned with the j^{th} frame in the reference video ($T_0 \rightarrow R_j$).

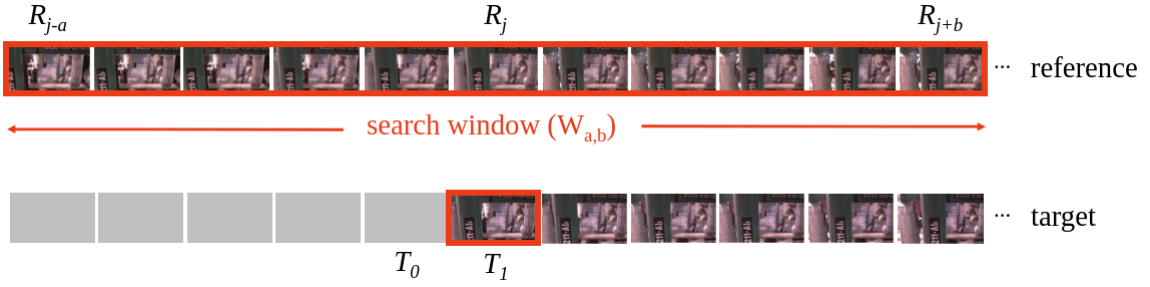


Figure 3.5: Alignment of target frame T_1 considering a search window $W_{a,b}$. The alignment of a target frame T_1 is performed by finding its best reference correspondence within a search window $W_{a,b}$, centered in the frame R_j , which is the best match of T_1 's previous frame, so that $T_0 \rightarrow R_j$.

size equals to 11 was obtained empirically.

The similarity between a target and a reference frame was measured with the Frobenius distance, defined as:

$$\text{similarity}(T_i, R_j) = \sqrt{\sum_x \sum_y (T_i(x, y) - R_j(x, y))^2}. \quad (3.1)$$

Although the alignment method used in our work does not find the globally optimal alignment as in DTW, it has some advantages:

- Real time: Differently from the standard DTW, our method does not demand both signals to be fully acquired to start the alignment process. If that were the case, one would have to wait for the complete acquisition of the target video to start the alignment, leading to undesired delay in our anomaly detector.
- Frames may have inverted orders: as the alignment permits correspondences, such that $T_i \rightarrow R_j$ and $T_{i+1} \rightarrow R_z$ with $z < j$, two consecutive reference frames may exchange positions. An example is shown in Figure 3.6. In practice, this alignment compensates undesired movements of the robot and small camera jitters during the video acquisition.

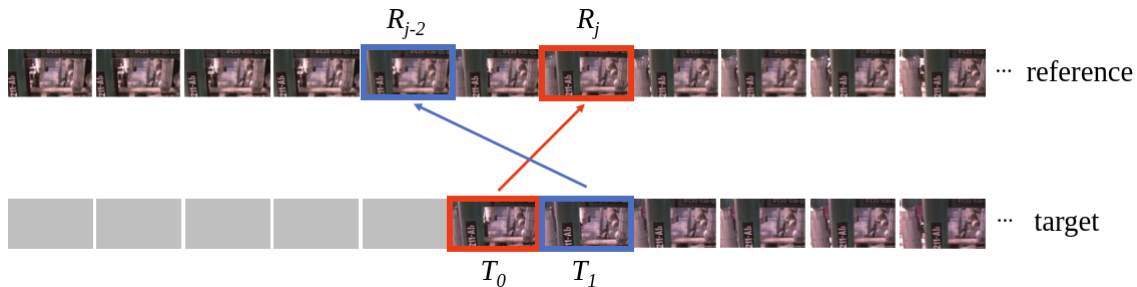


Figure 3.6: Example matching the target frame T_1 with reference frame R_{j-2} , which is temporally behind the match of T_0 .

Differently than a neural network, where the samples are passed to the model in batches, the random forest classifier needs to receive all training samples simultaneously to build the trees. Therefore, the temporal alignment was used separately from the pipeline to align the training samples.

After the video synchronization, all frames of the training target videos are matched with a frame of their corresponding reference video. Each pair of aligned frames ($T_i \rightarrow R_j$) is passed by a sequence of convolutional layers to have their features extracted, and evaluated by a random forest classifier.

During the acquisition of some target videos, the robot moved along a wider path. In consequence, some target videos show parts of the scenario that are not present in the reference videos and vice versa. Thus, if the robot moved in a longer path in a reference video than its associated target video, the far end frames of the reference video are manually discarded. Consequently, all scenes in the target video can be found in the reference video.

The number of aligned frames per video used for training is presented in Table 3.1. About 23% of the frames are anomalous and 77% of the frames do not contain anomalies. An auxiliary list containing the position of the aligned frames is available online at [81].

Table 3.1: Number of frames per video in the VDAO database after temporal alignment.

#	video	anomalous	not anomalous	total
1	obj-sing-amb-part01-video01	1,732 (22%)	5,987 (78%)	7,719
2	obj-sing-amb-part01-video02	1,884 (24%)	5,827 (76%)	7,711
3	obj-sing-amb-part01-video03	2,120 (27%)	5,596 (73%)	7,716
4	obj-sing-amb-part01-video04	821 (21%)	3,038 (79%)	3,859
5	obj-sing-amb-part01-video05	812 (21%)	3,045 (79%)	3,857
6	obj-sing-amb-part01-video06	1,488 (19%)	6,228 (81%)	7,716
7	obj-sing-amb-part01-video07	2,200 (29%)	5,514 (71%)	7,714
8	obj-sing-amb-part01-video08	1,200 (16%)	6,504 (84%)	7,704

Table 3.1 – continued from previous page

#	video	anomalous	not anomalous	total
9	obj-sing-amb-part01-video09	1,933 (25%)	5,772 (75%)	7,705
10	obj-sing-amb-part01-video10	2,180 (28%)	5,529 (72%)	7,709
11	obj-sing-amb-part02-video01	1,946 (20%)	7,672 (80%)	9,618
12	obj-sing-amb-part02-video02	1,900 (20%)	7,744 (80%)	9,644
13	obj-sing-amb-part02-video03	428 (04%)	9,175 (96%)	9,603
14	obj-sing-amb-part02-video04	1,857 (19%)	7,777 (81%)	9,634
15	obj-sing-amb-part02-video05	1,808 (19%)	7,793 (81%)	9,601
16	obj-sing-amb-part03-video01	2,780 (29%)	6,829 (71%)	9,609
17	obj-sing-amb-part03-video02	1,534 (16%)	8,066 (84%)	9,600
18	obj-sing-amb-part03-video03	1,868 (19%)	7,735 (81%)	9,603
19	obj-sing-amb-part03-video04	505 (11%)	4,292 (89%)	4,797
20	obj-sing-amb-part03-video05	521 (11%)	4,275 (89%)	4,796
21	obj-sing-amb-part03-video06	2,395 (25%)	7,196 (75%)	9,591
22	obj-sing-amb-part03-video07	2,649 (28%)	6,968 (72%)	9,617
23	obj-sing-amb-part03-video08	3,170 (33%)	6,454 (67%)	9,624
24	obj-sing-amb-part03-video09	3,251 (34%)	6,341 (66%)	9,592
25	obj-sing-amb-part03-video10	2,167 (23%)	7,444 (77%)	9,611
26	obj-sing-amb-part03-video11	3,743 (39%)	5,852 (61%)	9,595
27	obj-sing-amb-part03-video12	1,092 (23%)	3,703 (77%)	4,795
28	obj-sing-amb-part03-video13	1,057 (22%)	3,734 (78%)	4,791
29	obj-sing-amb-part03-video14	1,213 (25%)	3,584 (75%)	4,797
30	obj-sing-amb-part03-video15	1,222 (26%)	3,567 (74%)	4,789
31	obj-sing-amb-part03-video16	2,272 (24%)	7,314 (76%)	9,586
32	obj-sing-amb-part03-video17	2,993 (31%)	6,616 (69%)	9,609
33	obj-sing-ext-part01-video01	1,744 (23%)	5,990 (77%)	7,734
34	obj-sing-ext-part01-video02	1,883 (24%)	5,878 (76%)	7,761
35	obj-sing-ext-part01-video03	2,135 (28%)	5,621 (72%)	7,756
36	obj-sing-ext-part01-video04	1,667 (22%)	6,065 (78%)	7,732
37	obj-sing-ext-part01-video05	1,494 (19%)	6,254 (81%)	7,748
38	obj-sing-ext-part01-video06	2,197 (28%)	5,530 (72%)	7,727
39	obj-sing-ext-part01-video07	1,234 (16%)	6,491 (84%)	7,725
40	obj-sing-ext-part01-video08	1,924 (25%)	5,803 (75%)	7,727
41	obj-sing-ext-part01-video09	2,236 (29%)	5,509 (71%)	7,745
42	obj-sing-ext-part02-video01	1,958 (20%)	7,653 (80%)	9,611
43	obj-sing-ext-part02-video02	1,906 (20%)	7,742 (80%)	9,648
44	obj-sing-ext-part02-video03	454 (05%)	9,153 (95%)	9,607

Table 3.1 – continued from previous page

#	video	anomalous	not anomalous	total
45	obj-sing-ext-part02-video04	1,913 (20%)	7,694 (80%)	9,607
46	obj-sing-ext-part02-video05	1,807 (19%)	7,787 (81%)	9,594
47	obj-sing-ext-part03-video01	2,839 (30%)	6,757 (70%)	9,596
48	obj-sing-ext-part03-video02	1,514 (16%)	8,061 (84%)	9,575
49	obj-sing-ext-part03-video03	1,882 (20%)	7,708 (80%)	9,590
50	obj-sing-ext-part03-video04	1,032 (11%)	8,561 (89%)	9,593
51	obj-sing-ext-part03-video05	2,395 (25%)	7,197 (75%)	9,592
52	obj-sing-ext-part03-video06	2,610 (27%)	6,994 (73%)	9,604
53	obj-sing-ext-part03-video07	3,143 (33%)	6,454 (67%)	9,597
54	obj-sing-ext-part03-video08	3,315 (35%)	6,284 (65%)	9,599
55	obj-sing-ext-part03-video09	2,175 (23%)	7,416 (77%)	9,591
56	obj-sing-ext-part03-video10	3,756 (39%)	5,841 (61%)	9,597
57	obj-sing-ext-part03-video11	2,128 (22%)	7,448 (78%)	9,576
58	obj-sing-ext-part03-video12	2,279 (24%)	7,318 (76%)	9,597
59	obj-sing-ext-part03-video13	2,447 (26%)	7,132 (74%)	9,579
60	obj-sing-ext-part03-video14	2,975 (31%)	6,616 (69%)	9,591
Total		117,783 (23%)	386,128 (77%)	503,911

As it can be noted in Table 3.1, the training set contains more non anomalous samples than anomalous ones. For training, the classes were balanced, so that the number of samples of both classes are the same.

3.2 Extracting Features with Resnet-50

After AlexNet [26] won the ImageNet [82] challenge in 2012, the state-of-the-art architectures have been growing deeper and deeper. More recent studies have shown that not only depth, but also width and resolution can lead to better performance [83]. Usually, the deeper the network gets, the more parameters the learning process has to find, demanding more hardware memory. The trade-offs between efficiency, depth and accuracy have been explored by different models [84, 85].

ResNet [4], a short name for Residual Network, is a deep CNN architecture inspired by the results obtained by deep models on the ImageNet dataset. Although the deeper the models are, the better results they tend to produce [4], the problem of vanishing gradients is even more likely to occur in such cases. To overcome this problem, ResNet architectures propose a bypass structure, also named as *identity mapping*, that combines features from shallower layers to deeper layers. Such

mapping is presented in residual blocks along the layers of ResNets, as shown in Figure 3.7.

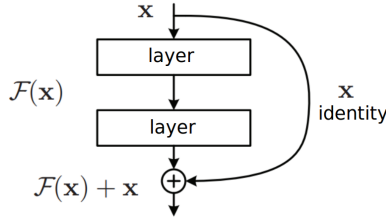


Figure 3.7: Residual block (adapted from [4]).

The identity mapping $\mathcal{F}(\mathbf{x}) + \mathbf{x}$ takes the feature map \mathbf{x} from a previous layer and performs an element-wise addition with the $\mathcal{F}(\mathbf{x})$, the output of the stacked layers. The element-wise addition requires the dimensions of \mathbf{x} and \mathcal{F} to be equal. In cases where the previous layer dimensions do not match the output, usually a 1×1 convolution is applied in \mathbf{x} , so it can have the same dimensions of $\mathcal{F}(\mathbf{x})$.

The original work Resnet released 5 architectures (Resnet-18, Resnet-34, Resnet-50, Resnet-101, and Resnet-152), differing by the number of convolutional layers. To keep consistency with our baseline work [2], and have comparable results, the Resnet-50 architecture was also used in our work to extract features from both reference and target frames.

Resnet-50 is 50-layer deep formed by 1 convolutional layer, 1 max pooling, 1 average pooling and 4 different stages. Each stage consists of repeated convolutional layers and 1 identity block. Figure 3.8 shows the original structure of Resnet-50 architecture.

Instead of training the whole network from scratch to adjust over 26 million trainable parameters [4, 84], a pre-trained Resnet-50 model with Imagenet [82] was used. By using such pre-trained model, we eliminate the need of the long training phase, which also demands a variety of training samples.

Our goal is straightforward: to obtain features from the aligned pair of frames $T_i \rightarrow R_j$ by convolutional layers and let a classifier determine the existence of anomalies in the frame T_i . Instead of using the total number of convolutional layers, we investigate which layer can provide the most adequate features to our problem. The spatial dimension of a feature map, given by W_{out} and H_{out} , follows the relations given by:

$$W_{\text{out}} = \frac{W - D_w + 2P}{S_w} + 1, \quad (3.2)$$

$$H_{\text{out}} = \frac{H - D_h + 2P}{S_h} + 1, \quad (3.3)$$

where D_w and D_h are the horizontal and vertical dimensions of the filter, respec-

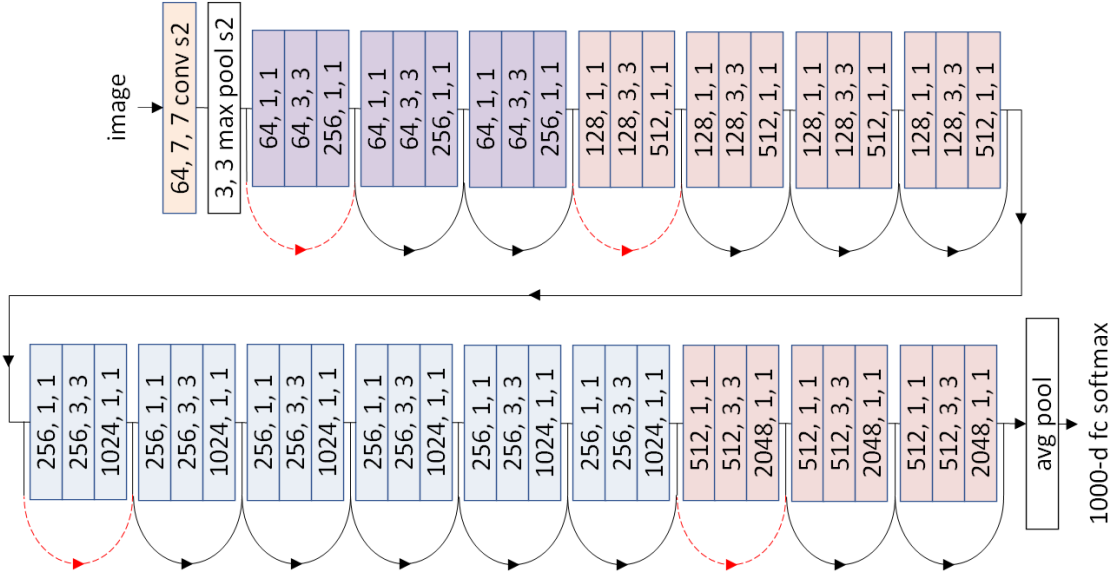


Figure 3.8: Original Resnet-50 architecture is formed by 50 layers, being 48 convolutional divided into 4 stages (here represented by different colors). Each stage contains 3, 4, 6 and 3 residual blocks respectively. The arrows indicate the identity mapping. The red outlined arrows indicate that the dimension of the feature map increases and 1x1 convolutions are applied to make the dimensions match.

tively, W and H represent the horizontal and vertical dimensions of the input image or feature map, P is the amount of padding, being 3, 1, or 0 in Resnet-50. The stride step along horizontal and vertical directions is given by S_w and S_h respectively. The stack of feature maps can be represented by a tensor of features, whose depth is given by the number of convolutional filters. For example, if we consider a 201x201x3 (width, height, and channels) input image to be passed through the first convolutional layer, composed by 64 7x7x3 filters with stride step 2 and padding 3, the output tensor will have dimensions 101x101x64.

The work [2] evaluated tensors generated by 17 distinctive layers of the Resnet-50. If the original frames of the VDAO with resolution 720x1280 were used, the dimensions of the feature maps would be very high, making the training of the classification step extremely costly, and requiring a lot of storage space. To work around this issue, the authors from [2] reduced the input frame resolution by half in each dimension (360x640) and average pooling layers were used in the end of each output layer. Figure 3.9 shows the modified architecture including the extra average pooling used to reduce the feature maps. In our work, for the sake of comparison, we also evaluate the outputs produced by the same 17 layers considering the same pooling sizes and the same input resolution.

A comparison of the feature map sizes extracted with and without pooling is shown in Table 3.2. The pooling operation reduces the total number of features drastically. In some layers the reductions were up to 99.9%.

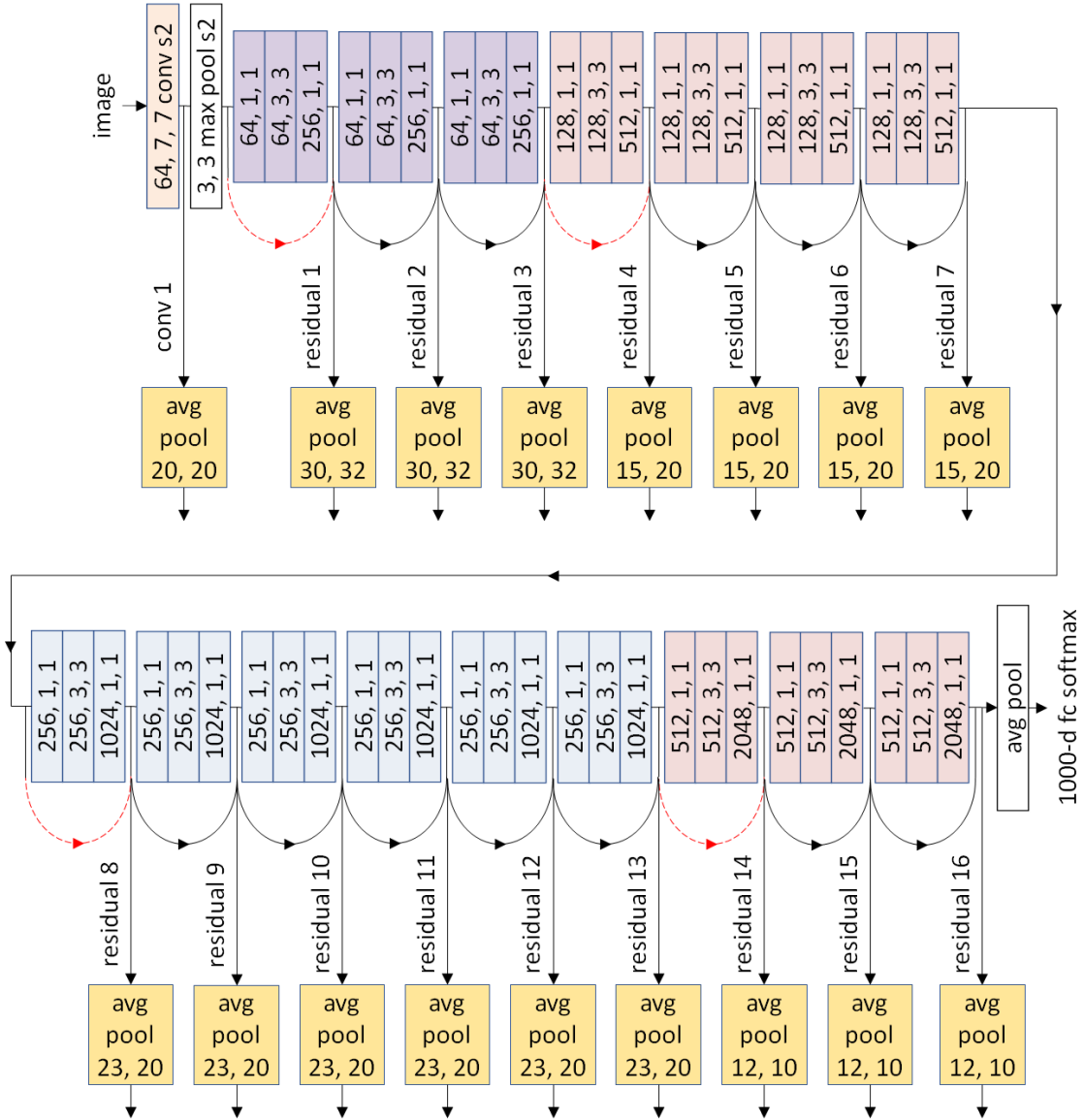


Figure 3.9: Modified Resnet-50 architecture proposed by [2]. After extracting the feature maps from each point of the network, average poolings with different sizes 21×21 , 28×28 , 14×14 , or 7×7) were used to reduce the number of features.

A popular CNN model that uses the same weights to work with two different input is the so called siamese convolutional neural networks [86]. Given two different inputs, they are passed by two identical CNNs to produce two feature tensors, one for each CNN. These tensors are then compared by some loss function that is able to distinguish how different the provided inputs are. Similarly to the siamese convolutional neural networks, given a pair of aligned target and reference frames, convolutional layers are used to extract features of a pair of images. In our case, Resnet-50 convolutional layers are used as siamese networks, as they receive a pair of reference and target frames and produces two feature tensors: one containing the features of the reference frame, and the other containing the features of the target

Table 3.2: Dimensions of feature tensors in the format *channels, height, width* produced by different Resnet-50 layers given an RGB input frame of dimensions 360x640x3. The modified version (on the right) introduces a pooling layer to reduce the spatial dimension of the feature maps.

original Resnet-50			modified Resnet-50 [2]		
layer	output dimension	total features	pooling size	output dimension	total features
conv 1	$64 \times 180 \times 320$	~ 3.68 M	20×20	$64 \times 9 \times 16$	9,216
residual 1	$256 \times 90 \times 160$	~ 3.68 M	30×32	$256 \times 3 \times 5$	3,840
residual 2	$256 \times 90 \times 160$	~ 3.68 M	30×32	$256 \times 3 \times 5$	3,840
residual 3	$256 \times 90 \times 160$	~ 3.68 M	30×32	$256 \times 3 \times 5$	3,840
residual 4	$512 \times 45 \times 80$	~ 1.84 M	15×20	$512 \times 3 \times 4$	6,144
residual 5	$512 \times 45 \times 80$	~ 1.84 M	15×20	$512 \times 3 \times 4$	6,144
residual 6	$512 \times 45 \times 80$	~ 1.84 M	15×20	$512 \times 3 \times 4$	6,144
residual 7	$512 \times 45 \times 80$	~ 1.84 M	15×20	$512 \times 3 \times 4$	6,144
residual 8	$1,024 \times 23 \times 40$	~ 0.94 M	23×20	$1,024 \times 1 \times 2$	2,048
residual 9	$1,024 \times 23 \times 40$	~ 0.94 M	23×20	$1,024 \times 1 \times 2$	2,048
residual 10	$1,024 \times 23, 40$	~ 0.94 M	23×20	$1,024 \times 1 \times 2$	2,048
residual 11	$1,024 \times 23, 40$	~ 0.94 M	23×20	$1,024 \times 1 \times 2$	2,048
residual 12	$1,024 \times 23, 40$	~ 0.94 M	23×20	$1,024 \times 1 \times 2$	2,048
residual 13	$1,024 \times 23, 40$	~ 0.94 M	23×20	$1,024 \times 1 \times 2$	2,048
residual 14	$2,048 \times 12, 20$	~ 0.49 M	12×10	$2,048 \times 1 \times 2$	4,096
residual 15	$2,048 \times 12, 20$	~ 0.49 M	12×10	$2,048 \times 1 \times 2$	4,096
residual 16	$2,048 \times 12, 20$	~ 0.49 M	12×10	$2,048 \times 1 \times 2$	4,096

frame. As suggested in [2], after the average pooling operation, an element-wise subtraction is applied on the reference and target feature tensors and the result is a single feature tensor.

In this chapter, considering the pooling and subtraction operations applied in the deep features, we want to investigate which layer of the Resnet-50 is able to produce the best discriminative features for a random forest classifier to distinguish if the target frame T_i is anomalous or not. During the attainment of the feature maps, a massive amount of data is extracted and stored. The subtracted deep features of the 17 sections of the Resnet-50 represent a massive amount of data, requiring 245 GB of storage space.

Next section covers the classification step performed on the feature maps.

3.3 Random Forest for Image Classification

Ensemble-learning methods use multiple learning models to construct a set of hypotheses to improve the performance of a regression or classification problem [87]. Experimental evidences have shown that ensemble methods are more accurate if

compared to methods that use a single hypothesis [88, 89]. Random forest is an ensemble method that combines tree predictors, such that each tree depends on the values of features randomly sampled. Its generalization error converges as the number of trees in the forest increases and depends on the classification strength of each individual tree [76]. The final prediction is obtained by the combination of the results of the trees in the forest.

Random forest classifiers are widely used in computer vision problems involving different tasks. Reference [90] successfully predicts 3D positions of body joints using a random forest classifier to categorize pixels of a depth-image. In [21], the problem of matching feature points was solved with a classification technique. They showed that random forest classifier is more efficient for the keypoint recognition problem in comparison to k -means and nearest neighbor classifiers. The work in [91] employed features obtained by sub-windows at arbitrary positions and scales using random forest to classify images, outperforming k -means in training and testing time, memory consumption, and classification accuracy.

An advantage of the random forest among other classifiers is the fact that only few hyperparameters have to be set during the training. Instead of using the default values used by [2], in this work we make a guided search to tune the following hyperparameters:

- **Maximum number of trees:** as the random forest classifier is an ensemble of decision trees, the number of trees in the forest can be adjusted. The work in [92] states that there is a limit beyond which increasing the number of trees does not result in better accuracy, and a larger number of trees in a forest may only increase its computational cost.
- **Maximum depth allowed:** The deeper the trees are, the more splits they may have and, consequently, the ability to capture more information about the data is increased [93].
- **Threshold:** As the random forest is an ensemble of trees, the predicted class probabilities of an input sample are computed as the mean predicted class probabilities of all the trees in the forest. By default, a random forest classifier labels a frame as anomalous if the predicted probability of belonging to the anomalous class is greater than or equal to 0.5. In this work, the threshold for the predicted class probabilities is considered a hyperparameter to be adjusted.
- **Voting window:** As this solution should be applied to videos, and a frame-based classifier is being used, to eliminate the misclassification of a small number of frames in a group of consecutive frames, a voting window considers the results of consecutive frames. The voting window hyperparameter is adjusted

apart from the random forest classifier as it is not related to the classifier. The intention of using this voting window is to eliminate false predictions.

To form a final optimal classifier, it is necessary to find the best combination among the four hyperparameters aforementioned. Practical experiments were performed to define limits of each hyperparameter as shown in Table 3.3.

Table 3.3: Ranges of values used by the hyperparameter tuning.

parameter	minimum limit	maximum limit
trees	1	210
max depth	1	150
threshold	0.4	0.6
voting window	0	100

In the next section the method used to search for the best hyperparameters will be described.

3.4 Hyperparameter Optimization

Classification models usually have many hyperparameters to be adjusted. It can be a challenging task to choose the right values when the number of degrees of freedom is high [94]. The most popular approaches to fine-tune the parameters are: grid search [95, 96], randomized search [97, 98] and Bayesian optimization methods [99–101].

The work in [97] investigates the hyperparameter optimization using grid search and randomized search. It states that randomly chosen trials are more efficient for hyperparameter optimization than grid search and that Bayesian optimization methods are more promising due to the weights it gives to each dimension.

In this work, as there are 4 hyperparameters to be tuned (trees, max depth, threshold, and voting window size), for each one of the 17 Resnet-50 layers used to obtain the features (see Figure 3.9), we decided to apply the Bayesian optimization due to its robustness and also it does not require much supervision [99, 100, 102] nor hyperparameters adjustments.

Bayesian optimization is an iterative process that constructs a posterior distribution of functions to best describe a function to be optimized. In each iteration, a different combination of parameters is tested by the model, improving the posterior distribution. A Bayesian process is used to decide the next evaluation points based on the exploration-exploitation trade-off. Along the process, the algorithm becomes more certain of which regions in the parameter space are worth exploring. The Bayesian optimization process used in this work follows the implementation

in [102], which considers a Gaussian process (GP) to express assumptions of the unknown classification function that we want to optimize. This implementation follows the work in [100].

As the number of iterations is undetermined, we allowed the Bayesian optimization to explore 1,000 combinations of parameters for each layer. During each iteration, a different combination of parameters (trees, max depth, threshold and voting window) is evaluated by the optimizer.

The videos were separated into 9 folds, each having their own training, validation and testing sets. The Bayesian optimization process was performed in each validation fold independently as it will be explained further in Section 3.5.

Since the robot speed is relatively slow, consecutive frames are very similar. To avoid using a high number of similar frames to train the classifier, we decided to sample the training frames skipping every 17 frames of the training videos. Also, as shown in Table 3.1, the number of positive pairs $T_i \rightarrow R_j$ (where T_i contains anomalies) in the training set is proportionally less than the number of negative pairs $T_i \rightarrow R_j$ (where T_i do not contain anomalies), thereby, the quantity of samples of both classes (anomalous and not anomalous) was balanced.

Before presenting the methodology used to compute the results obtained by each layer, next section details the methodology used to separate the samples into training, validation and test sets.

3.5 Database Subdivision

Historically, previous works that used the VDAO database [2, 52, 70–72, 80, 103, 104] compare their results using the VDAO-200 testing database. As already cited in Section 2.3.1, 59 out of the 60 single-object videos of the VDAO database are presented in the VDAO-200 testing set. To avoid contaminating the training set with videos from the testing set, we split the samples into folds and perform cross-validation in each fold to find the hyperparameters for our models using the validation samples.

As there are nine classes of objects in the VDAO database, initially the videos are separated into nine folds. Each fold is characterized by the class of the object left out. In other words, the fold F_{shoe} represents the fold containing the videos with all objects but the shoe, which is left out for testing. Thus, the videos with the object *shoe* does not influence the training nor the validation of the fold F_{shoe} . Table 3.4 contains the classes of objects used in all folds.

Inside each fold, a cross-validation process is run considering the proportion of 7:1 (videos of seven objects for training and videos of one object for validation). This proportion was obtained after experimenting different proportions (7:1, 6:2, 5:3 and 4:4). Our experiments revealed that the closest results between training

Table 3.4: Folds containing videos of each class. Each one of the 9 folds contains videos of 8 classes, leaving one class out, which is used exclusively for testing.

Folds	black backpack	black coat	brown box	camera box	dark-blue box	pink bottle	shoe	towel	white jar
$F_{\text{black backpack}}$	✗	✓	✓	✓	✓	✓	✓	✓	✓
$F_{\text{black coat}}$	✓	✗	✓	✓	✓	✓	✓	✓	✓
$F_{\text{brown box}}$	✓	✓	✗	✓	✓	✓	✓	✓	✓
$F_{\text{camera box}}$	✓	✓	✓	✗	✓	✓	✓	✓	✓
$F_{\text{dark-blue box}}$	✓	✓	✓	✓	✗	✓	✓	✓	✓
$F_{\text{pink bottle}}$	✓	✓	✓	✓	✓	✗	✓	✓	✓
F_{shoe}	✓	✓	✓	✓	✓	✓	✗	✓	✓
F_{towel}	✓	✓	✓	✓	✓	✓	✓	✗	✓
$F_{\text{white jar}}$	✓	✓	✓	✓	✓	✓	✓	✓	✗

and validation are obtained with the proportion 7:1, as it uses more objects for training. Table 3.5 illustrates the division of the videos into training and validation inside the fold $F_{\text{white jar}}$. Each column of the table represents the classes of objects used in this fold. By omitting the object white jar (restricted for testing), among the eight remaining objects, there are eight possible sets considering seven classes for training and one class for validation. Applying the same idea to all nine folds, there is a total of 72 sets ($9 \times 8 = 72$) considering seven classes of objects for training, one class for validation and one for testing.

The training and validation samples of each set are used by the Bayesian optimization process to find the best hyperparameters to produce the final random forest classifier used for testing. The training samples are used to train a random forest classifier, which is evaluated in its corresponding validation samples. This way, the iterative Bayesian optimization process aims to find the best hyperparameters (trees, depth, threshold and voting window) by evaluating a set of classifiers given validation samples. Once the Bayesian optimization process is completed, the hyperparameters associated to the best result are used to train a final classifier with both training and validation samples. The final classifier is then evaluated on the testing samples. In next section, the iterative process used by the Bayesian optimization process is explained, and the computation of the results is clarified.

Table 3.5: Objects used in the videos to train and validate for the fold $F_{\text{white jar}}$. The videos with the target object (white jar) are left out and will be used in the test only.

sets	black backpack	black coat	brown box	camera box	dark-blue box	pink bottle	shoe	towel
1	validation	training	training	training	training	training	training	training
2	training	validation	training	training	training	training	training	training
3	training	training	validation	training	training	training	training	training
4	training	training	training	validation	training	training	training	training
5	training	training	training	training	validation	training	training	training
6	training	training	training	training	training	validation	training	training
7	training	training	training	training	training	training	validation	training
8	training	training	training	training	training	training	training	validation

3.6 Performance Assessment

This section is dedicated to explain how the results are evaluated during the validation and testing phases.

The cross-validation process is utilized to select a collection of hyperparameters (amount of trees, depth, threshold and voting window) to train a random forest classifier, which will be evaluated on testing samples. As previously explained, the database is split into nine folds, each containing videos of one object to be used in the testing phase, and the remaining objects are separated into training and validation sets. Regarding the application of the hyperparameters to be used in the testing samples of each fold, in this work we considered two approaches:

- approach A: the same set of hyperparameters is shared among all final classifiers used for testing in all folds.
- approach B: each final classifier used for testing uses a specific set of hyperparameters selected especially for each fold.

Approach A involves the generation of 9 distinct classifiers, but they all use the same hyperparameters. In this strategy, the searching process of the unique hyperparameters involves a single run of the Bayesian optimization process considering 72 sets, being eight sets for each fold.

Approach B also involves the generation of nine distinct classifiers, each trained with specific hyperparameters found independently by the Bayesian optimization process. In this approach, the proportion 7:1 was used (videos of 7 objects are used for training and videos of 1 object are used for validation). To obtain the classifier used to test the object *white jar*, for instance, all sets of Table 3.5 were used.

The iterative Bayesian optimization process applied in the cross-validation stage carries the selection of such hyperparameters. The scheme in Figure 3.10 illustrates the cross-validation process considering a total of s sets.

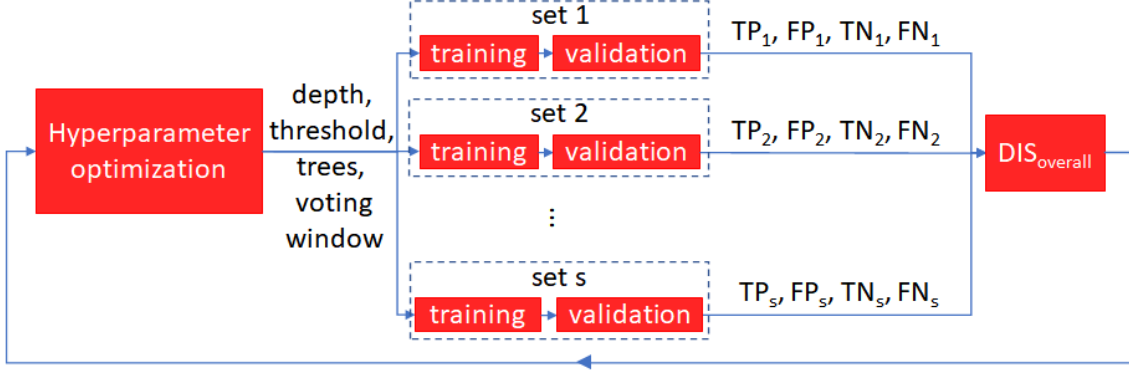


Figure 3.10: Scheme representing the selection of the hyperparameters with cross-validation. The hyperparameter optimization process provides four hyperparameters. In each set (in a total of s) classifier is trained with the provided hyperparameters, evaluated on its respective validation set. The individual results of each set (TP_i, FP_i, TN_i and FN_i) are used to compute the DIS_{overall} metric, which is fed back to the hyperparameter optimization process.

Each set i uses the hyperparameters provided by the Bayesian optimizer to train a random forest classifier with its respective training samples. Then, the classifier of each set i is evaluated on its respective validation samples, producing four results: TP_i, FP_i, TN_i and FN_i , being, respectively, the total number of true positive (TP), false positive (FP), true negative (TN) and false negative (FN) frames computed in all validation videos of the set i . The results of all sets are combined by the DIS_{overall} metric and fed back to the Bayesian optimizer, which will provide new hyperparameters. In this work we repeated this loop 1,000 times and selected the hyperparameters that provided the best DIS_{overall} metric to produce the final random forest classifier to be used in the testing phase.

The assessment of a classifier is measured with the DIS_{overall} metric, which considers all frames of all evaluated videos, and is the same metric applied in [2]. The TPR and FPR used to calculate the DIS in Equation (2.7) accounts the total amount of TP, FP, TN and FN frames in all evaluated videos considering the s sets, and is given by

$$TPR = \frac{\sum_{i=1}^s TP_i}{\sum_{i=1}^s (TP_i + FN_i)} \quad (3.4)$$

and

$$FPR = \frac{\sum_{i=1}^s FP_i}{\sum_{i=1}^s (FP_i + TN_i)}, \quad (3.5)$$

where s is the total amount of sets, $\sum_{i=1}^s (\text{TP}_i + \text{FN}_i)$ is the number of positive ground-truth frames in all s sets and $\sum_{i=1}^s (\text{FP}_i + \text{TN}_i)$ is the number of negative ground-truth frames in all s sets.

Finally, the $\text{DIS}_{\text{overall}}$ is computed by plugging Equations (3.4) and (3.5) into Equation (2.7), which becomes

$$\text{DIS}_{\text{overall}} = \sqrt{\left(1 - \frac{\sum_{i=1}^s \text{TP}_i}{\sum_{i=1}^s (\text{TP}_i + \text{FN}_i)}\right)^2 + \left(\frac{\sum_{i=1}^s \text{FP}_i}{\sum_{i=1}^s (\text{FP}_i + \text{TN}_i)}\right)^2}. \quad (3.6)$$

By using the TPR and FPR (Equations 3.4 and 3.5, respectively) to calculate the DIS (see Equation 2.7), the obtained result is similar to concatenating every video in the validation sets to form a single video and calculates a unique DIS for this longer video.

In approach A, the selection of the hyperparameters illustrated in Figure 3.10 is performed uniquely for all nine folds. Thus, a total of 72 sets ($s = 72$) are evaluated in each iteration of the Bayesian process using the proportion 7:1 (videos of seven objects for training and videos of one object for validation). The advantage brought by this approach is to use the same hyperparameters to create all final classifiers evaluated on the testing samples of each fold. However, in each iteration of the validation process, approach A considers all possible 72 folds to compute a $\text{DIS}_{\text{overall}}$ metric to be fed to the Bayesian optimization, and the final result takes into consideration all object classes producing somehow contaminated results.

In approach B, the selection of the hyperparameters illustrated in Figure 3.10 is performed once for each fold. Therefore, the cross-validation process is run nine times (one for each fold) consisting of eight evaluation sets ($s = 8$) in each iteration of the Bayesian process. See Table 3.5 as a reference for fold $F_{\text{white jar}}$. In approach B, the validation phase for each fold is run independently, being very time-consuming.

In this work, results of approaches A and B will be presented.

When the Bayesian optimization finishes all 1,000 iterations, the set of parameters which obtained the lowest $\text{DIS}_{\text{overall}}$ among all iterations is considered the best set of hyperparameters. In the testing phase, for each fold (F_{shoe} , $F_{\text{white jar}}$, etc), a random forest classifier is trained using the best set of hyperparameters found in the validation phase. As explained in Section 3.5, in each fold videos of one class are left out for testing. Therefore, the random forest classifier used for testing in fold F_{shoe} is trained with videos of 8 classes (black backpack, black coat, brown box, camera box, dark-blue box, pink bottle, towel, and white jar). The videos of the class shoe from the VDAO-200 dataset are used for testing. The $\text{DIS}_{\text{overall}}$ value

considered as the final testing result is obtained with Equation (3.6).

In the next section, the hyperparameters obtained by the Bayesian optimization for approaches A and B will be presented, as well as the results in the validation and testing sets.

3.7 Experimental Results

In this section the results obtained by the proposed pipeline shown in Figure 3.1 are discussed. The best hyperparameter sets obtained by each layer during the validation phase, and the performance achieved for each layer in the testing sets is also presented.

Before presenting the results with the proposed pipeline, we first compared the results in a reduced version of pipeline using two types of pooling operations: average pooling and max pooling. In this initial experiment, the hyperparameter optimizer was not used and the goal is to verify if a max pooling operation reaches better results than the average pooling suggested by [2]. Thus, for each layer, the two types of pooling operations were used to train a random forest classifier with fixed hyperparameters considering 100 trees, fixed threshold equals to 0.5, unlimited maximum depth, and no voting window was considered. As in this experiment there was no hyperparameter tuning, the validation set was not used, and in each fold eight objects were used to train the classifier and one object was used for testing. The DIS metric was used to compare the results obtained in the testing set which are presented in Table 3.6.

As it can be noted in Table 3.6, the average pooling operation obtained better results in 15 layers and in only two deep layers the max pooling performed better. Therefore, we opted to continue using the average pooling in our pipeline and not using the max pooling.

Now, considering the full pipeline with the average pooling operation and the hyperparameter optimization used in the cross-validation, 1,000 iterations were performed. The hyperparameter set which obtained the best DIS value was used to train the final classifier used for testing. As explained in the previous section, the approaches A and B are going to be evaluated.

3.7.1 Results with Approach A

The cross-validation performed by approach A aims to find a unique set of hyperparameters used to test the final classifiers in all nine folds and, therefore, is computationally less expensive than approach B. Thus, in this approach, the random forest classifier is trained using the scheme from Figure 3.10 and the cross-validation is

Table 3.6: Comparative DIS values obtained in the testing sets with features output by each Resnet-50 layer and reduced with average pooling and max pooling operations. The hyperparameters were not optimized, having the same values in [2]. The results are calculated with Equation (3.6).

	avg pooling	max pooling
conv 1	0.3933	0.4443
residual 1	0.3941	0.4830
residual 2	0.3927	0.4891
residual 3	0.3881	0.5290
residual 4	0.4192	0.5574
residual 5	0.4002	0.5238
residual 6	0.3972	0.5372
residual 7	0.4187	0.5265
residual 8	0.5062	0.5639
residual 9	0.5190	0.5886
residual 10	0.4915	0.5457
residual 11	0.5032	0.5391
residual 12	0.4989	0.5273
residual 13	0.5172	0.5307
residual 14	0.5287	0.5164
residual 15	0.5103	0.5201
residual 16	0.5322	0.5239

applied once for all folds. The hyperparameters, validation and testing DIS values for each layer are shown in Table 3.7. The evolution of the cross-validation process is shown in Figure 3.11.

Regarding the validation results, it is evident from Figure 3.11 that the first seven layers (conv1, residual1, residual2, residual3, residual4, residual5, and residual6) obtained the best results. Also, it is noticeable that as better combinations are being found throughout the process, the DIS decreases. For most of the layers, the most expressive reductions of the DIS occur before iteration 400. Layer residual 3 is the layer that reached the best results in both validation and testing sets.

In Figure 3.12, another way to represent the evolution of the cross-validation results obtained by layer residual 3 can be seen. It is noted that in the first iterations the optimizer explores parameters that do not obtain the best overall results. When new combinations are explored, the Bayesian optimizer tends to use improved combinations of hyperparameters and, therefore, the DIS tends to keep low in the last iterations.

For each layer, a different Bayesian optimization was run and the best combination of parameters found per layer are presented in Table 3.7. The same configuration of hyperparameters found during the cross-validation was applied in the testing set. Table 3.7 also contains the results obtained for each layer in the testing set. As it

Table 3.7: Results obtained with approach A for all layers with average pooling, where the cross-validation selects a unique set of hyperparameters to be used for all folds. The DIS values were calculated with Equation (3.6).

#	layer	hyperparameters				DIS validation	DIS test
		trees	max depth	threshold	voting window		
1	conv1	143	111	0.54	27	0.3846	0.3856
2	residual 1	175	94	0.49	28	0.3872	0.3542
3	residual 2	144	88	0.49	10	0.3758	0.3630
4	residual 3	152	148	0.51	28	0.3584	0.3500
5	residual 4	176	31	0.41	31	0.3853	0.3976
6	residual 5	170	92	0.46	22	0.3762	0.3724
7	residual 6	179	91	0.45	38	0.3634	0.3713
8	residual 7	92	88	0.45	28	0.3716	0.3845
9	residual 8	15	135	0.47	30	0.493	0.4831
10	residual 9	82	10	0.42	18	0.4887	0.5064
11	residual 10	94	10	0.42	27	0.4617	0.4757
12	residual 11	113	15	0.42	19	0.4656	0.4893
13	residual 12	169	17	0.41	20	0.4792	0.5021
14	residual 13	142	25	0.44	15	0.4768	0.4895
15	residual 14	89	29	0.46	14	0.509	0.5144
16	residual 15	150	49	0.49	11	0.4965	0.5216
17	residual 16	83	33	0.47	14	0.5074	0.5280

can be noted, the results in validation and testing sets of the 4th layer (residual 3) surpass all the other layers, with a DIS of 0.3584 and 0.35 respectively. The last layer (residual 16) obtained the worst result with a DIS of 0.5280 in the test set.

A graphical comparison between the validation and testing results of all layers is shown in the Figure 3.13. The limits highlighted by the blue shadow represent the standard deviation obtained in the validation set. The testing results are within the standard deviation limits. The standard deviation presented in Figure 3.13 was calculated among the DIS obtained by the 72 validation sets.

3.7.2 Results with Approach B

Considering approach B, the results were evaluated for the layer which obtained the best results with the average pooling: the residual 3. The hyperparameters found for each fold were obtaining training random forest classifiers using the scheme shown in Figure 3.10 and are shown in Table 3.8.

In this approach, the Bayesian optimizer was run independently for each fold and, therefore, different hyperparameters were found for each fold. Thus, for each type of object left in the scene, a specific classifier was trained. The number of trees is the parameter that presented the highest variation among all folds. By computing the DIS_{overall} using Equation 3.6, which gathers the classification results of all videos

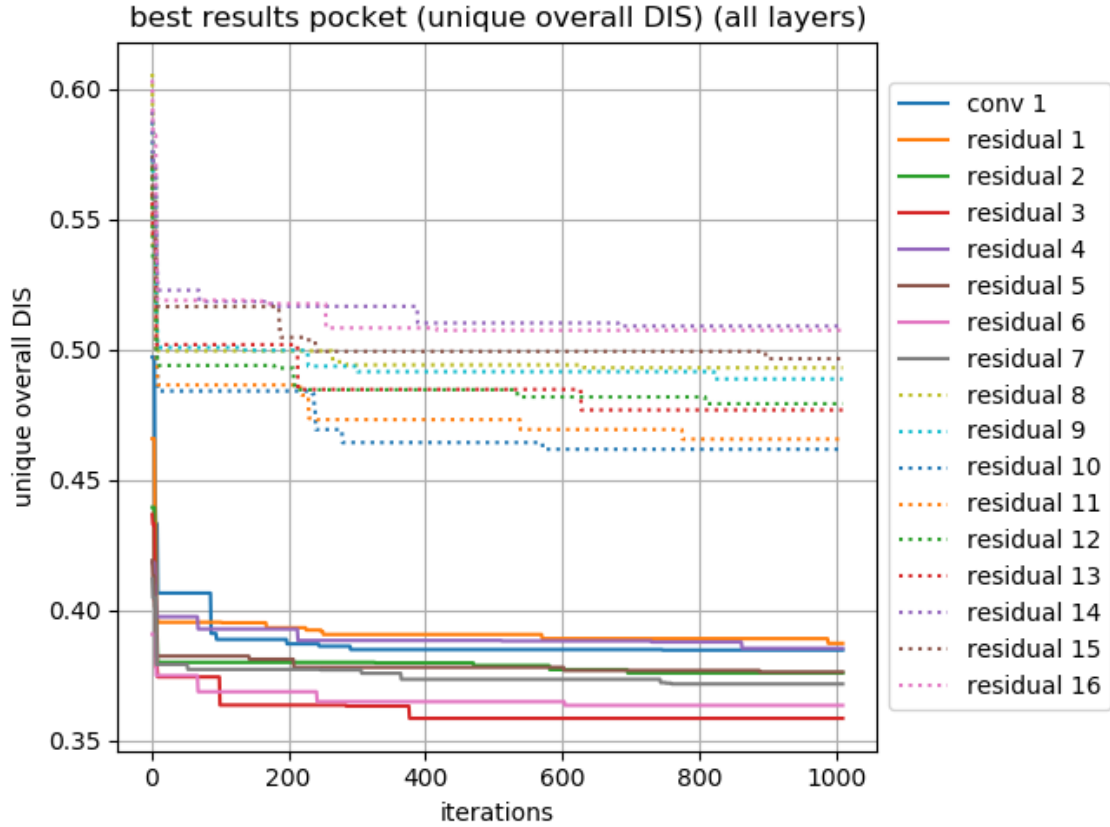


Figure 3.11: Best results obtained by the Bayesian optimization during the cross-validation process for all layers. The gap existing between $\text{DIS}=0.39$ (residual 1 layer) and $\text{DIS}=0.46$ (residual 9 layer), in the last iteration, makes evident the difference of quality of features generated by shallower and deeper layers. The best results (lowest DIS) are obtained by shallower layers.

Table 3.8: Hyperparameters obtained with approach B for the best layer (residual 3), with features reduced with average pooling. As the cross-validation was run independently for each fold, individual hyperparameters are obtained for each fold.

fold	hyperparameters			
	trees	max depth	threshold	voting window
$F_{\text{black backpack}}$	17	140	0.41	12
$F_{\text{black coat}}$	145	145	0.45	10
$F_{\text{brown box}}$	30	134	0.53	18
$F_{\text{camera box}}$	50	128	0.52	38
$F_{\text{dark-blue box}}$	40	130	0.42	25
$F_{\text{pink bottle}}$	52	125	0.58	1
F_{shoe}	35	99	0.43	19
F_{towel}	55	141	0.52	22
$F_{\text{white jar}}$	135	148	0.57	13

in a single metric, the $\text{DIS}_{\text{overall}}$ in the testing set is 0.3795.

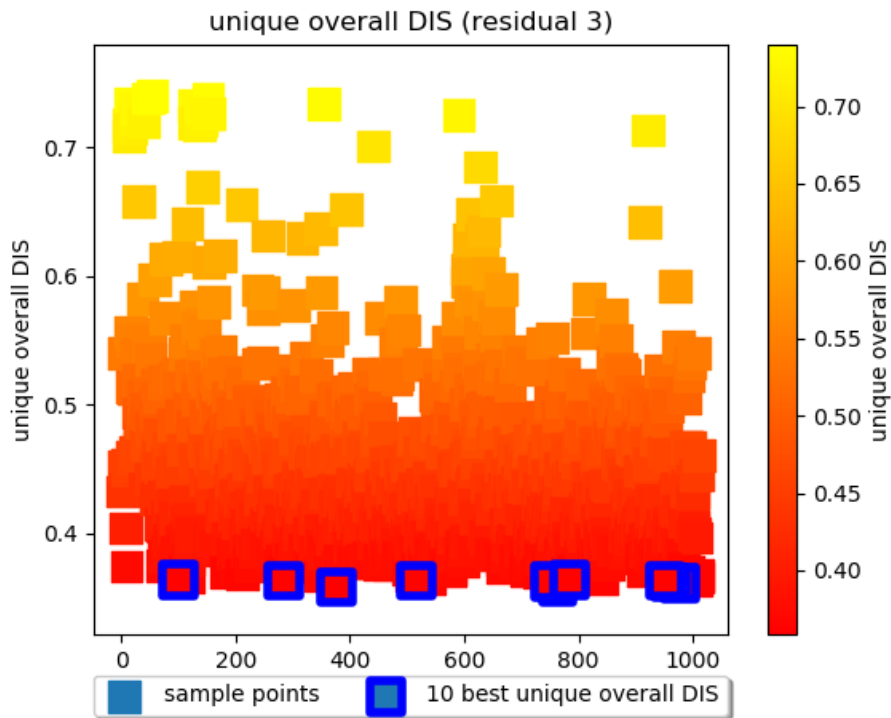


Figure 3.12: DIS values obtained along the iterations during the cross-validation of the 4th layer (residual 3) in approach A. Each square represents the DIS obtained by a random forest classifier with a different set of hyperparameters. The best 10 results are outlined in blue.

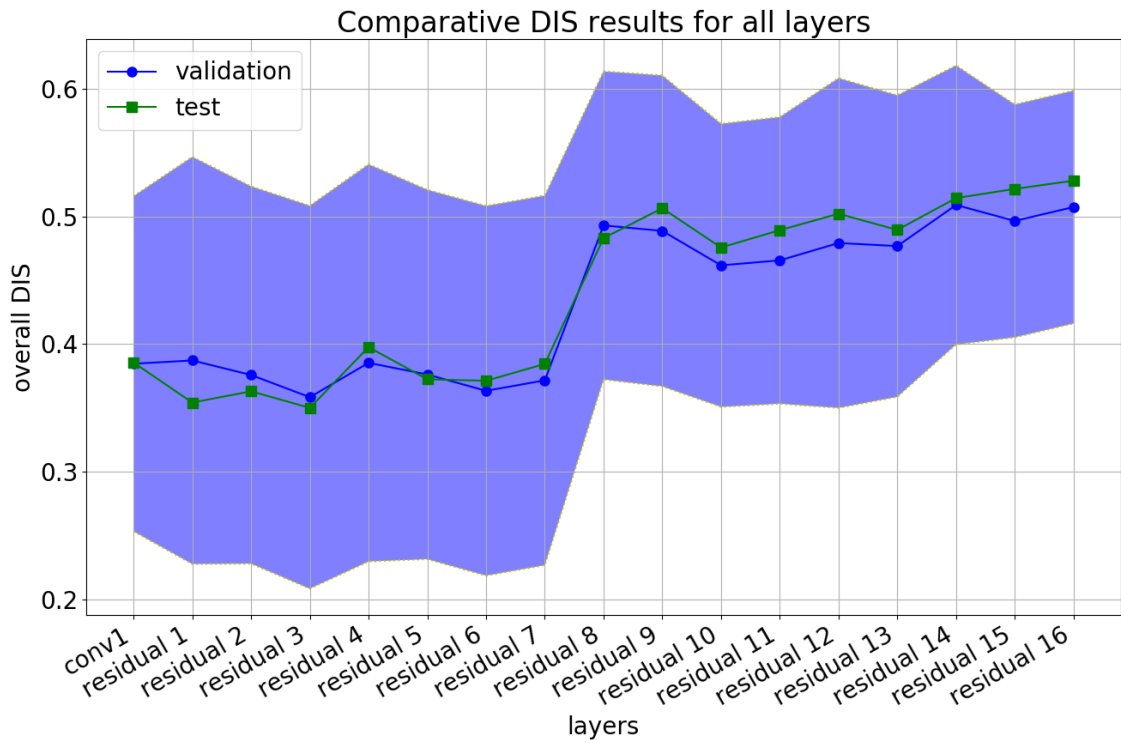


Figure 3.13: Results obtained by each layer in the validation and testing sets. The blue region represents the standard deviation among the DIS obtained by the 72 validation sets.

3.8 Conclusions

This chapter presented the methodology and results obtained by fine tuning a random forest classifier able to distinguish anomalous frames of the VDAO database. The baseline work proposed by [2] was modified in the following parts:

- Replace the DTW algorithm used to align the frames by a new one, able to run in real time. With the proposed algorithm, we eliminate the need to wait for the testing video to be fully acquired.
- A Bayesian optimizer was used to find the best combination of four hyperparameters (amount of trees, max depth of trees, threshold and voting window).
- The best set of hyperparameters to produce a final classifier to be applied in the testing sets was chosen through two cross-validation approaches. Approach A obtains a unique set of hyperparameters for all classifiers and are applied in the testing sets of all folds, while approach B finds one set of hyperparameters per fold.

By the obtained results, we concluded that:

- The application of a simpler technique to align the frames temporally, allowing real-time classification of the frames, did not compromise the results.
- Using a random forest classifier with Resnet-50 features yields a powerful classification method well adapted to detect anomalies in the VDAO database.
- The Bayesian optimizer does not require several adjustments to run and proved to be a useful tool to find the best hyperparameters for our model.
- The average pooling operation suggested by [2] obtained better results than a max pooling operation in 15 layers. For that reason the average operation was used in the pooling module of our proposed pipeline.
- Even though approach A is less burdensome to run (as only one cross-validation process is required for all folds), there is a data leaking between training and testing sets.
- Approach A produced better results than approach B, which was not affected by data leaking between training and testing sets. Being $DIS_{\text{overall}} = 0.35$ the result obtained by approach A and $DIS_{\text{overall}} = 0.3795$ the result of approach B.

- Shallower layers obtain the best results in validation and in test sets. This could be explained due to the fact that features produced by shallower layers are related to high level features (such as borders and salient edges), and the channel-wise subtraction of the reference and target feature maps accentuates the spatial differences.
- The 4th layer (residual 3) obtained the lowest DIS, performing better than all other layers. The same conclusion was observed by [2].
- The results obtained by fine tuning the model surpass the results obtained by the previous work [2], modifying the DIS in the testing set from 0.36 to 0.35.
- The experiments described in this chapter allowed a better comprehension and performance of the proposed system.

Besides the robustness of the random forest classifier and the hyperparameter tuning approach, the results were not significantly improved in comparison to the work in [2]. The aggressive spatial reduction of the features with the average pooling suggested that a better selection of the features might improve the performance of the classifier. In the next chapter, a feature selection technique was added into the pipeline replacing the average pooling used in the current chapter.

Chapter 4

Deep Features Selection with PCA and Classification with Random Forest

As described in the previous chapter, the pooling layer suggested by [2] reduces 99% of the features output by the Resnet-50 (see Table 3.2). In the current chapter, the pipeline presented in Chapter 3 is modified. The pooling layer is removed and two new blocks are introduced. The feature reduction process is now performed with principal component analysis (PCA), and the criterion used to select the features is based on their variances, also referred to as *energy criterion*. Due to the computational complexity, different types of PCA are computed and their results are compared.

The pipeline in Figure 4.1 illustrates the classification process of the target frames with the new approach. Similarly to the previous pipeline, the aligned frames pass through the Resnet-50, and their output features are obtained. Instead of reducing both feature tensors with the pooling operations, they are now subtracted before being reduced. The PCA, which is represented by two processes, transforms the features into a new basis composed by the principal components (PC), and the features with the highest variances are selected. Finally, the random forest classifier performs the anomaly detection task based on the selected features.

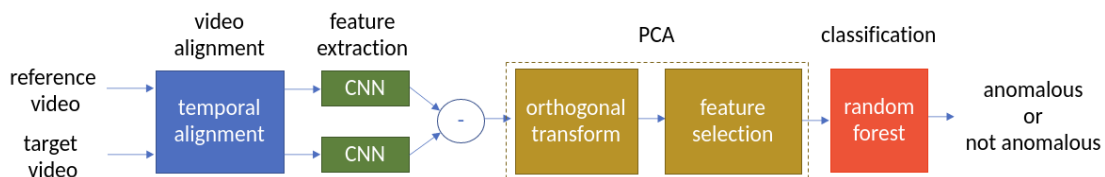


Figure 4.1: Pipeline proposed in this chapter using PCA to transform the features into a new basis composed by the principal components (PC), and a selection criterion determines the features used by the classifier.

Section 4.1 briefly reviews relevant works applying PCA to reduce dimensionality and describes three PCA algorithms, that we refer to as: the traditional PCA, the separable PCA, and the incremental PCA. The subsequent Section 4.2 includes experimental results using PCA to transform and select deep features, which are used to train the classifier. Finally, in Section 4.3, the results are compiled and conclusions are presented.

4.1 Feature Selection with PCA

Feature selection techniques have been widely applied to different tasks involving machine learning. Selecting a subset of relevant features is important in cases where high dimensional features are available. The three main problems that feature selection helps solving are pointed by [105]: the curse of dimensionality, overfitting, and data storage. The former refers to the phenomenon of the data becoming sparse as the dimension of data is increased, thus affecting the effectiveness of algorithms designed for low dimensional data. In such cases, although data is represented in high dimensions, the relevant information needed by the learning model is concentrated in only a few dimensions. This may also lead to the second mentioned problem, overfitting, which frequently occurs in models with more parameters than can be justified by the data, having their performance affected in unseen data. The latter issue involves the memory required to store high dimensional data and the computational costs for data processing, which can be prohibitive specially in video related applications.

Algorithms dealing with high dimensional data benefit from feature selection techniques, whose goals include reducing computational time, building simpler and understandable data, and describing the input data in an efficient way, so more comprehensive and lighter models can be used [105, 106].

PCA is a powerful tool to reduce the dimensionality of data, being present in areas such as image compression [107], face recognition [108, 109], data visualization [110, 111], data retrieval [112, 113], etc. The popular works [8, 114] were the first published works to use the Karhunen-Loeve transform to represent human faces efficiently through PCA. The eigenvectors, obtained from the covariance matrix generated from a set of images containing human faces, are used to represent the set of human faces as ghost-like images, called eigenfaces. Each image from the dataset can then be retrieved as a linear combination of the eigenvectors. The term eigenface first appeared in the work in [115] and was used to represent the eigenvectors, which form the basis of all face images in the set. By measuring the distance between the projected face images of two distinct individuals into the eigenfaces, the method is able to indicate if both faces belong to the same individual.

In the medical field, data is usually represented by multi-spectral images in a high dimensional space. The work in [7] applied PCA in structural magnetic resonance neuro images to reduce the dimensionality of the samples, which are used by an SVM classifier to identify Alzheimer’s disease in patients. The work in [116] used PCA to reduce the dimensionality of deep features obtained from computed tomography (CT) images of lungs before classification. By selecting only 3.3% of the features, PCA was able to reduce the training time of the model and reached 97.92% accuracy, a little less than the 98.74% of accuracy if all features were used. To detect brain tumor in magnetic resonance images, the works [117, 118] used PCA for feature selection. In [117], PCA was also able to separate lesion areas from brain images, and the remaining areas were classified into one of seven types of brain diseases, reaching an accuracy of 96%.

Hyperspectral images aggregate spatial and spectral information in different wavelengths, being widely applied in satellite images with applications in agriculture, defense, topography, botany, etc. Such images involve a large number of features, which are usually selected before being processed. The work in [119] used PCA to reduce the dimensionality of hyperspectral data, retaining only key features of aerial images. To achieve real-time performance, the work in [120] used an incremental PCA to reduce bands of hyperspectral images, reducing the processing time by about 17%. In [121], three different datasets were used to incorporate spatial and spectral information using PCA, reducing the amount of redundant information captured in different wavelengths.

Extracting features from images with deep CNNs also produces high dimensional features, which increases memory usage, as well as training and inference time. The work in [122] investigates the application of PCA to deep features, measuring the memory consumption and accuracy in image classification problems with SVM. In their experiments, the PCA algorithm not only reduced the dimension of image features and computational time, but also improved the classification accuracy.

Motivated by the success of the vast amount of works using PCA, we will explore the application of PCA in the deep features output by the 4th layer of the Resnet-50, which are used to classify the VDAO frames. Before presenting the application of PCA in the context of this work, a brief explanation of the traditional PCA algorithm is provided, and the complexity involved in such method is discussed. Two alternative approaches to compute the PCA transform are presented afterwards: the separable PCA and the iterative PCA.

4.1.1 Traditional PCA Transform

The goal of PCA is to identify the most meaningful basis to represent data. For that, a transformation matrix \mathbf{P} is computed. In the traditional PCA transformation, also referred to as Karhunen-Loeve transform (KLT) and holistic PCA [109], an $m \times n$ image \mathbf{I} is first changed to a one-dimensional vector $\mathbf{x} \in \mathbb{R}^{d \times 1}$, where $d = m \times n$. The set of N samples $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$ is used to calculate the covariance matrix or scatter matrix \mathbf{S} , which is given by

$$\mathbf{S} = \frac{1}{N} \sum_{i=1}^N (\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T, \quad (4.1)$$

where $\bar{\mathbf{x}}$ is the mean vector of all N images.

Each image \mathbf{I} contains a total of d pixels and the covariance matrix $\mathbf{S} \in \mathbb{R}^{d \times d}$ is a symmetric positive semi-definite matrix. Thus, its eigenvalues $\{\lambda_1, \lambda_2, \dots, \lambda_d\}$ are all real and non-negative, and its eigenvectors $\{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_d\}$ are orthogonal.

The set of the eigenvectors of matrix \mathbf{S} forms the new basis, where the samples will be projected to. The higher the eigenvalue, the higher the variance of the data is along the direction of its associated eigenvector. Therefore, the eigenvalues are ordered in such a way that $\{\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_d\}$, so the variance of the transformed data is greater in the \mathbf{v}_i direction than in \mathbf{v}_{i+1} direction.

The transformation of \mathbf{X} into \mathbf{Y} by matrix \mathbf{P} is represented by

$$\mathbf{Y} = \mathbf{P}\mathbf{X}. \quad (4.2)$$

The PCA algorithm is implemented when the transform matrix \mathbf{P} is composed column-wise by the ordered unitary eigenvectors $\{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_d\}$, which in this case are also referred to as the principal components (PCs).

As the basis vectors are ordered by their eigenvalues, each successive dimension of the resulting matrix \mathbf{Y} , represented by its rows, is ranked according to variance. So, instead of representing the transformed data with all d dimensions (all rows of \mathbf{Y}), it is possible to leave only the dimensions of the data with the highest variance. The cumulative variance, also referred to as energy e , is used to eliminate the dimensions of the data, whose energy are below a certain energy threshold, that is

$$e = \sum_{i=1}^d \lambda_i. \quad (4.3)$$

In summary, PCA is a transformation of the data \mathbf{X} by a linear combination of its basis vectors. The transformation is performed by matrix \mathbf{P} , made of the ordered unitary eigenvectors of the covariance matrix \mathbf{S} . Geometrically, \mathbf{P} can be seen as a rotation and scaling transformation matrix. The transformed data can be expressed

by its first dimensions (first rows of \mathbf{Y}), where most of the data energy is contained.

Computational Complexity Involved in Computing the Traditional PCA Approach

Although PCA can be applied to all types of features, the computational complexity to compute its transformation matrix increases drastically as the dimension of the data [123] increases. In the case of our feature tensor with more than 3 million features as an example, the computation of the eigenvectors and eigenvalues of the covariance matrix would be extremely costly, hindering the usage of the traditional PCA approach. The work [124] reviews the computational complexity of different PCA approaches. Besides its computational complexity, another problem caused by applying PCA in high dimensional data is the overfitting of the projection, which occurs when the number of features is much greater than the number of samples ($t \gg N$) [123, 125].

As alternatives for the traditional PCA, different methods have been proposed. The work in [108] proposes the two-dimensional PCA (2DPCA), a method applied to two-dimensional matrices rather than to one-dimensional vectors, being computationally more efficient than the traditional PCA. Three works of the same author [109, 126, 127] present the bi-directional PCA (BDPCA), which assumes that the transform kernel of PCA is separable, being a generalization of the 2DPCA approach. The work in [128] proposes the NGLRAM, a non-iterative version of the generalized low-rank approximations of matrices (GLRAM) algorithm. An interesting comparison of the aforementioned works is presented in [123], where the authors consolidate the definition of alternative PCAs into the concept of separable PCA (SPCA), one of the methods used in our work, which is described in Subsection 4.1.2.

Other approaches suggest incremental and iterative ways to compute the PCA transform. The work in [129] is one of the first ones to propose an incremental computation of PCs, the algorithm called non-linear estimation by iterative partial least squares (NIPALS-PCA). But if many samples are a linear combination of other samples, NIPALS-PCA suffers from loss of orthogonality due to the errors accumulated along iterations. Iterative principal components estimation can also be formulated in a probabilistic way, as in Bayesian PCA (BPCA) [130]. BPCA uses the expectation maximization algorithm to compute the PCs, making assumptions on the input data, and considers that the estimation error obeys a normal distribution. The work in [131] presents the SVDimpute method, which was first proposed to estimate missing values in DNA sequences. It estimates the PCs through singular value decomposition in an iterative way. As SVDs can only be computed on complete matrices, the SVDimpute is robust to noisy and incomplete data. A comparative analysis of the performance of iterative PCA approaches is made in [132], which con-

cludes that iterative methods such as BPCA, SVDimpute, and NIPALS-PCA are less time-efficient due to their iterations. Such time-inefficient probabilistic approaches inspired implementations of iterative PCA algorithms in GPUs and dedicated hardware [133]. The candid covariance-free incremental PCA (CCIPCA) [134, 135] uses a sequence of samples to calculate very efficiently the transform matrix without estimating the covariance matrix. With this technique, it is possible to improve the transform matrix when more samples are available, being suited for real-time applications with high-dimensional inputs. The CCIPCA is explained in details in Subsection 4.1.3.

4.1.2 Separable PCA

In the scope of this work, for each sample (one aligned pair of reference and target frames), the 4th Resnet-50 layer outputs a 3-dimensional feature tensor with dimensions $90 \times 160 \times 256$. If the traditional PCA was applied, the 3-dimensional feature tensor would first have to be changed to a one-dimensional vector \mathbf{x}_i with more than 3,686,400 features, leading to the high-dimensional problems discussed previously.

An alternative to the conventional PCA is the separable PCA, which is an approximation that yields computational efficiency gains and allows the computation of the PCA in each dimension of the feature tensor separately. The work in [123] demonstrates that existing methods such as BD-PCA, 2DPCA and NGLRAM are either special cases or equivalent cases to the separable PCA.

To explain the separable PCA, let us consider one sample as the feature tensor \mathbf{x}_i represented by the 3-dimensional block shown in Figure 4.2.

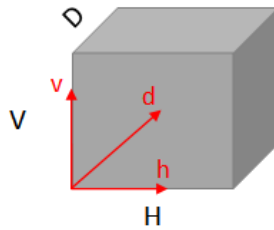


Figure 4.2: Feature tensor represented as a 3-dimensional block with three axes v , h and d .

In the current work, each pair of aligned reference and target frames produces a 3-dimensional feature block \mathbf{x}_i with three axes (v, h, d) , formed by the $V = 90$ columns, $H = 160$ rows and $D = 256$ channels. Each block \mathbf{x}_i can be represented with three types of one-dimensional vectors as in Figure 4.3. These three different representations of one-dimensional feature vectors of sample \mathbf{x}_i are here defined as:

- $\mathbf{x}_i^v(m, k)$: one-dimensional vector $\in \mathbb{R}^{V \times 1}$ in the direction v , located on the

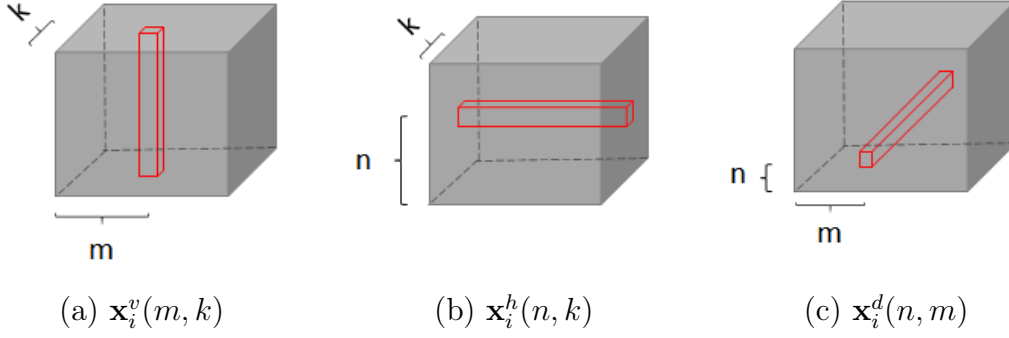


Figure 4.3: Tubes representing one-dimensional feature vectors in each direction. (a) feature vector $\mathbf{x}_i^v(m, k)$ in the direction v , (b) feature vector $\mathbf{x}_i^h(n, k)$ in the direction h , and (c) feature vector $\mathbf{x}_i^d(n, m)$ in the direction d .

m th column and k th channel of the feature tensor (see Figure 4.3 (a)),

- $\mathbf{x}_i^h(n, k)$: one-dimensional vector $\in \mathbb{R}^{H \times 1}$ in the direction h , located on the n th row and k th channel of the feature tensor (see Figure 4.3 (b)), and
- $\mathbf{x}_i^d(n, m)$: one-dimensional vector in the direction $d \in \mathbb{R}^{D \times 1}$ located on the n th row and m th column of the feature tensor (see Figure 4.3 (c)).

The separable PCA computes individual transformations applied along each axis of the feature tensor. Adopting the same approach used in [109], we can define exclusive covariance matrices for each axis of the feature block. The scattered matrices for each direction v , h and d , are shown in Equations (4.4), (4.5) and (4.6) respectively.

$$\mathbf{S}^v = \frac{1}{NHD} \sum_{i=1}^N \sum_{m=1}^H \sum_{k=1}^D (\mathbf{x}_i^v(m, k) - \bar{\mathbf{x}}^v)(\mathbf{x}_i^v(m, k) - \bar{\mathbf{x}}^v)^T, \quad (4.4)$$

where the product $H \times D$ is the total number of vertical vectors $\mathbf{x}_i^v(m, k)$ in each feature block \mathbf{x}_i as seen in Figure 4.3(a), and $\bar{\mathbf{x}}^v$ is the mean vertical vector computed with all N samples.

$$\mathbf{S}^h = \frac{1}{NVD} \sum_{i=1}^N \sum_{n=1}^V \sum_{k=1}^D (\mathbf{x}_i^h(n, k) - \bar{\mathbf{x}}^h)(\mathbf{x}_i^h(n, k) - \bar{\mathbf{x}}^h)^T, \quad (4.5)$$

where the product $V \times D$ is the total number of horizontal vectors $\mathbf{x}_i^h(n, k)$ in each feature block \mathbf{x}_i as seen in Figure 4.3(b), and $\bar{\mathbf{x}}^h$ is the mean horizontal vector computed with all N samples.

$$\mathbf{S}^d = \frac{1}{NVH} \sum_{i=1}^N \sum_{n=1}^V \sum_{m=1}^H (\mathbf{x}_i^d(n, m) - \bar{\mathbf{x}}^d)(\mathbf{x}_i^d(n, m) - \bar{\mathbf{x}}^d)^T, \quad (4.6)$$

where the product $V \times H$ is the total number of channel-wise vectors $\mathbf{x}_i^d(n, m)$ in

each feature block \mathbf{x}_i as seen in Figure 4.3(c), and $\bar{\mathbf{x}}^d$ is the mean channel-wise vector computed with all N samples.

If the traditional PCA was used, the covariance matrix \mathbf{S} would be a square $V \times V$ matrix, leading to an extremely high computational cost to obtain its eigenvectors and eigenvalues. As the dimensions of the covariance matrices \mathbf{S}^v , \mathbf{S}^h and \mathbf{S}^d are now $V \times V$, $H \times H$ and $D \times D$ respectively, the computational cost to calculate the eigenvalues and eigenvectors is reduced drastically.

As done in the traditional PCA, for each covariance matrix, a transformation matrix is obtained by computing its eigenvectors ordered by their respective eigenvalues. Thus, the transformation matrix in direction v is $\mathbf{P}^v \in \mathbb{R}^{V \times V}$, whose rows are the ordered eigenvalues of \mathbf{S}^v . The transformation matrix in direction h is $\mathbf{P}^h \in \mathbb{R}^{H \times H}$, whose rows are the ordered eigenvalues of \mathbf{S}^h . And the transformation matrix in direction d is $\mathbf{P}^d \in \mathbb{R}^{D \times D}$, whose rows are the ordered eigenvalues of \mathbf{S}^d .

Let us consider the sample to be transformed a 3-dimensional feature block $\mathbf{x}_i \in \mathbb{R}^{V \times H \times D}$. The first transformation \mathbf{P}^v is applied to each vector $\mathbf{x}_i^v(m, k)$ in direction v .

$$\mathbf{y}_i^v(m, k) = \mathbf{P}^v \mathbf{x}_i^v(m, k). \quad (4.7)$$

The resulting 3-dimensional block $\mathbf{y}_i \in \mathbb{R}^{V \times H \times D}$ will have its vectors $\mathbf{y}_i^h(n, k)$, in direction h , transformed by \mathbf{P}^h :

$$\mathbf{t}_i^h(n, k) = \mathbf{P}^h \mathbf{y}_i^h(n, k). \quad (4.8)$$

The resulting 3-dimensional block $\mathbf{t}_i \in \mathbb{R}^{V \times H \times D}$ will have its vectors $\mathbf{t}_i^d(n, m)$, in direction d , transformed by \mathbf{P}^d :

$$\mathbf{z}_i^h(n, k) = \mathbf{P}^d \mathbf{t}_i^d(n, m). \quad (4.9)$$

The 3-dimensional block $\mathbf{z}_i \in \mathbb{R}^{V \times H \times D}$ is the final transformed sample \mathbf{x}_i with three consecutive one-dimensional PCAs. To summarize the separable PCA process, the first PCA transforms \mathbf{x}_i considering its vectors in direction v , generating the 3-dimensional feature block \mathbf{y}_i . The second PCA transforms \mathbf{y}_i considering its vectors in direction h , producing the 3-dimensional feature block \mathbf{t}_i . The third PCA transforms \mathbf{t}_i considering its vectors in direction d , creating the final 3-dimensional feature block \mathbf{z}_i .

4.1.3 Incremental PCA

The popularization of PCA motivated researchers to find alternatives to compute the transformation matrix incrementally, preventing a new computation of the trans-

form matrix when new samples are available. The works in [134] and [135] propose the candid covariance-free incremental PCA (CCIPCA), which uses a sequence of samples to calculate the transform matrix without estimating the covariance matrix. With this technique, it is possible to improve the transform matrix when more samples are available, being suited for real-time applications with high-dimensional inputs.

The computation of the eigenvectors incrementally is done as follows. Suppose the vectors $\{\mathbf{x}_0, \mathbf{x}_1, \dots\}$ are acquired sequentially, and each \mathbf{x}_i is a d -dimensional vector. Let us assume that all \mathbf{x}_i vectors have zero mean, which can be calculated incrementally as new samples are available. By definition, an eigenvalue λ associated to its eigenvector \mathbf{t} of matrix \mathbf{A} satisfies

$$\lambda \mathbf{t} = \mathbf{A} \mathbf{t}. \quad (4.10)$$

By replacing matrix \mathbf{A} with the covariance definition from Equation (4.1), and replacing \mathbf{t} from Equation (4.10) with its estimate $\mathbf{t}(\mathbf{i})$ in the i th step, we obtain the following expression for $\mathbf{v}(n) = \lambda \mathbf{t}$:

$$\mathbf{v}(n) = \frac{1}{n} \sum_{i=1}^n \mathbf{x}_i \mathbf{x}_i^T \mathbf{t}(\mathbf{i}), \quad (4.11)$$

where $\mathbf{v}(n)$ is the estimate of \mathbf{v} in the n th step. With the estimate of \mathbf{v} , the eigenvector \mathbf{t} and eigenvalue λ can be computed since $\lambda = \|\mathbf{v}\|$ and $\mathbf{t} = \mathbf{v} / \|\mathbf{v}\|$. Considering $\mathbf{t} = \mathbf{v} / \|\mathbf{v}\|$, we may choose $\mathbf{t}(i)$ as $\mathbf{v}(i-1) / \|\mathbf{v}(i-1)\|$, resulting in the incremental expression:

$$\mathbf{v}(n) = \frac{1}{n} \sum_{i=1}^n \mathbf{x}_i \mathbf{x}_i^T \frac{\mathbf{v}(i-1)}{\|\mathbf{v}(i-1)\|} \quad (4.12)$$

The first direction of data spread is given by the first sample \mathbf{x}_1 , so $\mathbf{v}(0) = \mathbf{x}_1$. Equation (4.12) can be written in its recursive form, becoming:

$$\mathbf{v}(n) = \frac{n-1}{n} \mathbf{v}(n-1) + \frac{1}{n} \mathbf{x}_n \mathbf{x}_n^T \frac{\mathbf{v}(n-1)}{\|\mathbf{v}(n-1)\|}, \quad (4.13)$$

where $(n-1)/n$ is the weight for the last estimate and $1/n$ is the weight for a new incoming data. An amnesic parameter L could be added in (4.13) giving more weight to new data, so older data will eventually have less contribution, which becomes:

$$\mathbf{v}(n) = \frac{n-1-L}{n} \mathbf{v}(n-1) + \frac{1+L}{n} \mathbf{x}_n \mathbf{x}_n^T \frac{\mathbf{v}(n-1)}{\|\mathbf{v}(n-1)\|}. \quad (4.14)$$

The CCIPCA considers the computation of all k eigenvectors given an incoming sample \mathbf{x}_i , and defines $\mathbf{v}_j(i)$ as the estimation of the j th eigenvector given the i

incoming sample (or in the i th step). The full algorithm is presented in Algorithm 1.

Algorithm 1: Incremental PCA (CCIPCA) algorithm proposed in [134] and [135].

```

Data: Available samples
for  $i=1, 2, \dots$  do // loop through the indexes of the samples
     $\mathbf{x}_i(1) \leftarrow \mathbf{x}_i$ ;
    for  $j=1, 2, \dots$  do // loop through the indexes of the
        eigenvectors
            if  $j == i$  then
                 $\mathbf{v}_j(i) \leftarrow \mathbf{x}_i(j)$ ;
            end
            if  $j < i$  then
                //  $j$ th eigenvector with  $i$ th sample using Equation (4.14)
                 $\mathbf{v}_j(i) \leftarrow \frac{i-1-L}{i}\mathbf{v}_j(i-1) + \frac{1+L}{i}\mathbf{x}_i(j)\mathbf{x}_i(j)^T \frac{\mathbf{v}_j(i-1)}{\|\mathbf{v}_j(i-1)\|}$ ;
                // Update sample  $i$  value for the next eigenvector  $j$ 
                 $\mathbf{x}_i(j+1) \leftarrow \mathbf{x}_i(j) - \mathbf{x}_i(j)^T \frac{\mathbf{v}_j(i)}{\|\mathbf{v}_j(i)\|} \frac{\mathbf{v}_j(i)}{\|\mathbf{v}_j(i)\|}$ ;
            end
            if  $j > i$  then
                 $\mathbf{v}_j(i) \leftarrow 0$ ;
            end
        end
    end
end

```

The performance and accuracy tests reported in [134] and [135] show a fast convergence rate of the CCIPCA algorithm with low computational cost.

4.2 Experimental Results with PCA

As reviewed in the previous section, different variations of PCAs can be used to reduce computational burden. As described in Chapter 3, the best anomaly-detection results were obtained by features extracted from the 4th layer (*residual 3*) of the Resnet-50 backbone. Thus, in order to obtain a possible improvement of the DIS metric, the feature reduction techniques using PCA are applied in the deep features produced by the 4th layer.

The pipeline used in the experiments described here is illustrated in Figure 4.4. For each pair of aligned reference and target frames, a total of 3,686,400 features are output by Resnet-50's 4th layer distributed in a 3-dimensional feature tensor. Both reference and target tensors are subtracted, and the resulting feature tensor has its dimensions reduced with PCA. To keep a fair comparison to the previous pipeline, the same number of features produced by the pooling operation are selected (3,840 features). By that, our intention is to evaluate how the PCA transformation and

the variance energy criterion can improve the classification of the target frames, and if it is possible to achieve similar results with fewer features.

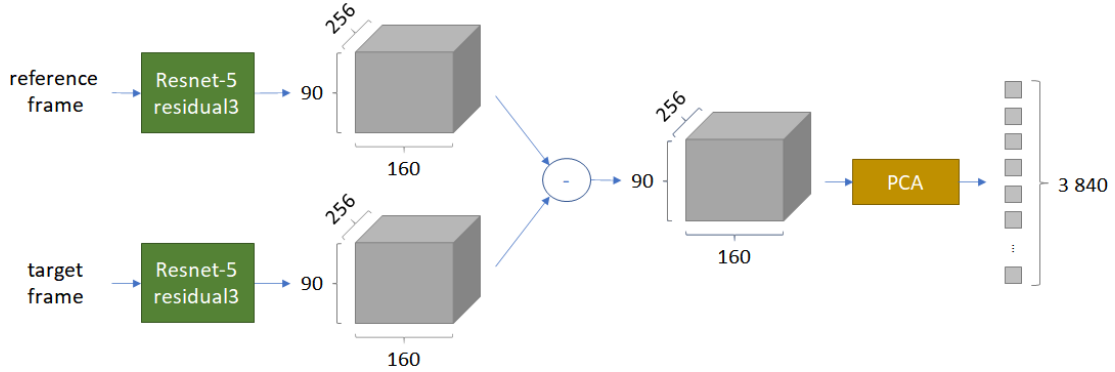


Figure 4.4: Feature selection scheme used to obtain features with the 4th convolutional layer of the Resnet-50 using PCA. Given a feature tensor with about 3 million features, the best 3,840 features are selected. This is the same number of features used by the classifier in the previous pipeline.

First, a separable PCA is applied to the 3-dimensional feature tensor considering all three directions. In an initial experiment, the same number of features used by the random forest classifier were selected using the energy criterion, and the hyperparameters of the classifier were chosen with the Bayesian optimization process. In a second experiment still using the separable PCA, in an attempt to decrease the number of features, the Bayesian optimization was also responsible by selecting the best number of PCs. Next, two other experiments were performed, one using traditional PCA to compute what we refer to as spatial PCA, and another experiment using the incremental PCA to compute the PCA in blocks. In all experiments, the bounds used by the Bayesian optimization to find the best hyperparameters were the same, being

- max depth: from 1 to 150,
- trees: from 1 to 210,
- threshold: from 0.4 to 0.6,
- voting window: from 1 to 150.

4.2.1 Experimental Results with Separable PCA

The features output by the layer 4 (residual 3rd layer) are first transformed with separable PCA and selected using the energy criterion. The two experiments covered in this subsection use the same methodology from Chapter 3, illustrated in the scheme of Figure 3.10: 72 random forest classifiers are trained with frames extracted

from videos of 5 objects. Each classifier is used to validate videos of 3 objects, which are the videos of the object that were neither used during training nor validation are used for testing. The validation results are used by a Bayesian optimization process to find the best hyperparameters (number of trees, max depth, threshold and voting window). The iterative process ran 1,000 times, with 10 first iterations used for random exploration.

In the first experiment, the features are transformed with separable PCA and the same number of features used by the pooling is selected based on the energy criteria. In the second experiment, the number of principal components was selected by the Bayesian optimization process, in an attempt to reduce the number of features.

Selecting the Same Number of Features as the Average Pooling

In this experiment, the fixed number of features with the highest energy (variance) were selected. In the previous chapter, the number of features was reduced to 3,840 with the average pooling, thus we decided to keep the same number of features, transforming them with PCA and using the 3,840 features with the highest energy. As there are 72 sets containing distinct training samples, the amount of energy contained in 3,840 features with the highest energy varies among the sets. In average, among all 72 sets, the 3,840 principal components concentrate 6.6% of the total energy with a standard deviation of 1.78%.

For each set, the transformation matrices \mathbf{P}^v , \mathbf{P}^h and \mathbf{P}^d are calculated using the training samples, and are used to transform the training and validation samples. For the testing phase, the transformation matrices are computed considering the samples of the training and validation sets, which are also used to train a new classifier with the hyperparameters chosen during the validation phase. Figure 4.5 shows the $\text{DIS}_{\text{overall}}$ obtained along the 1,000 iterations during training/validation.

For a better comparison with results from Chapter 3, where the average pooling operation was used, Table 4.1 presents the validation results with both approaches. Classifiers trained with features transformed and selected with separable PCA obtained better validation results in videos with all objects (lower DIS). The $\text{DIS}_{\text{overall}}$, which considers videos with all objects, also performed better. Table 4.2 compares the hyperparameters found by the Bayesian optimization process using the selection of features with separable PCA and the average pooling.

Results obtained in the testing phase are shown in Table 4.3. Differently from the validation, in the testing phase not all classifiers performed better with features obtained with separable PCA. Besides the significant improvements reached with some classes of objects, the final $\text{DIS}_{\text{overall}}$ result with separable PCA did not have a considerable deviation from the pooling approach.

So far, as the results with the average pooling were not significantly changed

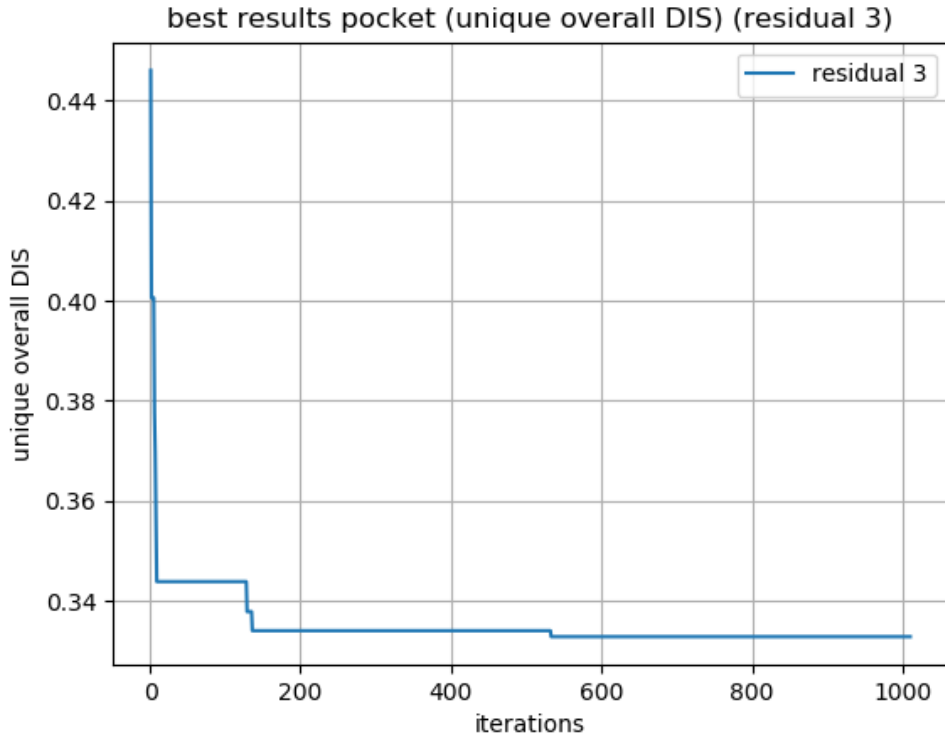


Figure 4.5: $\text{DIS}_{\text{overall}}$ results calculated with Equation (3.6). These results are obtained during training/validation phase with the 3,840 features with the highest energy transformed with the separable PCA.

Table 4.1: DIS values, calculated with Equation (3.6), for each object class obtained during validation with 3,840 features.

	$\text{DIS}_{\text{overall}}$ (average pooling)	$\text{DIS}_{\text{overall}}$ (separable PCA)
black backpack	0.3957	0.3878
black coat	0.3886	0.3595
brown box	0.4075	0.3434
camera box	0.3948	0.3777
dark-blue box	0.4424	0.3369
pink bottle	0.3941	0.3783
shoe	0.3999	0.3114
towel	0.3542	0.3493
white jar	0.3816	0.3275
$\text{DIS}_{\text{overall}}$	0.3584	0.3329

when the separable PCA was applied, a question is raised: can the number of features (3,840) be reduced, while achieving the same results? To answer this question, instead of performing a grid search with multiple experiments using different amounts of features, a guided search using the Bayesian optimizer is performed.

Table 4.2: Hyperparameters found by the Bayesian optimization during the training/validation with 3,840 features selected with average pooling and separable PCA.

	iteration	trees	max depth	threshold	voting window
average pooling	377	152	148	0.51	28
separable PCA	533	168	137	0.53	18

Table 4.3: DIS values, calculated with Equation (3.6), obtained in the testing sets with 3,840 features selected with average pooling and separable PCA.

	DIS_{overall} (average pooling)	DIS_{overall} (separable PCA)
black backpack	0.4003	0.5728
black coat	0.3169	0.3379
brown box	0.3036	0.3889
camera box	0.4025	0.3767
dark-blue box	0.2145	0.1903
pink bottle	0.4869	0.4766
shoe	0.2041	0.2115
towel	0.7134	0.3295
white jar	0.4677	0.4011
DIS_{overall}	0.3500	0.3527

Selecting the Principal Components Used to Transform Features with Separable PCA

In a second experiment, besides selecting random forest training hyperparameters, we also let the Bayesian optimizer select the amount of principal components within the interval of 1 and 3,840 principal components. By that, we want to evaluate if a significant reduction of features can achieve, at least, the same results. Figure 4.6 shows the DIS_{overall} obtained along the training/validation iterations.

The results and hyperparameters obtained with separable PCA can be seen in Table 4.4. The validation and testing DIS results practically did not change, but a significant reduction on the number of features was achieved. Both separable PCA approaches had slightly better results in the validation set in comparison with the average pooling approach. But, essentially, the results in the testing set are about the same. The advantage of using the separable PCA is that the same results can be obtained, but with 25% less features. Considering all 72 sets, the average amount of energy contained in the 2,850 features is practically the same as in the 3,840 features, being 6.45% with standard deviation equals to 1.93%.

In these experiments, the Bayesian optimization process takes too long to run with the 72 sets (about 21 days running in a machine Intel(R) Xeon(R) Gold 5120 CPU @ 2.20GHz with 125 GB of RAM), and the set of hyperparameters with the most significant gains in the DIS metric are achieved in the first iterations of the total

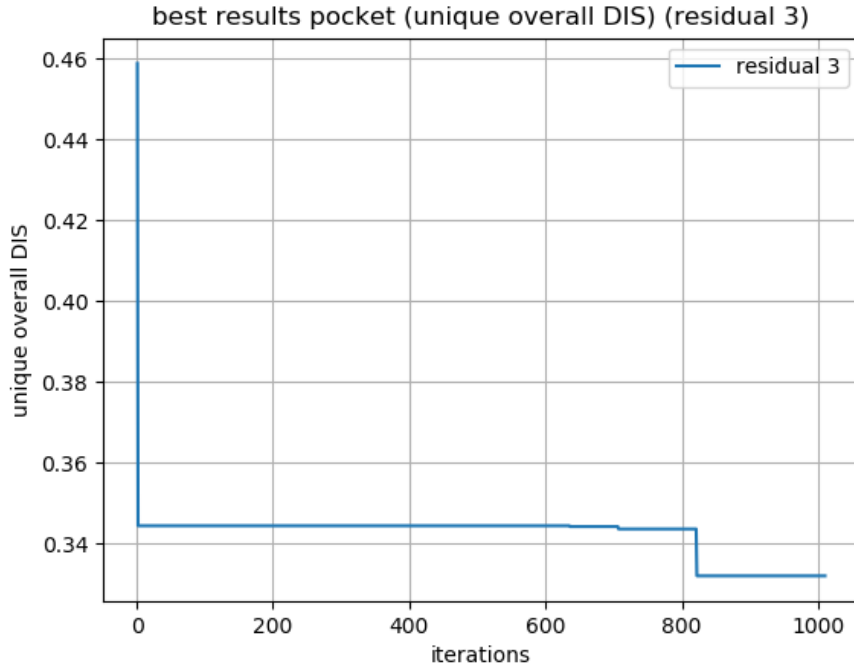


Figure 4.6: DIS values, calculated with Equation (3.6), obtained during training/validation. Features were transformed with separable PCA and the Bayesian optimization process selected the amount of principal components.

Table 4.4: Comparative DIS results calculated with Equation (3.6), and hyperparameters used to train random forest classifiers with features transformed with separable PCA and average pooling operation.

feature reduction approach	features	trees	max depth	threshold	voting window	DIS validation	DIS testing
average pooling	3840	30	93	0.53	25	0.3584	0.3500
separable PCA fixed PC	3840	168	137	0.53	18	0.3328	0.3527
separable PCA selecting PC	2850	152	148	0.51	28	0.3320	0.3587

1,000 used. That being said, the further experiments will run with 300 iterations considering only two classes of objects for testing (shoe and dark-blue box). These two classes were chosen among the others, because the results of these classes using the separable PCA approach deviate a lot in comparison with the average pooling approach, as it can be seen in Table 4.1. Therefore, if a substantial improvement is noted in the metrics of these two classes, we can extend to the other classes. For a matter of comparison, we repeated the tests in this section with these new considerations.

Under this new configuration, to perform the test with videos containing the shoe, a classifier was trained with videos of all objects, except videos containing the

shoe. The Bayesian optimizer was run for 300 iterations (being 10 iterations used for exploration), so the best hyperparameters used by the classifier were selected. In the end, the trained classifier with the best hyperparameters was used to test the videos containing the object shoe. A similar procedure was done to test the videos containing the object dark-blue box.

The object shoe obtained a DIS equal to 0.1879 with features reduced with the average pooling operation, and 0.1862 when features were selected using the separable PCA. The object dark-blue box obtained a DIS equals to 0.0645 with features reduced with the average pooling operation, and 0.0703 using features selected with the separable PCA. In these new experiments, the classifiers performed better than in the previous experiments. This might be explained because instead of using videos of 5 objects to train, we are now using videos of 9 classes of objects, making the classifier more robust. As noted before, the classifiers performed similarly with features reduced with average pooling and with separable PCA.

4.2.2 Experimental Results with Spatial PCA

In Resnet-50, the 4th layer (residual 3) outputs a 3-dimensional feature tensor, which is formed by 256 2-dimensional feature maps. In this case, a PCA transform could be applied in each 2-dimensional feature map using the traditional PCA approach.

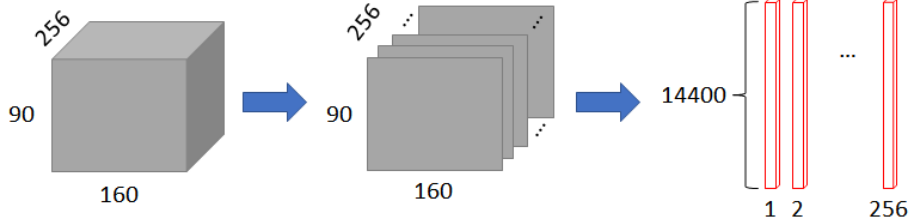


Figure 4.7: A 3-dimensional feature tensor represented as 256 2-dimensional feature maps and 256 one-dimensional vectors.

By changing each 2-dimensional feature map $\in \mathbb{R}^{90 \times 160}$ into a one-dimensional vector $\mathbf{x}_i \in \mathbb{R}^{14400 \times 1}$, each pair of aligned reference and target frames can be represented by the set $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{256}\}$ as illustrated in Figure 4.7. The 256 one-dimensional feature vectors of N training samples could be stacked together producing $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{N,256}\}$, which can be plugged into Equation (4.1) to compute the traditional PCA transformation. The computation of PCA using the traditional PCA method is now feasible, since the covariance matrix is 14400×14400 .

Each 2-dimensional feature map $\in \mathbb{R}^{90 \times 160}$ is now transformed with the traditional PCA, and for each feature map, we select the 15 features with the highest energy. This way, for each sample (feature tensor), the same number of features used in the previous approaches (3,840) are used to train the classifier.

As done in the last experiment with average pooling, we evaluated the performance of the spatial PCA with videos of two objects only: shoe and dark-blue box. With features reduced with the spatial PCA, the DIS value for the class shoe was 0.2361 and for the class dark-blue box it was 0.1033. For both objects, the performance of the classifiers using the spatial PCA was inferior than the one obtained with the average pooling and separable PCA approaches, which refrained us from evaluating this scheme for the other object classes. With average pooling, the objects shoe and dark-blue box obtained the DIS values 0.1879 and 0.0645 respectively. When separable PCA was applied, the objects shoe and dark-blue box obtained the DIS values 0.1862 and 0.0703 respectively.

4.2.3 Experimental Results with Incremental PCA in Blocks

Inspired by the idea of the pooling operation, which reduces the feature maps spatially, the PCA transform could be applied in blocks to output the same spatial dimension produced by the pooling. In the image domain, the average pooling takes the average value considering a region of 30×32 , with a total of 960 pixels. A PCA is now computed using the features in regions 30×32 in each channel and, after transforming them to the new basis, only the feature with the most energy is selected. Figure 4.8 illustrates the 3-dimensional feature tensor 90×160 with $D = 256$ channels. By changing each $30 \times 32 \times 1$ block into one-dimensional vector $\mathbf{x}_i \in \mathbb{R}^{960 \times 1}$, each pair of aligned reference and target frames results into $15D = 3,840$ vectors. The 3,840 one-dimensional feature vectors of N training samples could be stacked together producing $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{3840N}\}$, and used to calculate the PCA transform with the traditional PCA approach. As each sub-block $30 \times 32 \times 1$ transformed with PCA results into one feature, for a 3-dimensional block $90 \times 160 \times 256$, a total of 3,840 features are selected, which is the same number of features produced by the pooling operation.

In this approach, the incremental PCA was applied to compute the PCA in blocks. The results obtained by a classifier trained with samples transformed with PCA in blocks were not better than the ones from previous experiments. The tests for the class shoe resulted in a DIS equal to 0.2228, and with the class dark-blue box are 0.0983 against 0.1879 and 0.0645, respectively, obtained with the average pooling scheme.

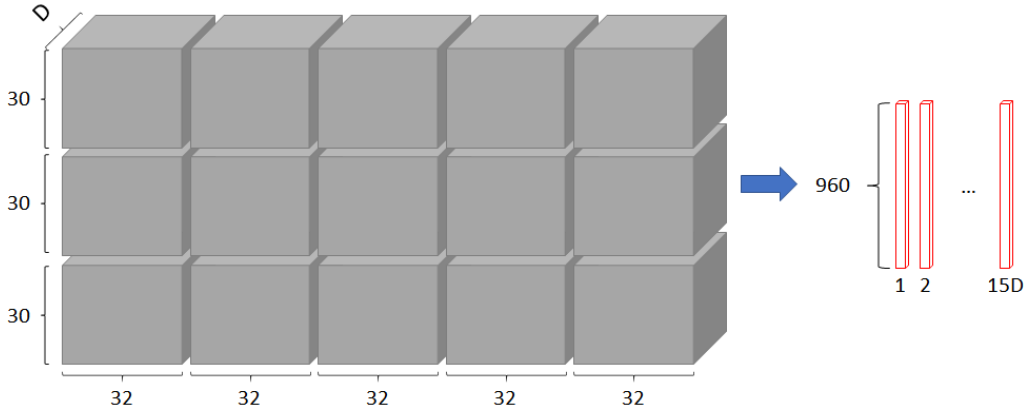


Figure 4.8: Representation of the feature tensor $V \times H \times D$ into 15 smaller blocks, which are represented by $15D$ one-dimensional vectors.

4.3 Final Results and Conclusions

In this chapter, the average pooling was replaced by a feature selection approach using different PCA approaches. The criterion to select features was performed based on their variances. Three techniques, namely traditional PCA, separable PCA and incremental PCA, were applied to compute the transformation matrix used by PCA in different configurations of the 3-dimensional feature tensor. The metric used to compare these experiments was the DIS value, the same metric used in previous chapters and computed with Equation (3.6).

To overcome the long computational processing time needed to train the classifiers using the Bayesian optimization for all classes of objects, only the experiments using the separable PCA were performed considering all classes of objects. An extra experiment was executed with all classes of objects, letting the Bayesian optimization process choose the amount of principal components.

In the experiment aiming to reduce the amount of PCs (described in Subsection 4.2.1), the Bayesian optimization process was also used to optimize the amount of PCs, whose bounds used in the searching process were defined as being from 1 to 3,840.

Our results show that the feature selection criterion applied to features transformed with separable PCA obtained similar results when average pooling was used with the same number of features (3,840). Even though the validation results were slightly better when separable PCA was applied, the testing results practically did not change. The energy contained in the 3,840 features represents 6.6% of the total energy in the original 3.6 million features. In another experiment the Bayesian optimization process was used to select the number of principal components, reducing the amount of features to 2,850, which keeps 6.45% of the total energy from the original feature tensor. The drop of 25% in the number of features does not

affect the DIS results in both validation and testing sets. Therefore, the Bayesian optimization proved to be an effective tool that led to a reduction in the number of features, while keeping the same results.

In another group of experiments, only two classes of objects were used for testing. Table 4.5 presents the results for each approach.

Table 4.5: Testing results with DIS values calculated with Equation (3.6) for two classes of objects considering 3,840 features.

	shoe	dark-blue box
average pooling	0.1879	0.0645
separable PCA	0.1862	0.0703
spatial PCA	0.2361	0.1033
incremental PCA in blocks	0.2228	0.0983

Although features transformed and reduced with separable PCA obtained nearly the same results of features reduced with pooling, different PCA approaches were not able to provide better results than the ones obtained with the average pooling. Thus, our experiments show that features transformed with PCA and selected using the variation criteria did not provide any positive improvement in the DIS metric in comparison with a non-selective criterion, such as the average pooling operation.

These results suggest that the pre-trained Resnet-50 already outputs features decorrelated and discriminative enough in a way that PCA is not able to improve them for our classification results. In this work, PCA transformation does not worsen the capability of the features to classify the VDAO frames. Also, a possible explanation for the invariability of the results obtained with high energy features criteria, is the fact that low energy features are in fact important for this classification task. But a confirmation of this fact demands deeper investigations that could be performed in a future work. The lessons learned from this chapter motivated us to explore a new approach, without PCA. In this sense, a new approach using an end-to-end training network is proposed and presented in the next chapter.

Chapter 5

An End-to-End Differentiable Anomaly Detection Pipeline

The two previous chapters investigated pipelines for anomaly detection in videos based on the extraction of deep features from the reference and target frames using twin networks. In both pipelines, essentially, a random forest classifier receives as input the difference between features from the two networks and decides if a frame contains an abandoned object. A natural path to design a pipeline with improved performance would be to replace, as input to the random forest classifier, the difference between the features of the twin networks by their concatenation. Unfortunately, given the size of the training set provided by the VDAO dataset, the feature concatenation generates large input samples. As the random forest classifier requires all training samples simultaneously to be constructed, large input samples prevent its application. Even though the difference of features of the twin networks reduces the number of features by half, it is still necessary to apply feature reduction techniques to make the random forest application feasible.

The experiments and results presented in Chapters 3 and 4 confirmed that such approaches limit the classification performance. Since the mere use of the difference between features of the twin networks disregards a great deal of valuable information that could be used to improve the detection performance, a way to exploit richer information provided by the full feature tensors without leading to a prohibitively large number of parameters would be welcome.

In this chapter, a new pipeline is proposed to deal with this issue. It is illustrated in Figure 5.1. It consists of modules separated according to their functions: video alignment, feature extraction, feature processing, and classification. Initially, a geometrical alignment is performed on each pair of reference and target frames previously aligned with the temporal alignment present in Chapter 3, Section 3.1. Given the pair of re-aligned frames, their features are individually extracted from the 4th convolutional layer of the Resnet-50 model. A dissimilarity module (DM)

aims to exploit richer information provided by the feature tensors from the twin networks producing a single binary image. The temporal consistency module (TCM) is responsible to remove possible false positive regions in consecutive frames while maintaining the true positive ones. To remove persistent remaining false positive regions in each frame, as well as filling false negative holes, a mathematical morphology module (MM) is introduced to simulate morphological open and close operations using a circular structuring element with a learnable radius. An innovation proposed in this thesis is that the MM is differentiable with respect to the radii of the opening and closing structuring elements, and therefore become learnable parameters using backpropagation. In the end, based on the number of positive pixels in the resulting image, the classification module (CM) predicts the class of the input target frame (anomalous or not anomalous). With the exception of the alignment stage, that performs a sort of pre-processing of the data, the whole structure is differentiable in the parameters of the proposed pipeline and allows end-to-end training using the backpropagation algorithm.

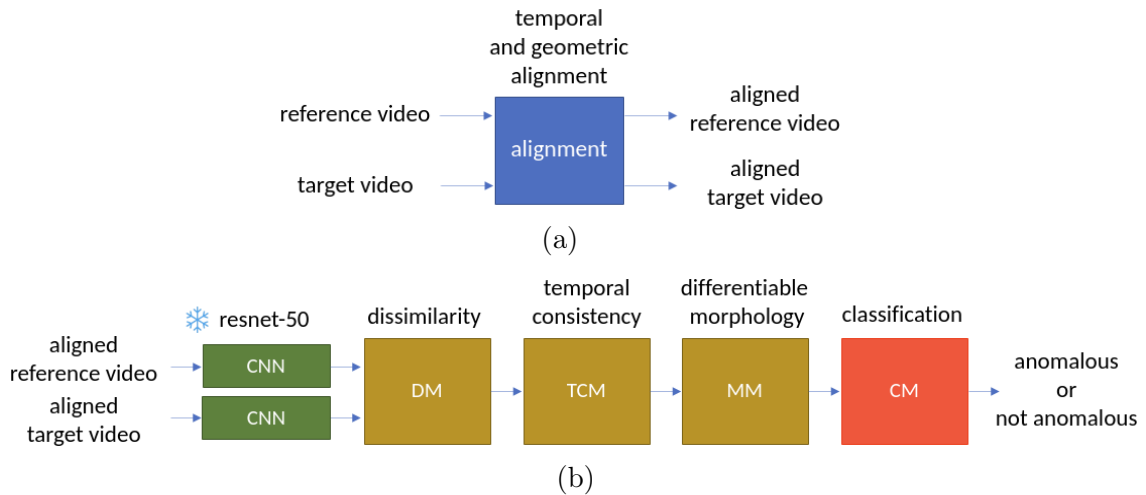


Figure 5.1: Pipeline proposed in this chapter. A pre-processing module is introduced to align temporally and geometrically a pair of reference and target videos (a). Three modules (DM, TCM, and MM) are responsible for processing the features extracted from Resnet-50 (frozen) in order to feed a classification module (CM) as illustrated in (b).

The design of each module will be specified in subsequent sections. A sample with results obtained by each module is shown in Figure 5.2. The goal of the DM is to combine both reference and target feature tensors producing a binary-like image, whose white pixels represent regions of the target frame with the most discrepancies in comparison to the target frame. A sample of an output produced by the DM can be seen in Figure 5.2 (c). Due to the different illuminations, shadows, and misalignment of both reference and target videos, the binarized image produced by the DM may contain white pixels in regions where the anomalies are not present.

During the construction of the network, it was noted large false positive regions in the image output by the DM. This problem drove the development of further modules to eliminate false positive pixels while maintaining the true positive ones. Thereby, the temporal consistency module (TCM) and the mathematical morphology module (MM) were designed to eliminate specific problems, as discussed later.

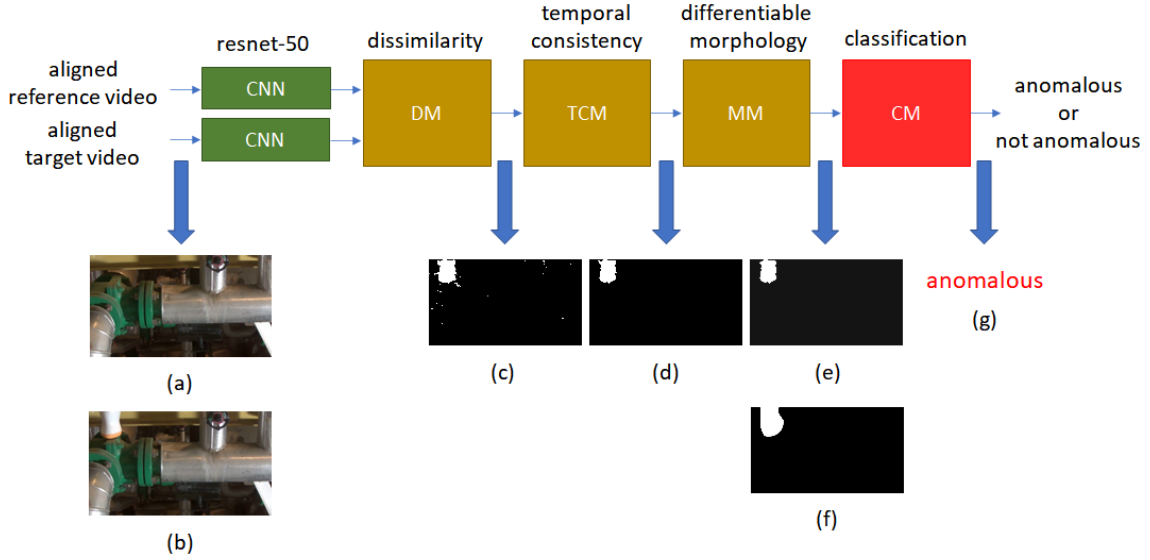


Figure 5.2: Examples of output by each module proposed in this chapter. Given a pair of aligned reference and target frames (a) and (b) respectively, the DM converts their pair of feature tensors into a single binary image (c). The TCM removes FP pixels that do not persist in consecutive frames, as seen in (d). The MM applies two morphological operations, opening, and closing, resulting in images (e) and (f), respectively. Based on the number of pixels in (f), the CM outputs the classification of the target frame, as in (g).

The TCM aims to eliminate the white pixels presented in images generated by the DM that do not persist in consecutive frames. This module removes the blinking-pixel effect of some areas of the image, which may be caused by the illumination changes and misalignment of both target and reference frames. As an anomalous object is expected to be noticeable in successive frames, they should not be eliminated by the TCM. Figure 5.2 (d) shows an output image produced by the TCM. Note that some FP pixels are eliminated by the TCM.

The MM is designed to apply differentiable morphological operations to remove the pixels that could not be removed by the TCM while trying to preserve the shape of anomalous objects. In the MM module, the opening and closing operations are performed in sequence with a circular structuring element, whose radius is optimized during the learning process. Figures 5.2 (e) and (f) show the output images produced by the opening and closing operations, respectively. Note that FP pixels are eliminated by the opening operation, and the area of the object is partially expanded by the closing operation.

Based on the number of white pixels left by the MM, the CM learns a threshold value to classify the image as anomalous or not anomalous. The classification given by the CM refers to the presence or absence of anomalies in the target, which can be seen in Figure 5.2 (g).

In Section 5.1, the proposed geometrical alignment of the reference and target frames is covered. During the training phase, images output by the DM, TCM, and MM are compared to binary ground-truth images with the Matthews correlation coefficient (MCC) metric, as described in Section 5.2. The dissimilarity module (DM), temporal consistency module (TCM), differentiable morphology module (MM), and classification module (CM) are presented in Sections 5.3, 5.4, 5.5 and 5.6, respectively. In section 5.7, the training, validation and testing processes are reviewed. Results are presented in Section 5.8 and conclusions in Section 5.9.

5.1 Geometric Alignment

The temporal alignment described in Section 3.1 aims to find the best reference frame given a target one. Even though such approach provides a real-time alignment, bringing advantage in comparison to other works using the VDAO database [31, 33, 80, 103], it solves the alignment problem only to a certain extent. During the first experiments with the pipeline proposed in the current chapter, we noticed that the classification results drop significantly in parts of the videos where the alignment is visually deficient. The poor results occur due to the incapacity of the temporal alignment to correct different camera movements, which occur in about 1/3 of the robot path in all videos.

As the results of previous works [31, 80, 103] on the VDAO database were obtained using geometrical alignment techniques, in order to better evaluate our proposed pipeline against those from [31, 80, 103], we opted to develop an alternative geometrical alignment on top of the temporal alignment. To assess our temporal alignment, the root mean squared difference (RMSD), also referred to as root mean squared error (RMSE), is used. RMSE is a conventional intensity-based quantitative metric used to compare the similarity of two images and has been employed by different works to evaluate the registration accuracy of video frames [9, 136]. It can be computed by measuring the differences of the pixels intensities of the target image T_i and reference image R_j as

$$\text{RMSE}(T_i, R_j) = \sqrt{\frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \left(T_i(x, y) - R_j(x, y) \right)^2}. \quad (5.1)$$

As shown in Figure 5.3, the absolute difference of misaligned regions of a reference

and a target frame is visually distinguishable from the one of well-aligned regions. A badly aligned pair of frames is highly correlated to a high RMSE value. Therefore, by analyzing the RMSE values along temporally aligned frames of a pair of reference and target videos, it is possible to notice in which parts of the video the camera movement impairs the quality of the overall alignment and, consequently, impacts the quality of the outcomes of the dissimilarity module.

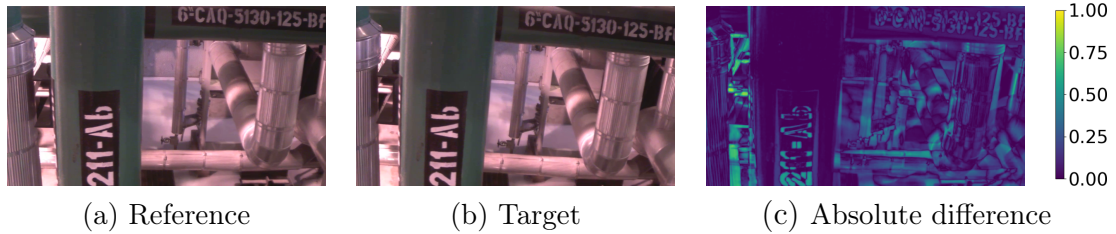


Figure 5.3: Example of reference (a) and target (b) frames temporally aligned and their absolute difference (c). This pair represents frame 180 from Figure 5.4 with $RMSE=0.1685$.

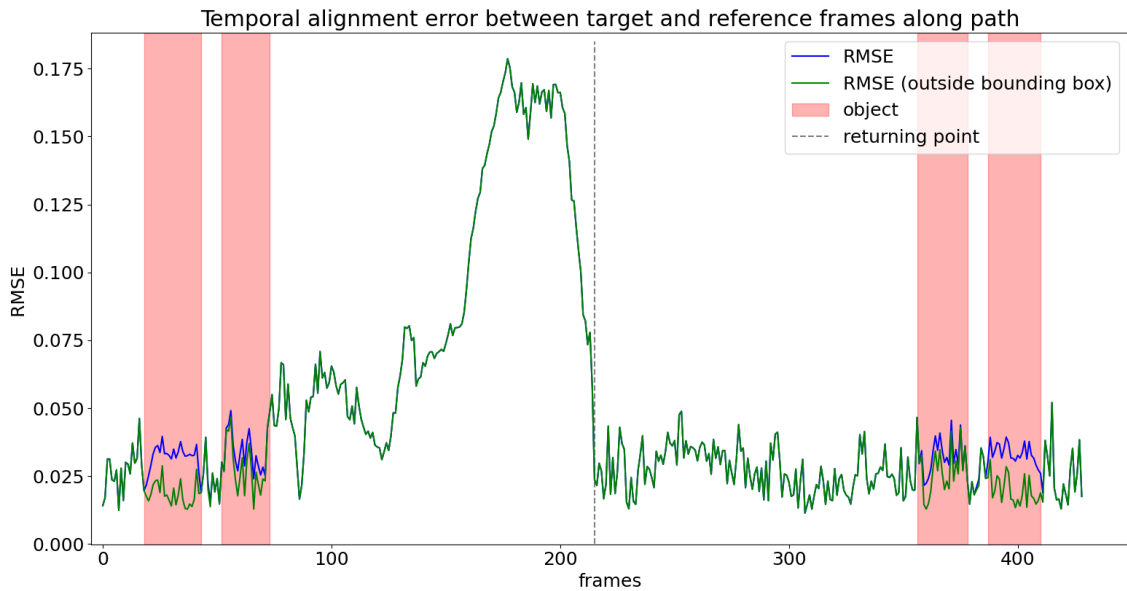


Figure 5.4: RMSE between corresponding frames of the reference and target videos aligned with the temporal alignment approach.

The plot in Figure 5.4 shows RMSE values of all temporally aligned frames of a target and reference video pair. The blue line represents the RMSE values considering the pixels of the whole frame, while the green line represents the RMSE excluding the pixels of the bounding boxes encompassing the anomalous object. The region between frames 120 and 220 represents a segment of the robot’s track with vibrations that tend to increase the geometric misalignment between reference and target frames, and thus lead to high RMSE values.

As pixels of anomalous objects in the target frame differ considerably from pixels in the same region in the reference frame, the RMSE is expected to be higher

only in parts of the videos where the target frames contain the anomalous object (represented by the pink regions in Figure 5.4). Nevertheless, this is not always true, since misaligned regions can increase the computed RMSE considerably. This is illustrated in Figures 5.3, 5.5 and 5.6. Figure 5.5 shows a pair of well-aligned frames from a relatively low RMSE region where the object is not present, and Figure 5.6 shows a pair of well-aligned frames in which the object is present in the bottom-left part of the target frame. Figure 5.3 shows a pair of frames with a high RMSE value in which the object is not present. Comparing these three examples, one notes that the presence of the object in the bottom-left corner of the target frame of Figure 5.6(b) corresponds to a region with relatively high values in the absolute difference image, highlighting the presence of the anomalous object. Similarly, regions containing high RMSE values are also present in the image with absolute difference of Figure 5.3(c). Such regions do not correspond to the presence of any anomalous object, but they do represent misaligned regions.

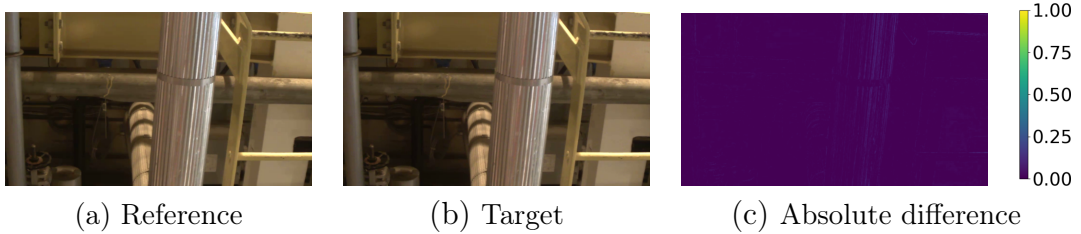


Figure 5.5: Example of reference (a) and target (b) frames temporally aligned, and their absolute difference (c). This pair represents frame 0 from Figure 5.4 with RMSE=0.0141.

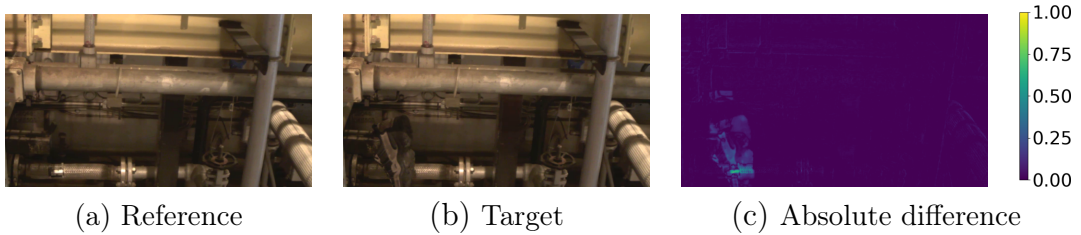


Figure 5.6: Example of reference (a) and target (b) frames temporally aligned, and their absolute difference (c). This pair represents frame 28 from Figure 5.4. Excluding the bounding box region containing the object, the computed RMSE is 0.0126.

To overcome the misalignment caused by the camera movement, which makes the DM to produce false positive pixels, a pre-processing operation that performs geometrical alignment is proposed. The quantitative criterion used to measure the efficiency of the proposed method is the RMSE, and visual results showing the improvement in the quality of the absolute difference will be presented in the end of this section.

To perform the geometrical alignment between the temporally aligned target and reference frames, an homography represented by a transformation matrix \mathbf{H}

is used to transform geometrically the reference image by means of a rotation and translation. By that, we expect to improve the overall alignment of given target and reference frames. Let us consider the following isometric transformation \mathbf{H} applied at the pixel coordinates (x, y) of the reference frame R_j to rotate and translate the position of its pixels. The transformation matrix \mathbf{H} is represented as

$$\begin{aligned} \mathbf{x}' &= \mathbf{H}\mathbf{x} \\ \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} &= \begin{bmatrix} \cos(\theta) & -\sin(\theta) & t_x \\ \sin(\theta) & \cos(\theta) & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}. \end{aligned} \quad (5.2)$$

By applying transformation \mathbf{H} , a reference frame can be rotated by θ , translated horizontally by t_x and vertically by t_y . As a complete mapping $(x, y) \rightarrow (x', y')$ is not guaranteed, the nearest neighbor interpolation method was performed to fill missing values at (x', y') coordinates.

The matrix \mathbf{H} is estimated based on the motion vectors between a pair of reference and target frames. The motion vectors are obtained by computing the optical flow [137], a technique used to estimate the motion of every pixel between two images.

Knowing that the scene of the VDAO database contains pipes and objects in different depths and sizes, as the camera moves, pixels appear to be moving with different velocities, which results in estimated motion vectors with different lengths. Even so, the camera movements tend to produce a smooth optical flow. However, in some cases, due to the presence of anomalous objects and misaligned regions, the motion vectors of the same pair of frames can become quite erratic at the region of the anomaly, as illustrated in Figure 5.7.

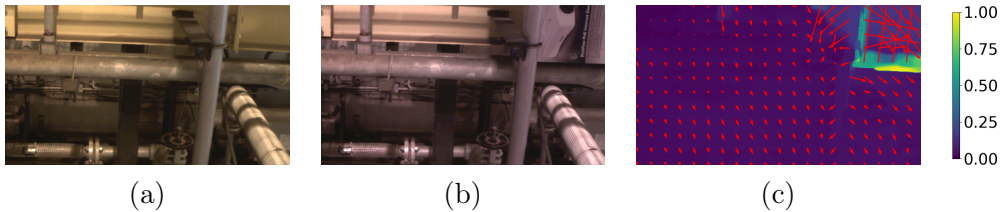


Figure 5.7: Motion vectors (c) obtained with optical flow between reference (a) and target (b) frames, which contains an anomalous object in the upper-right corner. The arrows in (c) represent the motion vectors (for better visualization, only the vectors corresponding to some of the pixels are shown). The color indicates the normalized norm of the vectors of each pixel.

The matrix \mathbf{H} is then computed based on the motion vectors obtained with optical flow. Vectors located in regions where an object might be positioned, where the vector field is erratic, as well as in locations representing bad alignments, should be discarded. Such vectors are considered outliers and may result in a bad estimation

of \mathbf{H} . The random sample consensus (RANSAC) [138] algorithm is used to select the motion vectors which best fit the model. RANSAC is an iterative robust method used to estimate parameters of a linear model by sampling points and finding the optimal fitting result.

To find the reference frame that best aligns with a given target frame T_i , the transformed reference frame R' that produces the best geometrical alignment with T_i may not be the one computed with reference frame R_j that resulted from the temporally aligned pair R_j and T_i . There are cases where the previous reference frame R_{j-1} or the next reference frame R_{j+1} can produce a better geometrical alignment with the target frame T_i . For that, a search method is applied in neighboring frames of the reference frame R_j . We consider a window containing nine neighboring frames of R_j (from R_{j-4} to R_{j+4}), and applied RANSAC to compute nine transformations between T_i and each reference frame within the window. Each reference frame is then transformed with its respective H_j . The scheme in Figure 5.8 illustrates this process using 7 neighboring reference frames.

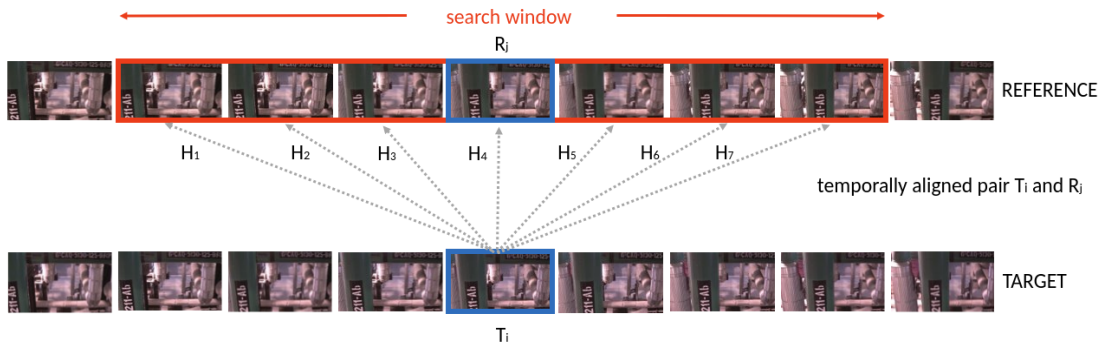


Figure 5.8: Process to compute the transformation matrices H_1, H_2, \dots, H_7 for a given pair of temporally aligned frames R_j and T_i considering 7 neighboring frames of R_j .

Transforming the coordinates of a reference frame R_j with matrix H_j may map pixels outside the frame to the transformed one, thus producing black borders on the resulting image as shown in Figure 5.9. To eliminate the black borders, 5% of each side of both target and transformed reference frames are cropped. So, the original resolution 360×640 is reduced to 324×576 .



Figure 5.9: Observable black borders in a reference frame after rotation of 10 degrees.

After eliminating the border of each transformed reference frame $R'_{j+\delta}$ and its associated target frame T_i , the RMSE between $R'_{j+\delta}$ and T_i is computed. Among the 10 geometric transformed reference frames, the one which obtains the lowest RMSE is chosen from the geometrically aligned frame with the target frame T_i . The reference frames were transformed instead of the target ones, due to the presence of the annotated bounding boxes in the target frames. If the target frames were transformed with the rotation and translating matrix \mathbf{H} , the annotated bounding boxes coordinates would also have to be transformed, which would demand an extra work.

Implementation details: Although the search for the best-transformed reference frame can produce better-aligned pairs, the process is slow. To reduce the processing time, frames are resized to $\frac{1}{4}$ of their original resolution before being processed with the optical flow and RANSAC. Once the best reference frame is chosen, a new transformation \mathbf{H} is found with the frames in their original resolution. This way, the processing time of an eight-minute video drops from 10 to 6 hours using a computer with a processor i7 CPU@4 GHz.

To eliminate illumination differences between the target and reference frames, an extra experiment was performed by pre-processing the reference and target frames with contrast limited adaptive histogram equalization (CLAHE) [139]. Nevertheless, the attempt to produce better results did not work as expected. Figure 5.10 compares the absolute difference of a pair of temporally aligned frames with and without CLAHE. When both frames are pre-processed with CLAHE, the region containing the anomalous object in the absolute difference, Figure 5.10(g), presents higher values when compared with the same region in Figure 5.10(c) - when CLAHE is not used. On the other hand, when CLAHE is applied, the absolute difference of regions where no object is present also holds higher values, similarly to badly aligned regions.

The plot in Figure 5.11 shows the RMSE values of frames obtained with the geometrical alignments with CLAHE and without CLAHE. Note that pre-processing the frames with CLAHE not only produced visually bad results (see Figure 5.10) but also led to geometrically aligned frames with higher RMSE values. So, processing the frames with CLAHE was found out not to be a useful procedure. The plot of Figure 5.12 shows that most of the frames aligned with the proposed geometrical alignment reduce the RMSE between the target and reference frames.

Due to the many computations of the transformation matrices \mathbf{H} , the geometrical alignment is very time-consuming and cannot be applied in real time. Therefore, the geometrical alignment was used to produce a version of the VDAO database containing aligned videos, whose pair of frames were used to train, validate and test the proposed model. Note that in practical applications inertial and magnetic

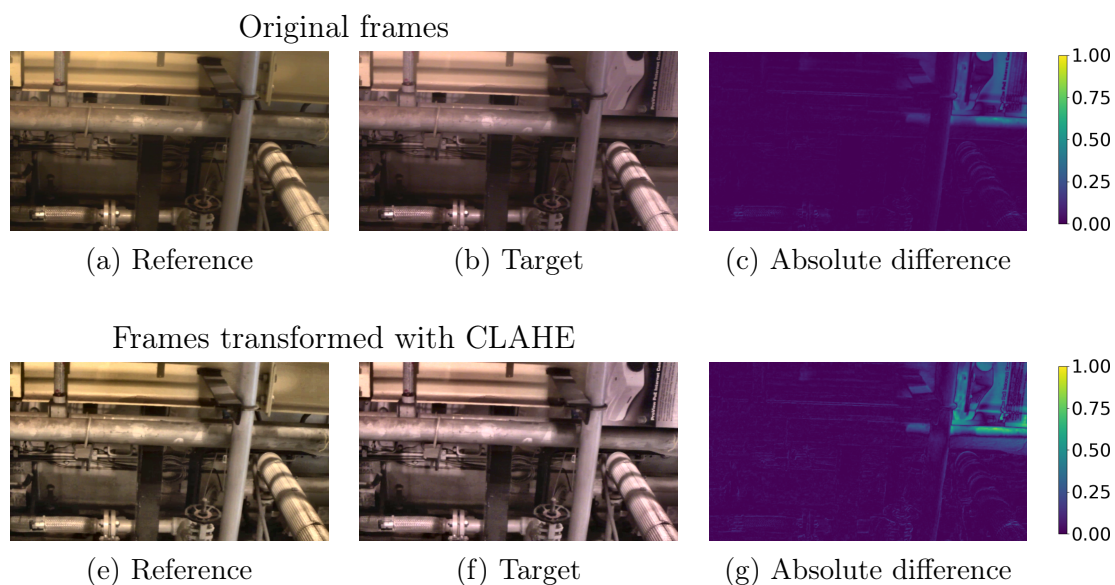


Figure 5.10: Absolute differences of a pair of temporally aligned reference and target frames. The absolute difference between frames without being processed with CLAHE ((a) and (b)) is represented in (c) with $\text{RMSE}=0.0603$. The absolute difference of frames processed with CLAHE ((e) and (f)) is represented in (g) with $\text{RMSE}=0.0984$.

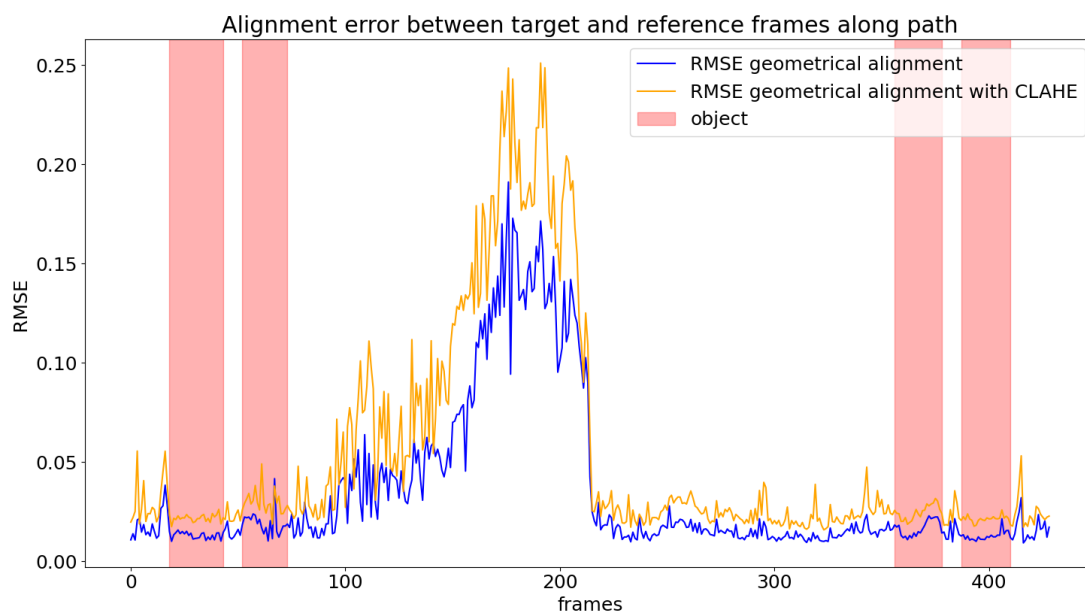


Figure 5.11: Quantitative results obtained by geometrical alignment with and without CLAHE. The pink regions indicate anomalies in the target frames.

sensors can be used to speed up the alignment operation. The work in [140] uses auxiliary sensors such as accelerometer and gyroscope to improve the quality of the alignment of the VDAO videos. Light detection and ranging (Lidar) sensors are also applied to align videos, as shown in [141, 142].

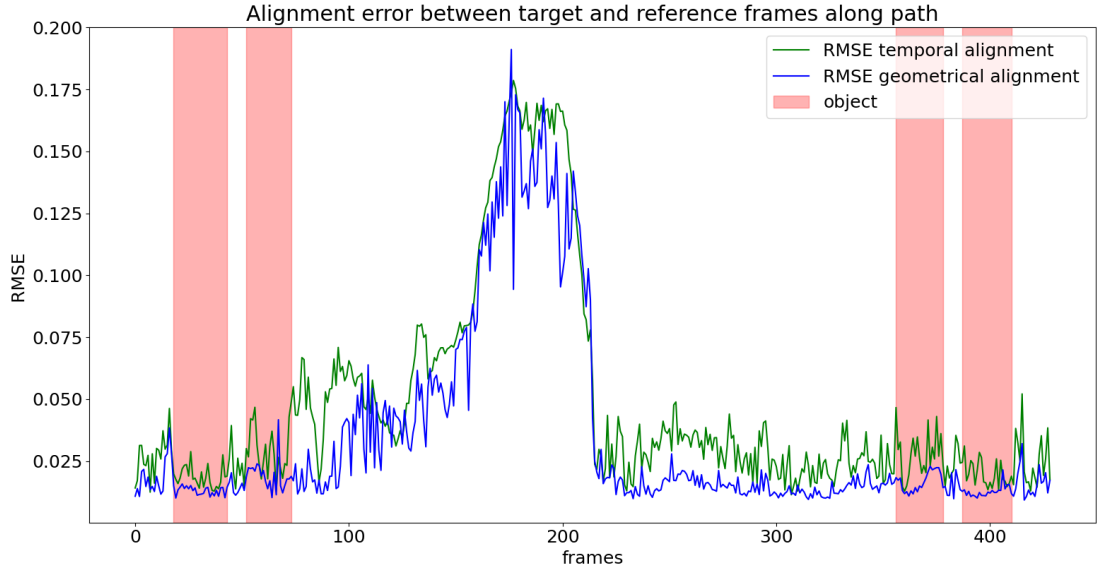


Figure 5.12: Comparative quantitative results obtained by geometrical and temporal alignments. The pink regions indicate anomalies in the target frames.

5.2 Matthews Correlation Coefficient (MCC)

The products of the dissimilarity (DM), temporal consistency (TCM) and differential morphology (MM) modules are binary-like images (grayscale images containing mostly pixels near the two extremes of the dynamic range), whose pixels with large values (white) represent regions of the target frame that are substantially different from the reference frame. The connected white pixels of these output images are expected to represent the silhouettes of anomalous objects. As the current version of the VDAO database contains only rectangular ground-truth annotations to represent the position of the anomalous objects, to evaluate the images output by the aforementioned modules, we use these available rectangular annotations to create ground-truth binary images to be compared to the output images as shown in Figure 5.13. Due to the irregular shapes of the anomalous objects, the output images (as the one in Figure 5.13(c)) are commonly formed by sinuous shapes representing the silhouettes of the objects. Even though the ground-truth binary images formed by the bounding boxes (Figure 5.13(b)) are not perfect representations of the expected output images, in order to avoid a manual re-labeling of the VDAO dataset to obtain ground-truth silhouettes of the objects, we decided to use the rectangular ground-truth annotations, which produced satisfactory results. In databases where anomalous objects are provided at the pixel level (silhouette annotations), the techniques presented in this work can also be used and, as the annotations are more trustworthy, the results might be improved relative to the ones obtained with the rectangular bounding boxes annotations.

In a pixel-level comparison of two binary images, the first and straightforward

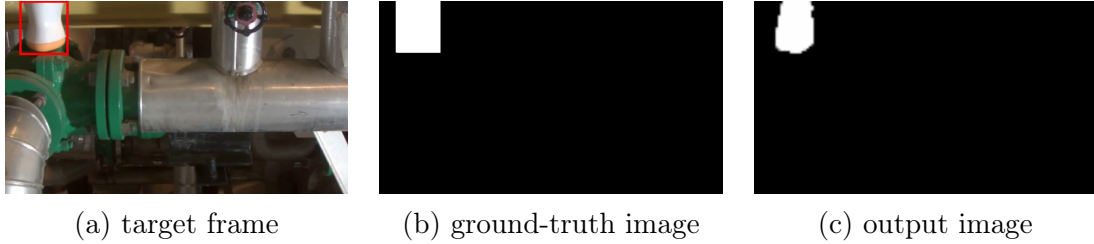


Figure 5.13: Examples of a ground-truth image and an output image used to compute the MCC. Given a target frame (a), whose ground-truth bounding-boxes are provided (in red), a ground-truth binary image (b) is created to be compared to an image output by the DM, TCM, and MM, as in (c).

metric used to measure the difference between the two images is the mean-squared error (MSE). However, as in this work the number of white pixels (anomalous) and black pixels (not anomalous) in the ground-truth images are unbalanced, the MSE when used as a loss function leads to unsatisfactory results, impairing the learning process of our model. More specifically, if our model produces a binary image by randomly guessing the pixel values (black or white) following the same probability distribution of the ground-truth images, MSE would provide relatively low loss values [143–146], as shown in Figure 5.14. Thus, MSE is not able to distinguish such a random model from a reasonable performing one. Confirming what has been demonstrated in [146], in our experiments, we observed that using the Matthews correlation coefficient (MCC) as a loss function to express the differences between the ground-truth and the output images, better results than the ones obtained with other metrics such as the MSE have been achieved. Therefore, in order to train and optimize our pipeline we adopted the MCC.

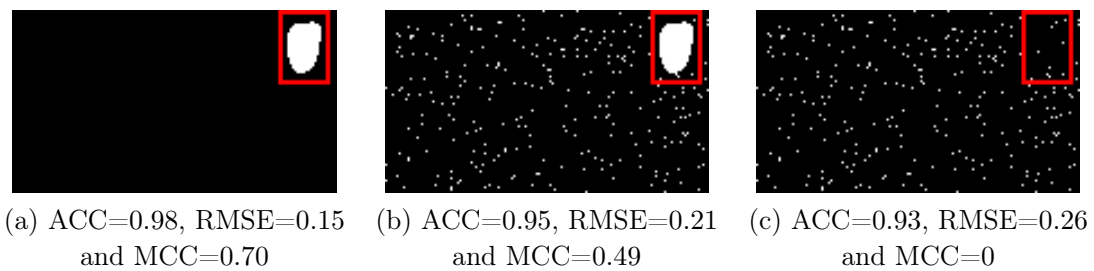


Figure 5.14: Comparative metrics: accuracy (ACC), RMSE and MCC. All three cases (a), (b) and (c) show different detections considering the same ground truth, represented by the red rectangle. In (b), random false positives were added, and (c) only random pixels are considered. Note that the accuracy does not vary much in all 3 cases. The RMSE cannot represent precisely the quality of the detection. The MCC is reduced when false detections increase, as noted in (b). And $MCC=0$, as in (c), when the positives are randomly chosen following the proportion of the positives in the ground-truth.

MCC is an informative metric to deal with skewed distributions and is calculated

with the following equation

$$\text{MCC} = \frac{(\text{TP} \times \text{TN}) - (\text{FP} \times \text{FN})}{\sqrt{(\text{TP} + \text{FP})(\text{TP} + \text{FN})(\text{TN} + \text{FP})(\text{TN} + \text{FN})}}, \quad (5.3)$$

where TP, FP, TN, and FN are respectively the numbers of true positive, false positive, true negative, and false negative pixels of the output image in comparison with the ground-truth image. In the best-expected case, where FN=0 and FP=0, MCC is 1, and in the worst case, where TP=0 and TN=0, MCC is -1 . When the probability of the positive detections is equal to the proportion of the positive samples of the ground-truth, the MCC is zero [146], as shown in Figure 5.14(c).

A more intuitive way of defining the MCC is as follows. If the ground-truth image is represented as a vector $\mathbf{x} \in \mathbb{R}^{d \times 1}$, where $d = m \times n$, m and n being its height and width, respectively, and the output image represented as a vector $\mathbf{y} \in \mathbb{R}^{d \times 1}$, the MCC value can be computed using

$$\text{MCC} = \frac{\sum_{i=1}^d (\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{y}_i - \bar{\mathbf{y}})}{\sqrt{\sum_{i=1}^d (\mathbf{x}_i - \bar{\mathbf{x}})^2} \sqrt{\sum_{i=1}^d (\mathbf{y}_i - \bar{\mathbf{y}})^2}}, \quad (5.4)$$

where $\bar{\mathbf{x}} = \frac{1}{d} \sum_{i=1}^d \mathbf{x}_i$ and $\bar{\mathbf{y}} = \frac{1}{d} \sum_{i=1}^d \mathbf{y}_i$ are the mean values of the ground-truth and output images, respectively.

Works [143–145] have indicated that MCC results are more truthful if positive and negative classes are imbalanced and have the same importance in the expected classification results. Also, it is important to note that the computation of the MCC may be undetermined in certain cases, which would prevent its application as a loss function. Nevertheless, small modifications in the images generating vectors \mathbf{x} and \mathbf{y} can be made, making the use of MCC as a loss function feasible, as shown next.

5.2.1 Modifications Necessary to Use MCC as a Loss Function

The MCC value is undefined if the denominator in Equation (5.4) is 0, which occurs when $\mathbf{x}_i = \bar{\mathbf{x}}$ or $\mathbf{y}_i = \bar{\mathbf{y}}$, $\forall i$. Such cases can happen when the binary ground-truth image \mathbf{x} or the output image \mathbf{y} is constant, being either an all-white or all-black image, which is reasonably likely to happen in practical cases. For instance, when the output of a module is an image with no positives, $\mathbf{y}_i = \bar{\mathbf{y}} = 0$, $\forall i$, as in Figure 5.15 (c), the MCC is undefined. Another case where MCC is undefined can be seen in Figure 5.15 (d), where the ground-truth target frame does not contain an anomalous

object, and so $\mathbf{x}_i = \bar{\mathbf{x}} = 0, \forall i$. Therefore, we had to make small modifications on the ground-truth and target frames, to make the use of the MCC as a loss function possible.

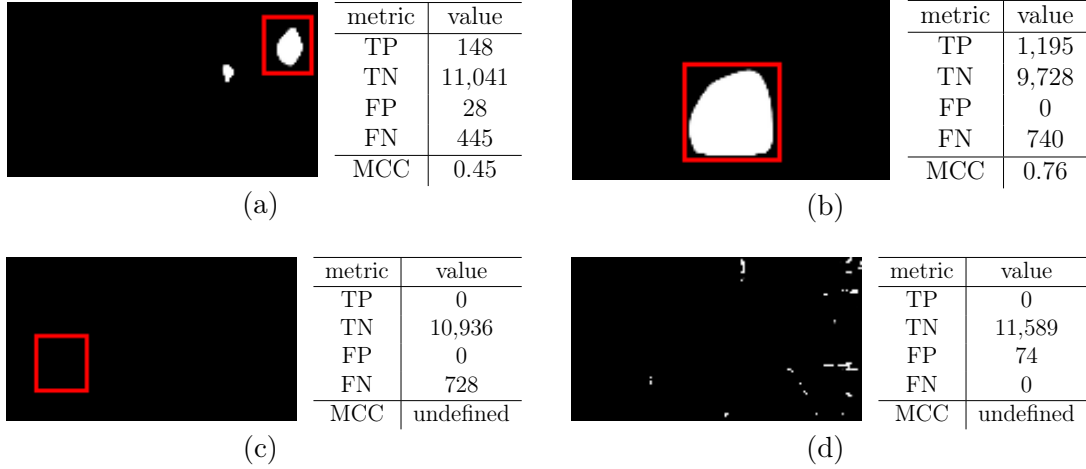


Figure 5.15: Examples of images output by the modules DM, TCM, and MM and their respective metrics. The MCC values are calculated with Equation (5.3). The red boxes in (a), (b), and (c) represent the areas of the ground-truth annotations where the anomalous object is located. In (d), there is an example of an output image whose ground-truth target frame is not anomalous.

The strategy we propose to avoid undefined MCC values is to change the value of corresponding pixels of both ground-truth image \mathbf{x} and output image \mathbf{y} in cases where the conditions $\mathbf{x}_i = \bar{\mathbf{x}}$ and $\mathbf{y}_i = \bar{\mathbf{y}}$ occur. To illustrate the proposed strategy, let us consider the examples shown in Figure 5.16. The original images \mathbf{x} and their respective output images \mathbf{y} result in undefined MCC values according to Equation (5.4). Without loss of generality, let us consider all images with resolution width= W and height= H , and the total number of pixels $Z = W \times H$. Our strategy can be summarized in the following cases:

- Case 1: In situations as seen in Example 1 and Example 2, where both ground-truth \mathbf{x} and output images \mathbf{y} present all pixels with constant values, a random pixel in the ground-truth image and its corresponding pixel in the output image will have their values inverted, resulting in images as shown in (b) and (d). Therefore, samples in Example 1, which previously led to TP=0, FP=0, TN= Z , FN=0 and MCC=*undefined*, are now modified to (b), resulting TP=1, FP=0, TN= $Z - 1$, FN=0 and MCC=1. Example 2, which previously had TP=0, FP= Z , TN=0, FN=0 and MCC=*undefined*, is now modified to (d), having TP=0, FP= $Z - 1$, TN=0, FN=1 and MCC=-1.
- Case 2: In situations as seen in Example 3, where only the ground-truth image \mathbf{x} is formed by pixels with constant values and the output image \mathbf{y} contains

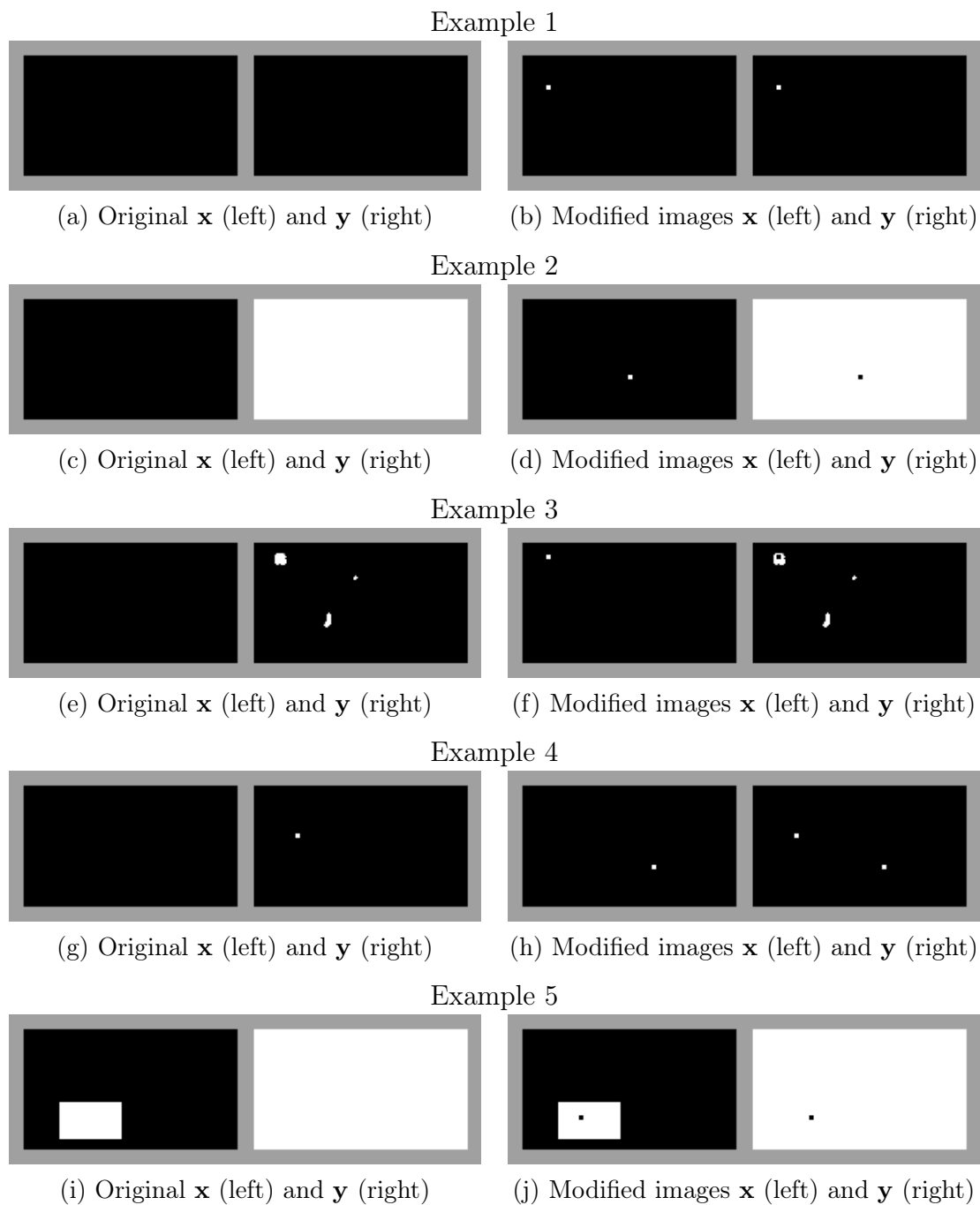


Figure 5.16: Examples of ground-truth \mathbf{x} and output images \mathbf{y} resulting in undefined MCC values. Five examples containing pairs of ground-truth \mathbf{x} and output \mathbf{y} images are shown in (a), (c), (e), (g) and (i), and their respective modified versions to allow the computation of MCC are shown in (b), (d), (f), (h) and (j). The outer gray frame was added around the images just for visualization purposes, since it makes the image bounds perceptible. The size of the modified pixels was exaggerated for better visualization.

the total number of anomalous pixels p , being $1 < p < Z$, a pixel is selected in the output image \mathbf{y} whose value is different than its corresponding pixel in the ground-truth image. Then, the selected pixel in the output image and

its corresponding pixel in the ground-truth image are switched, as seen in (f). Thus, Example 3 which previously had $TP=0$, $FP=1 < p$, $TN=Z - p$, $FN=0$ and $MCC=undefined$, now becomes $TP=0$, $FP=p - 1$, $TN=Z - p$, $FN=1$ and $MCC=\frac{-(p-1)}{\sqrt{(p-1)(Z-1)(Z-p+1)}}$.

- Case 3: Slightly different from case 2, Example 4 illustrates a situation where only the ground-truth image is formed by pixels with constant values and the output image contains a single anomalous (white) pixel $p = 1$. In this case, a pixel is selected in the ground-truth image whose value is equal to its corresponding pixel in the output image. The selected pixel in the output image and its corresponding pixel in the ground-truth image have their values inverted (a black pixel becomes white and a white pixel becomes black), as seen in (h). Thus, Example 4, which previously had $TP=0$, $FP=p = 1$, $TN=Z - 1$, $FN=0$ and $MCC=undefined$, now becomes $TP=1$, $FP=1$, $TN=Z - 2$, $FN=0$ and $MCC=\frac{(Z-2)}{\sqrt{2(Z-1)(Z-2)}}$.
- Case 4: In situations where the output image is formed by pixels with constant values, and the ground-truth image contains a ground-truth object in a rectangular area formed by a pixels, as in Example 5(i), a pixel in the ground-truth image inside area a and its corresponding pixel in the output image are inverted (a black pixel becomes white and a white pixel becomes black), as seen in (j). Therefore, Example 5(g), which previously had $TP=a$, $FP=Z - a$, $TN=0$, $FN=0$ and $MCC=undefined$, now becomes $TP=a - 1$, $FP=Z - a - 1$, $TN=1$, $FN=0$ and $MCC=\frac{a-1}{\sqrt{(Z-2)(a-1)(Z-a)}}$. In the training samples used in this work, there was no case where $a = 1$.

With this strategy, the MCC can be computed and used as a metric to optimize the learning parameters of the DM, TCM, and MM, as presented in the following sections.

5.3 Dissimilarity Module (DM)

Given the input tensors representing features of an aligned pair of reference and target frames, the goal of the DM is to produce a binary image whose white pixels represent the most contrasting regions of both frames. The operations performed in the feature tensors by DM are represented in Figure 5.17.

As explained in the previous section, the frames output by the geometric alignment are 324×576 pixels. When processed by the Resnet-50, each frame results in 256 81×144 feature maps. In this new approach, instead of subtracting the feature tensors (as done in Chapters 3 and 4), the idea is to weight each feature map,

so those which perceive anomalous regions are accentuated, and the contribution of those which cannot distinguish the anomalous regions is lessened. Then, more operations are included in order to produce a binary image, and the richness of the features tensors can be exploited to a fuller extent than it is by a simple difference. To that end, since there are 256 feature maps ($\mathbf{T}_{t_1}, \dots, \mathbf{T}_{t_{256}}$) for each target frame, and 256 feature maps ($\mathbf{T}_{r_1}, \dots, \mathbf{T}_{r_{256}}$) for each reference frame, the first set of operations contemplates 512 learnable weights, being $(w_{r_1}, w_{r_2}, \dots, w_{r_{256}})$ the weights of the 256 reference feature maps and $(w_{t_1}, w_{t_2}, \dots, w_{t_{256}})$ the weights of the 256 target feature maps. The weighted feature tensors are then element-wise subtracted, forming a unique feature tensor.

In addition, each channel of the resulting tensor is added to a bias $(b_1, b_2, \dots, b_{256})$ and is applied a non-linearity, producing one feature map \mathbf{T}_i for each input feature map $i = 1, 2, \dots, 256$, as shown in Equation (5.5). Two different non-linearities were tested, the tanh and sigmoid functions. As the convergence with tanh required much fewer iterations and led to lower error rates, in our experiments this function was preferred over the sigmoid.

$$\mathbf{T}_i = \tanh(w_{r_i} \mathbf{T}_{r_i} - w_{t_i} \mathbf{T}_{t_i} + b_i \mathbf{1}), \quad (5.5)$$

where $\mathbf{1}$ denotes a matrix with the same dimensions as \mathbf{T}_i containing only ones.

The output feature maps ($\mathbf{T}_1, \mathbf{T}_2, \dots, \mathbf{T}_{256}$) are then weighted with another set of weights $(w_1, w_2, \dots, w_{256})$, and the resulting channels are added along the channel axis forming a single feature map with resolution 81×144 . To generate a binary-like image, the feature map is thresholded with a sigmoid function, producing an output image with values as close as possible to 0 or 1. The set of operations are represented in the scheme of Figure 5.17.

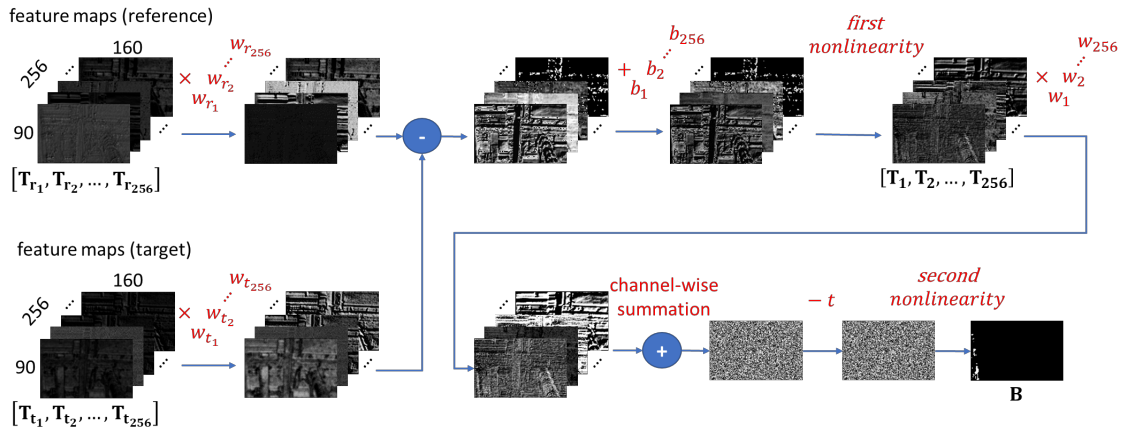


Figure 5.17: Scheme representing the operations performed by the DM to produce a 90×160 binary image. There is a total of 1,025 learnable parameters in the DM.

A hyperparameter γ and a learnable threshold t are used by the sigmoid function to produce the output binary image. The sigmoid function is defined as

$$s(x) = \frac{1}{1 + e^{-\gamma(x-t)}}, \quad (5.6)$$

where γ controls the slant of the function and t is the threshold point which moves the function along the x-axis. Figure 5.18 illustrates sigmoid functions centered in $t = 0.5$ produced with different values of γ . Therefore, the DM output image \mathbf{B} is obtained as

$$\mathbf{B} = \text{sigmoid} \left(\left(\sum_{i=1}^C w_i \mathbf{T}_i \right) - t \mathbf{1} \right), \quad (5.7)$$

with $\mathbf{1}$ as before.

The partial derivatives of the binary image \mathbf{B} with respect to each adjustable parameter can be obtained with

$$\frac{\partial \mathbf{B}}{\partial t} = -\gamma \mathbf{B} \odot (\mathbf{1} - \mathbf{B}), \quad (5.8)$$

$$\frac{\partial \mathbf{B}}{\partial w_i} = \gamma \mathbf{T}_i \odot \mathbf{B} \odot (\mathbf{1} - \mathbf{B}), \quad (5.9)$$

$$\frac{\partial \mathbf{B}}{\partial b_i} = \gamma w_i \mathbf{B} \odot (\mathbf{1} - \mathbf{B}) \odot (\mathbf{1} - \mathbf{T}_i \odot \mathbf{T}_i), \quad (5.10)$$

$$\frac{\partial \mathbf{B}}{\partial w_{r_i}} = \gamma w_i \mathbf{B} \odot (\mathbf{1} - \mathbf{B}) \odot \mathbf{T}_{r_i} \odot (\mathbf{1} - \mathbf{T}_i \odot \mathbf{T}_i), \quad (5.11)$$

$$\frac{\partial \mathbf{B}}{\partial w_{t_i}} = -\gamma w_i \mathbf{B} \odot (\mathbf{1} - \mathbf{B}) \odot \mathbf{T}_{t_i} \odot (\mathbf{1} - \mathbf{T}_i \odot \mathbf{T}_i), \quad (5.12)$$

where \odot represents the Hadamard point-wise product and $\mathbf{1}$ is defined as before.

The DM contains, together with the 256×3 weights, the 256 biases and the threshold t , only 1,025 learnable parameters, which makes the proposed structure well trainable using the VDAO database.

To produce a binary image using the sigmoid function, its steepness is controlled by a hyperparameter γ , which has to be adjusted according to the range of the values of the expected output image. The higher the γ value, the closer to a step function the sigmoid function becomes, producing images containing values closer to 0 or 1 and, therefore, being suited to be used in the same way as ones produced by a hard threshold. This can be observed in Figure 5.18, where to produce an output image with values being either 0 or 1, a higher γ value should be preferably chosen. Nevertheless, the very large values of γ that are necessary to generate outputs equal

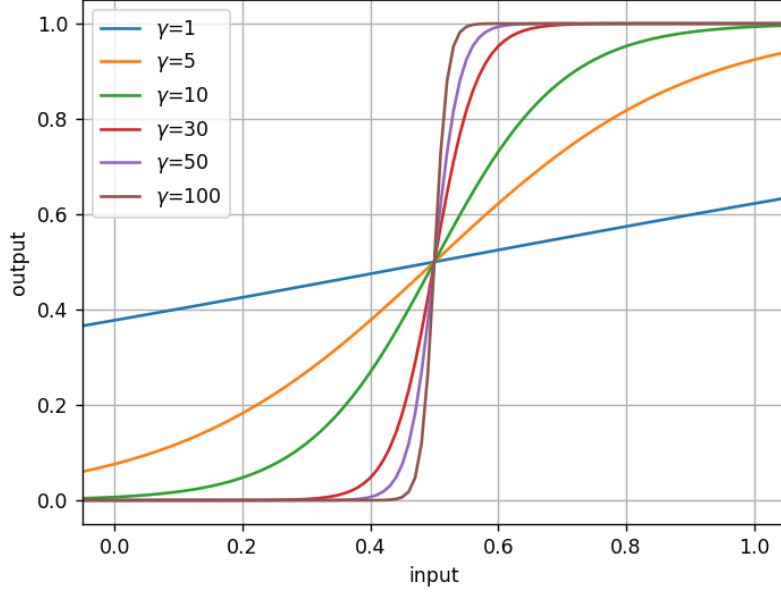


Figure 5.18: Sigmoid functions represented by Equation (5.6) with $t = 0.5$ and different values of γ .

to 0 or 1, tend to produce very large gradient values in the backpropagation process, and thus hinder the learning process of all parameters of the DM. This point will be further explored by a simple analysis as done in the sequel.

Consider the scheme exemplified in Figure 5.19. The input variable a passes by two functions $g(a)$ and $s(x)$ before being transformed into the output y . Function $s(x)$ represents the sigmoid from Equation (5.6). The output y is then compared to the expected output \hat{y} through the loss function $loss$. The p and t are learnable parameters of functions $g(a)$ and $s(x)$, respectively. According to the chain rule, the gradient of the loss with respect to the parameter p is given by

$$\frac{\partial l}{\partial p} = \frac{\partial l}{\partial y} \times \frac{\partial y}{\partial x} \times \frac{\partial x}{\partial p}. \quad (5.13)$$

In accordance with the gradient descent algorithm, the update of parameter p in the iteration i is given by

$$p_i = p_{i-1} - \frac{\partial l}{\partial p} \eta, \quad (5.14)$$

where η is a chosen learning rate.

The derivative of the sigmoid function $s(x) = y$ from Equation (5.6) with respect to the input x is given by:

$$\frac{\partial y}{\partial x} = \gamma y(1 - y). \quad (5.15)$$

Gathering all equations into Equation (5.14), we have:

$$p_i = p_{i-1} - \gamma \eta \frac{\partial l}{\partial y} \frac{\partial x}{\partial p} y(1-y). \quad (5.16)$$

If a high enough γ is chosen, so that the sigmoid function produces an output $y = 0$ or $y = 1$, one can see from Equation (5.16) that $p = p_{i-1}$ and parameter p is never updated. The same problem occurs in the updating process of threshold t . In this work the hyperparameter $\gamma = 10$ value was empirically chosen to produce an output image as close as possible to a binary image, such that the threshold t can be learned.

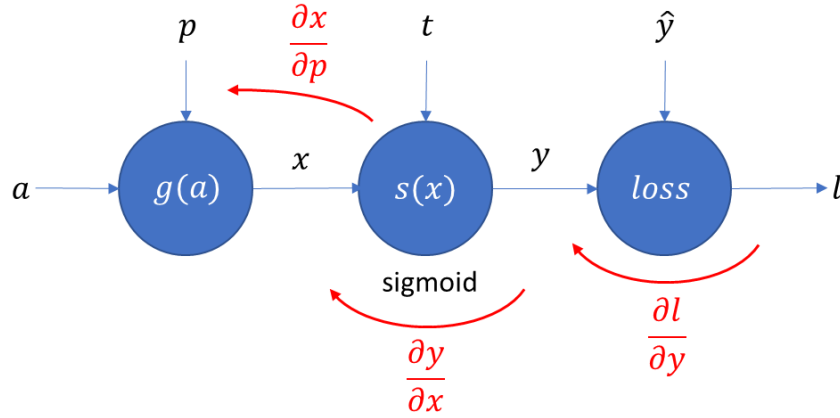


Figure 5.19: Scheme representing operations performed in sequence by a function $g(x)$ and a sigmoid function $s(x)$ to transform the input a into the output y , which is compared to an expected value \hat{y} . The update of the learnable variable p is determined by Equation (5.14) using the computed gradients represented in red.

The examples in Figures 5.21, 5.20 and 5.22 show outputs produced by a trained DM under different circumstances. In all cases, the white pixels in the output images represent dissimilar regions between the reference and target frames. Such regions represent an anomalous object, a misaligned region, or reflections on metallic surfaces.

The DM applied in the deep features proved to be very robust to illumination differences on non-reflecting surfaces, but very sensitive to shadows and misaligned regions between target and reference frames. In some frames, the anomalous objects have not entered the scene yet or have already left the scene, but their shadows are detected by the DM. As the DM does not take into account the semantic content of the scene, it is not able to distinguish if a region with strong dissimilarities refers to the presence of objects or to their shadows. Figure 5.20 shows an example where the shadow of an object that is not in the scene is noted by the dissimilarity module.

Even though the geometrical alignment of reference and target videos improves the registration of the frames, the movement of the camera hinders perfectly aligned

frames. Such a problem affects the assertiveness of the image produced by the DM. As small anomalous objects can also be noted by the DM, artifacts caused by a poor alignment and small objects cannot be distinguished by the DM, as seen in Figure 5.22. Many false positive pixels in the output of the DM are associated with spatially misaligned regions of the target and reference frames. Such sensitivity of the DM requires further post-processing modules, which are described next.

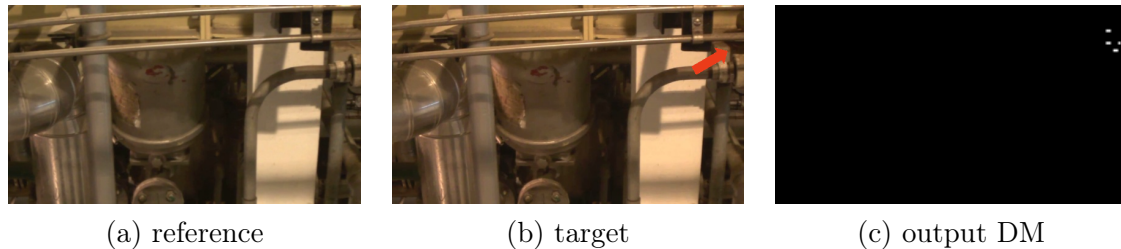


Figure 5.20: Example of a pair of aligned frames where no anomalous object is present in the target frame. The red arrow was added on the target frame pointing to a shadow area produced by an anomalous object which is not present in the scene.

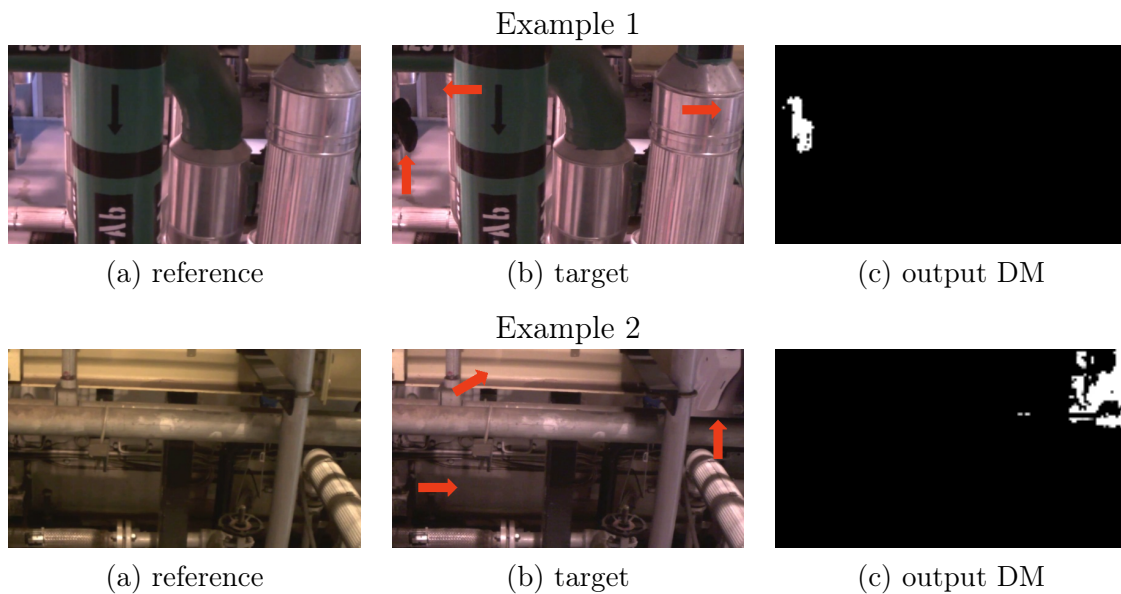


Figure 5.21: Examples of pairs of aligned reference and target frames under different illuminations. The arrows added on target frames (b) point to regions where the object is present and the differences in illumination have the most contrast. Their respective outputs produced by the DM are shown in (c).

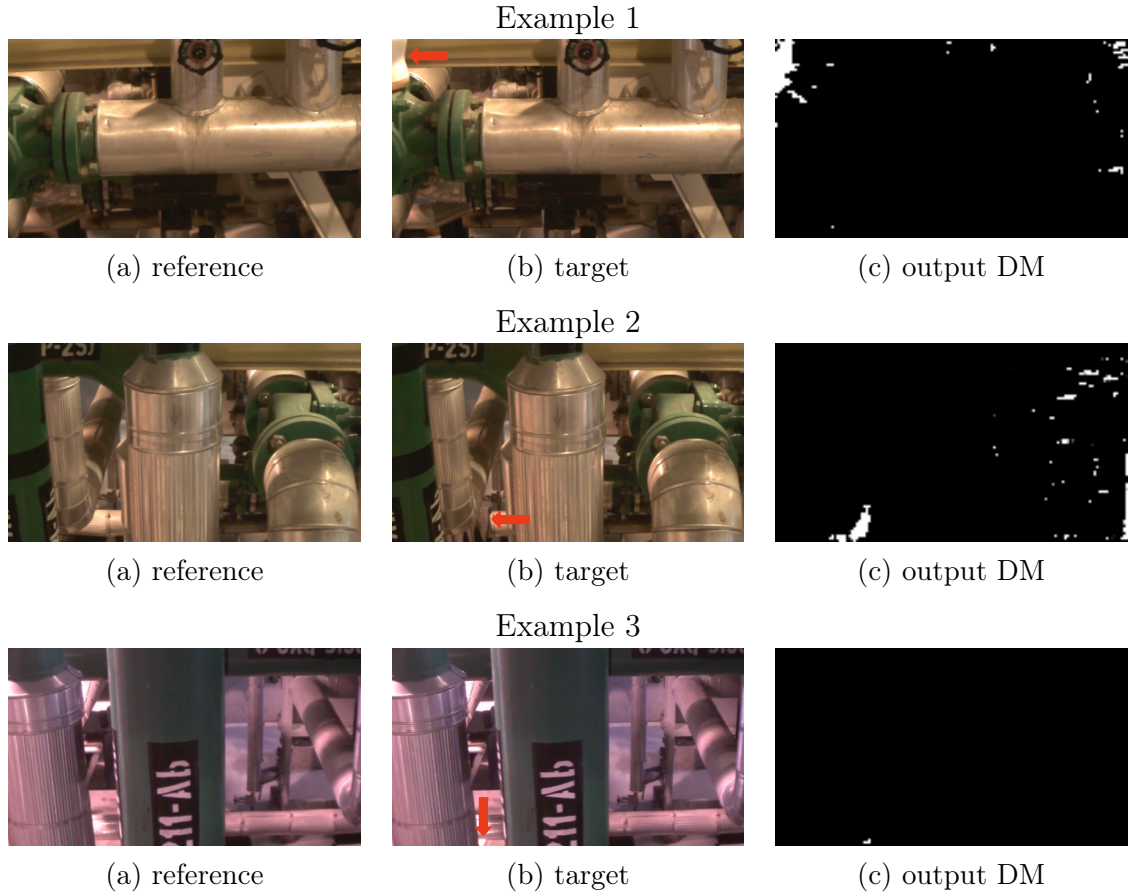


Figure 5.22: Examples of outputs produced by the DM given pairs of aligned frames under the same illumination. Red arrows are added on target frames pointing to the anomalous objects. The output produced by the DM shows white pixels representing objects and/or noises caused by a not-perfect alignment. Example 3 shows a very small object on the target frame that can be easily misidentified as noise on the output image of the DM.

5.4 Temporal Consistency Module (TCM)

Although the DM is robust to produce binary images revealing most of the structural differences between the target and reference frames, it is unable to ignore the differences caused by light reflections, misalignment, and shadows varying in consecutive frames. The TCM aims to reduce false positive pixels in consecutive frames. Visually, such false positive regions are noted as ‘blinking points’ as demonstrated in Figure 5.23.

To remove false positive pixels that are not present in consecutive output images, a temporal voting window $W_{t,s}$ with size s and centered in t is adopted. To keep the positive pixels (represented in white) of frame t , corresponding pixels in its neighboring frames are evaluated. The adopted criterion only keeps a positive

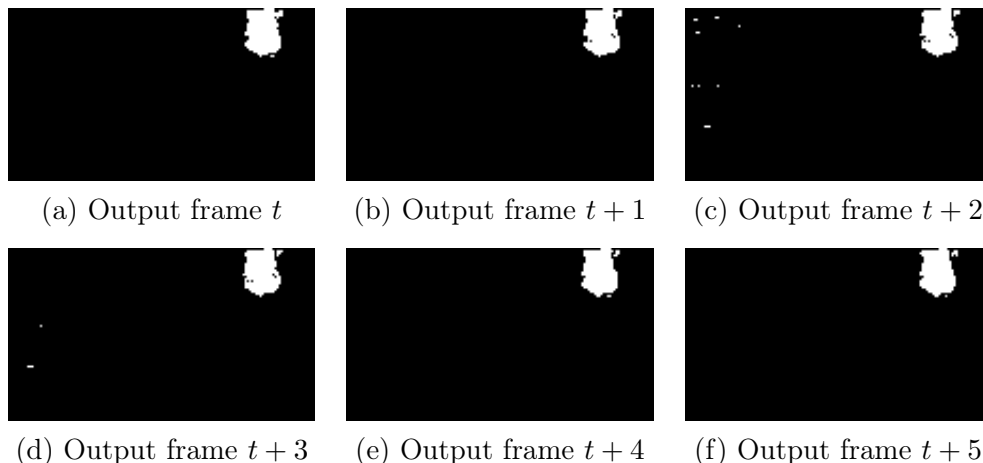


Figure 5.23: Examples of six consecutive frames produced by DM. Note the existence of false positive pixels producing the blinking effect on the left side of images (c) and (d).

(white) pixel at position (x, y) in frame t , if most pixels in the position (x, y) of frames within window s are also positive (white).

The TCM is optimized by comparing its output image with the ground truth using the MCC metric. Therefore, to choose the best window size, different window sizes are evaluated (1, 3, 5, 7, 9, 11, 13 and 15), and the one that achieves the best MCC value is picked.

5.5 Differentiable Morphology Module (MM)

Mathematical morphology operations can transform the shape of objects while preserving their structures [147]. Their applications are broad and include texture and image segmentation [148], shape identification for background removal, [147], noise suppression [149], among many others.

In morphological operations, a set represents a group of pixels within a binary or gray level image. A morphological operation consists of probing pixels of image f with the set b , called structuring element. The most basic morphological operations are erosion and dilation. Erosion transforms each pixel $f(x, y)$ by replacing it with its minimum over the structuring element; conversely, dilation transforms each pixel $f(x, y)$ by replacing it with its maximum over the structuring element. In their most general form, if f is a gray-level image and b is a non-flat structuring element, erosion and dilation are represented by Equations (5.17) and (5.18) [150], respectively,

$$[f \ominus b](x, y) = \min_{(s,t) \in b} \{f(x + s, y + t) - b(s, t)\} \quad (5.17)$$

$$[f \oplus b](x, y) = \max_{(s,t) \in b} \{f(x + s, y + t) + b(s, t)\} \quad (5.18)$$

Since the min and max are non-differentiable operations, morphological operations cannot be directly integrated in a backpropagation-based learning framework, such as a neural network. Many works have proposed alternatives to approximate such functions by differential operations, so they can be used in neural networks [151–153].

In the context of this work, as the input and output samples of the MM are expected to be binary-like images¹ it is useful to describe the erosion and dilation operations within the context of set theory. We define a binary image as a subset of \mathbb{Z}^2 containing its white pixels only. Each element i of the set is a white pixel represented by a vector with two coordinates $\mathbf{x}_i = (x, y)$. Given two sets $A, B \in \mathbb{Z}^2$, we also define the following operations [150], which are further illustrated in Figure 5.24.

- Complement: $A^C = \{\mathbf{x} \mid \mathbf{x} \notin A\}$
- Intersection $A \cap B = \{\mathbf{x} \mid \mathbf{x} \in A \text{ and } \mathbf{x} \in B\}$
- Translation: $A_z = \{\mathbf{c} \mid \mathbf{c} = \mathbf{a} + \mathbf{z}, \mathbf{a} \in A\}$
- Difference: $A - B = \{\mathbf{x} \mid \mathbf{x} \in A \text{ and } \mathbf{x} \notin B\} = A \cap B^C$
- Reflection: $\hat{A} = \{\mathbf{x} \mid \mathbf{x} = -\mathbf{b}, \mathbf{b} \in A\}$

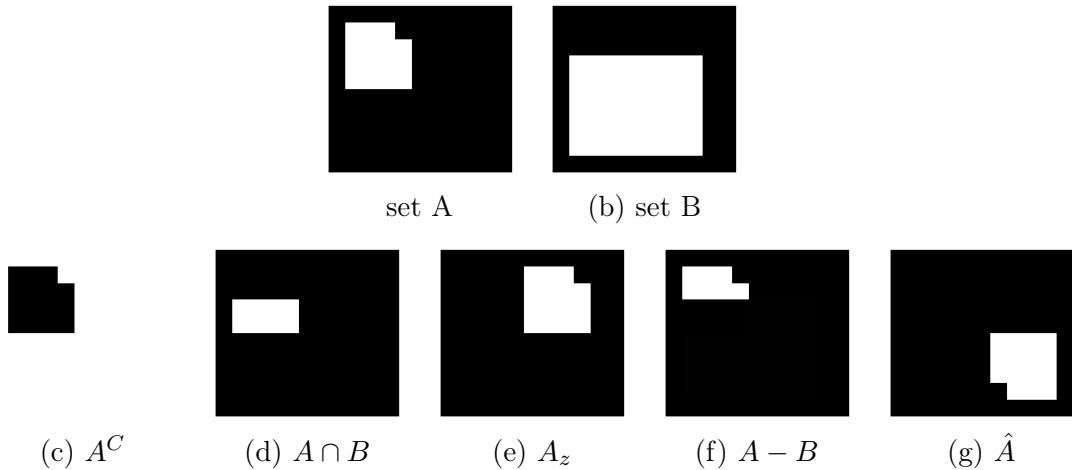


Figure 5.24: Set operations applied to binary images.

¹In this work, a binary-like image is an image whose pixels are either very similar to the maximum value or very similar to the minimum value of the image. An example of a binary-like image is one output by a steep sigmoid function given a regular image in the input.

Based on the set operations above, with set $A \in \mathbb{Z}^2$ representing the white pixels of the image, and $B \in \mathbb{Z}^2$ a structuring element, the binary morphological operation erosion is defined as

$$A \ominus B = \{\mathbf{z} \mid B_{\mathbf{z}} \subseteq A\}, \quad (5.19)$$

that is, erosion of A by B is the set of all \mathbf{z} such that B translated by \mathbf{z} is contained in A .

The binary morphological operation dilation is defined as

$$A \oplus B = \{\mathbf{z} \mid \hat{B}_{\mathbf{z}} \cap A \neq \emptyset\}, \quad (5.20)$$

that is, dilation of A by B is the set of all structuring element origin locations where the reflected and translated B overlaps A by at least one element.

It is important to mention that erosion and dilation are dual operations, which means one can be defined in terms of the other. The dilation operation $A \oplus B$, for instance, can be also expressed in terms of erosion. By eroding the complement of image A with \hat{B} , and complementing the result, one can obtain the dilation $A \oplus B$. Therefore,

$$A \oplus B = (A^C \ominus \hat{B})^C. \quad (5.21)$$

Erosion and dilation can be combined resulting in other operations. Opening, for instance, is the dilation of the erosion of a set using the same structuring element. Closing, on the other hand, is the erosion of the dilation of a set with the same structuring element. Thus, opening is defined as

$$A \circ B = (A \ominus B) \oplus B, \quad (5.22)$$

and closing as

$$A \bullet B = (A \oplus B) \ominus B. \quad (5.23)$$

One can combine Equation (5.21) with Equation (5.22) and Equation (5.21) with Equation (5.23) to represent opening and closing operations by means of erosion only. Thus, opening and closing operations become, respectively

$$A \circ B = ((A \ominus B)^C \ominus \hat{B})^C \quad (5.24)$$

and

$$A \bullet B = (A^C \ominus \hat{B})^C \ominus B. \quad (5.25)$$

Considering the most basic morphological operation, the binary erosion of image A by the structuring element B is interpreted as the position of all points reached by the center of B when B is completely inserted into A , as illustrated in Figure 5.25.

Figure 5.26 shows the results of four morphological operations using a circular structuring element. Note that the shape of the object in the eroded image shown in

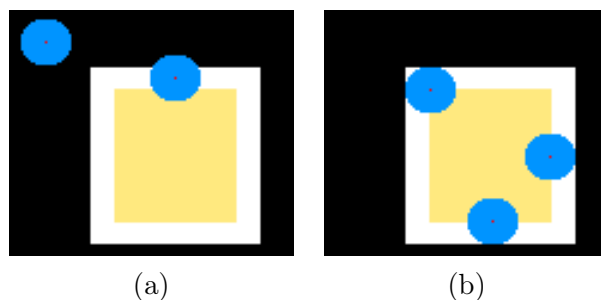


Figure 5.25: Erosion of the foreground (white square) with a structuring element (blue circle) producing the eroded image represented by the yellow shape. (a) situations where the structuring element is not completely inserted into the foreground, so its center will not produce the eroded image. (b) situations where the structuring element is completely inserted into the foreground, so its center will produce the eroded image.

Figure 5.26(c) is a shrunk version of its original shape. All groups of white pixels in the original image whose areas cannot be contained within the structuring element are removed by the erosion. As seen in Figure 5.26(d), the dilation expands the shape of the objects. The opening in Figure 5.26(e) is the dilation of the shrunk shape. As the erosion removed completely the two smallest structures from the original image, the subsequent dilation increases the remaining structure. The opening can be interpreted as an attempt to remove small objects while preserving their original size and shape. The results of the closing operation, as illustrated in Figure 5.26(f), represent an erosion of the dilated structure shown in (d). The closing operation, as its name suggests, is used to fill holes in the image, while trying to preserve the size and shape of the structures in the image.

In the context of this work, our goal is to design the MM so that it can remove residual false positives produced by the DM, such as the ones shown in Figure 5.22. The opening and closing morphological operations applied consecutively in the binary image produced by the DM meet our needs. The opening can be applied to remove false positive pixels without affecting drastically the shape of the original object, and the closing operation can be used to fill possible holes in the produced blobs. As the sizes of the blobs vary, the radius of the structuring element must be learned from training and validation sets. One option is to adjust the radius of the structuring element in the MM as a hyperparameter during validation. However, more powerful architectures might be obtained if such radius could be a learnable parameter during the training phase, using, for example, the backpropagation algorithm. Unfortunately, as pointed out above, neither the erosion nor the dilation operators are differentiable with respect to their structuring element radii, which makes them unsuitable for being learned in a neural network architecture.

One of the contributions of our work is the approximation of morphological

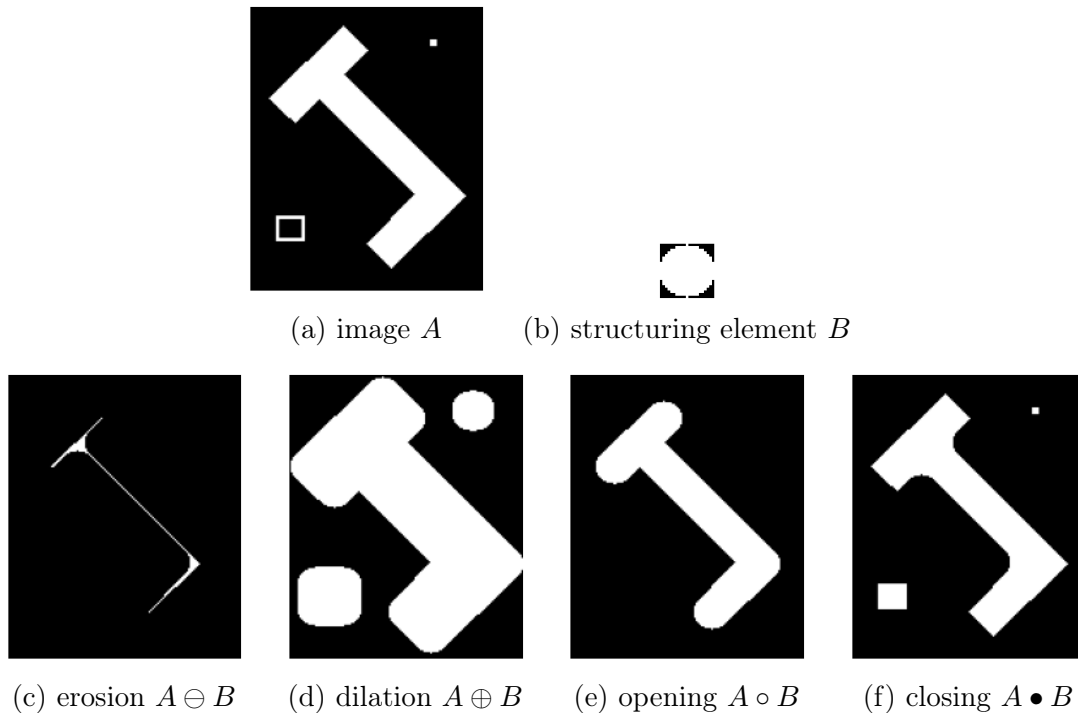


Figure 5.26: Examples of results obtained with four morphological operations applied in A (a) using a circular 21×21 structuring element B (b).

operations using operators that are differentiable with respect to the structuring element size. This way, the sizes of the structuring elements could be learned inside a neural network, for example. Our strategy to obtain such approximations of morphological operations consists in expressing them as a function with the following characteristics:

- (i) It should be parameterizable by the radius structuring element.
- (ii) It should be differentiable with respect to this radius.
- (iii) The pipeline used to approximate the function must be capable of back-propagating gradients in a neural network pipeline.

In the sequel we detail a new approach that allows one to obtain functions with characteristics (i) for approximating erosion and dilation operations.

Morphological Operations as Functions Parameterized by Their Radii

We start by approximating the erosion operation by using a convolution followed by thresholding. The dilation is obtained from the erosion using the property expressed in Equation (5.21). The process to approximate the erosion by a convolution followed by thresholding can be better understood by analyzing how a one-dimensional

erosion can be obtained. In the binary one-dimensional mathematical morphology, the structuring element $h(x)$ is a pulse of width Δ , defined by the equation below:

$$h(x) = \begin{cases} 1, & \text{if } -\frac{\Delta}{2} \leq x \leq \frac{\Delta}{2} \\ 0, & \text{otherwise.} \end{cases} \quad (5.26)$$

Likewise, an object $f(x)$ of width L that extends from x_{\min} to $x_{\max} = x_{\min} + L$ can be represented by the pulse given by

$$f(x) = \begin{cases} 1, & \text{if } x_{\min} \leq x \leq x_{\max} \\ 0, & \text{otherwise.} \end{cases} \quad (5.27)$$

In the case of the erosion operation, one has to identify the regions of the x axes where the structuring element $h(x)$ is totally contained inside the set where $f(x) = 1$. The discrete convolution given by the equation

$$g(x) = f(x) * h(x) = \sum_{x'=-\infty}^{\infty} f(x')h(x-x'). \quad (5.28)$$

Figure 5.27 illustrates the convolution process between a structuring element $h(x)$ and $f(x)$. The convolution result represents the intersection area between both signals, and its highest possible value occurs when the structuring element is completely inserted into the foreground.

The expression of the convolution is

$$g(x) = \begin{cases} 0, & \text{if } x \leq x_{\min} - \frac{\Delta}{2} \\ x - (x_{\min} - \frac{\Delta}{2}), & \text{if } x_{\min} - \frac{\Delta}{2} < x \leq x_{\min} + \frac{\Delta}{2} \\ \Delta, & \text{if } x_{\min} + \frac{\Delta}{2} < x \leq x_{\max} - \frac{\Delta}{2} \\ -x + x_{\max} + \frac{\Delta}{2}, & \text{if } x_{\max} - \frac{\Delta}{2} < x \leq x_{\max} + \frac{\Delta}{2} \\ 0, & \text{otherwise,} \end{cases} \quad (5.29)$$

where Δ represents the area of the structuring element $h(x)$. From Figure 5.27, one can see that the structuring element is totally inserted into the background when the convolution result is equal to the area of the structuring element, which is the case when $g(x) = \Delta$, as seen in Equation 5.29. Such verification can be done with a hard thresholding function $v(x)$, so that

$$v(x) = \begin{cases} 0, & \text{if } g(x) < \Delta - \epsilon \\ 1, & \text{otherwise.} \end{cases} \quad (5.30)$$

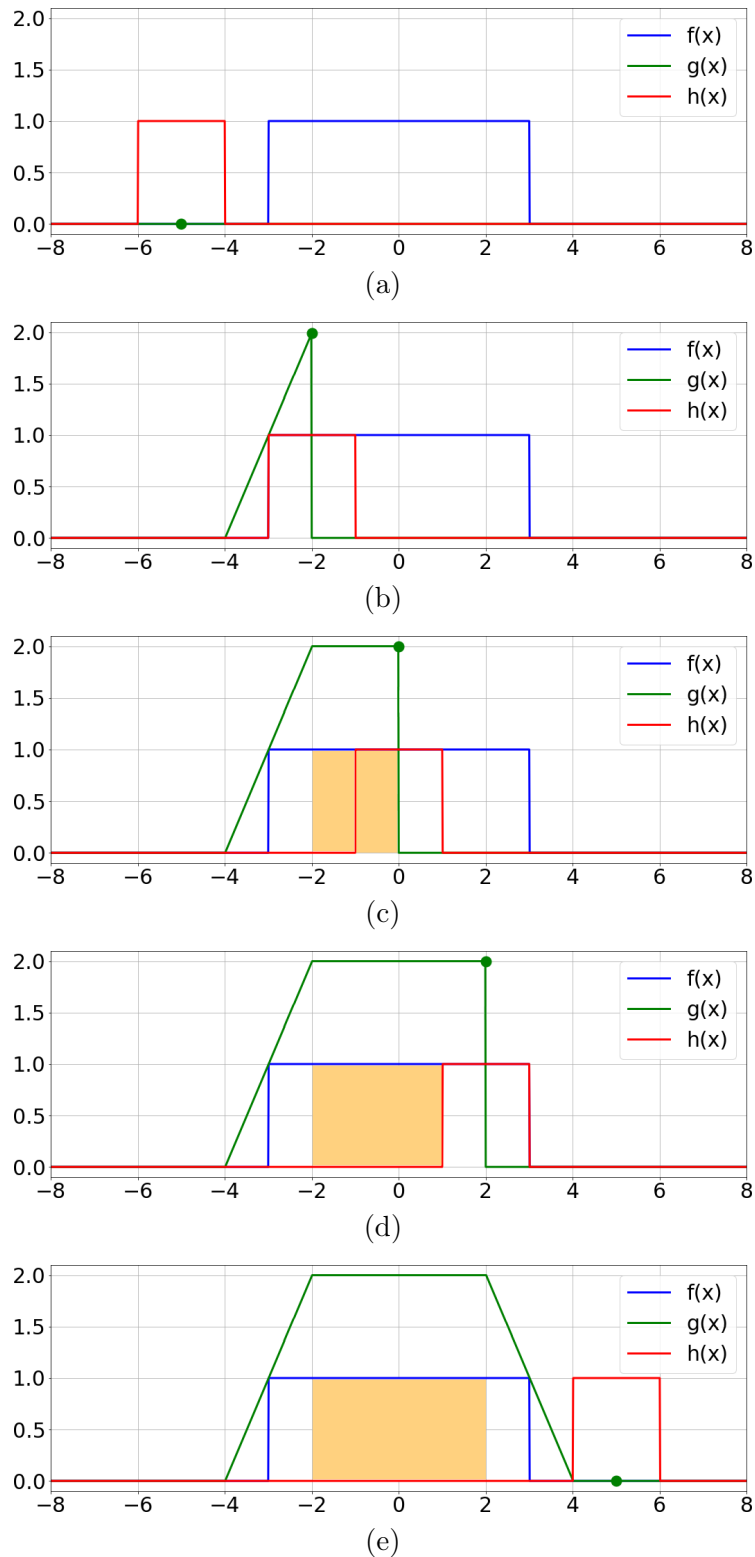


Figure 5.27: Example of convolution $g(x) = f(x) * h(x)$. The convolution result $g(x)$ (green) is computed with Equation 5.29 considering $x_{\min} = -3$, $x_{\max} = 3$ and $\Delta = 2$. By moving the structuring element $h(x)$ (red), whose area is Δ , the convolution result $g(x)$ represents the intersection area between $h(x)$ and $f(x)$. When $h(x)$ is completely inserted into $f(x)$, as in (b), (c) and (d), $g(x) = \Delta = 2$, which is the area of $h(x)$. The orange area represents the regions where $g(x) = \Delta = 2$.

This way, the desired erosion operation of a binary foreground $f(x)$ with a structuring element $h(x)$ is approximated by the convolution $g(x) = f(x) * h(x)$ followed by a hard thresholding function $v(x)$, as in Equation (5.30). Larger values of ϵ produce larger versions of the eroded $f(x)$, as depicted in Figure 5.28. The parameter ϵ can be seen as a tolerance value that is used to control when the structuring element starts being considered as totally inserted into $f(x)$.

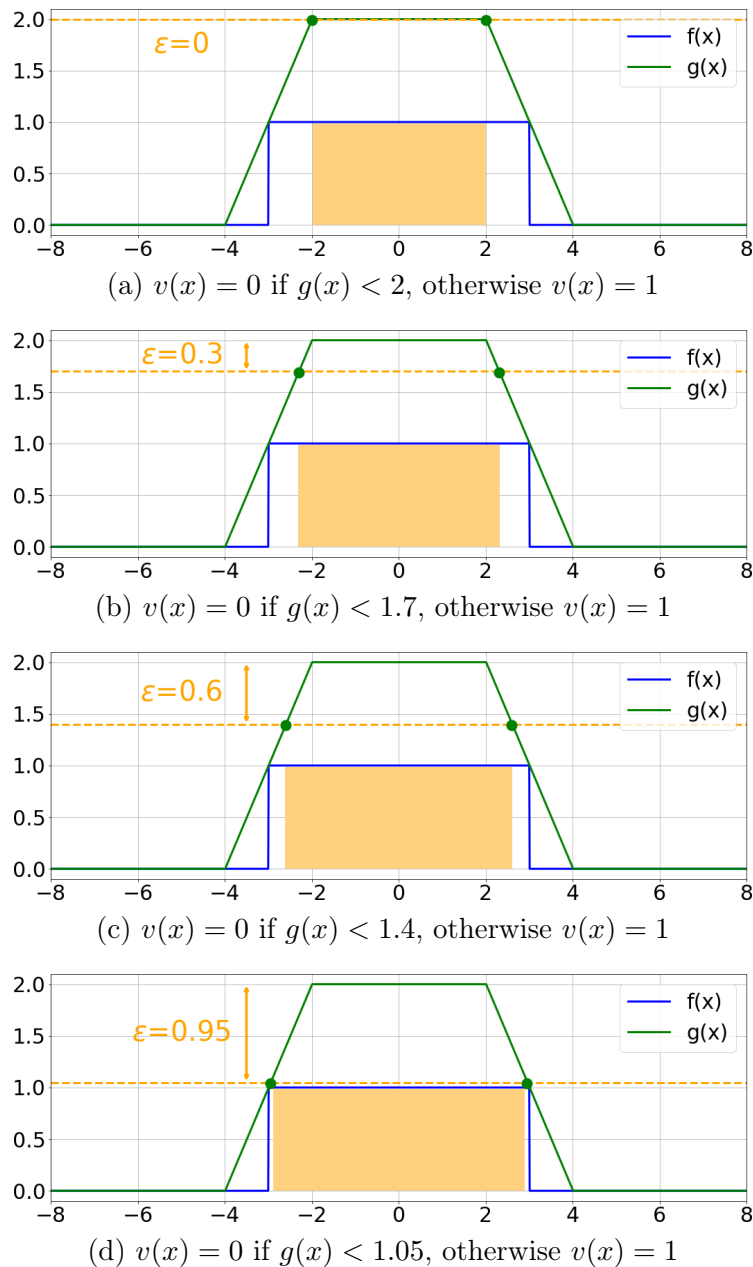


Figure 5.28: Erosion results obtained with convolution followed by thresholding with different ϵ values. The results of the thresholding operation, computed with Equation (5.30) and represented by the orange areas, correspond to the eroded $f(x)$.

Following the duality of erosion and dilation, we can express the dilation of the foreground $f(x)$ with the structuring element $h(x)$ by computing the erosion of the

complemented $f(x)$ with $h(x)$, and compute the complement of the obtained result, similarly as shown in Equation (5.21). Therefore, the dilation is determined by applying the convolution $g(x) = (1 - f(x)) * h(x)$, followed by the inversion of the result produced by the thresholding function $v(x)$. Figure 5.29 illustrates the dilation operation by means of a convolution followed by thresholding considering $\epsilon = 0$. Note that the gray area in Figure 5.29(e) represents the dilation of $f(x)$ computed with the inversion of Equation (5.30), which is $v(x) = 1$ if $g(x) < \Delta$ and $v(x) = 0$, otherwise.

Figure 5.30 shows dilation results obtained with different values of ϵ , used to reduce the area of the dilated foreground. Note that differently from erosion, a higher ϵ value produces a smaller dilated area of the foreground.

All things considered, as we are dealing with images (2D discrete signals), including one extra dimension into the discrete 1D convolution represented in Equation (5.28), the 2D discrete convolution can be expressed as

$$g(x, y) = f(x, y) * h(x, y) = \sum_{x'=-\infty}^{\infty} \sum_{y'=-\infty}^{\infty} f(x', y')h(x - x', y - y'), \quad (5.31)$$

and the thresholding in Equation (5.30) now becomes

$$v(x, y) = \begin{cases} 0, & \text{if } g(x, y) < V - \epsilon \\ 1, & \text{otherwise.} \end{cases} \quad (5.32)$$

Note that in the 2D case, Equation 5.32 replaces the area Δ of the structuring element by the volume V of the 2D discrete structuring element.

Similarly to the 1D case, the structuring element $h(x, y)$ can be designed to have any desired shape.

A straightforward way to create a 2D structuring element from a generic 1D function $h(x)$ is to shift $h(x)$ to $x = R$ and rotate it around the origin. One can achieve this by using the unit step function $u(x)$, yielding the two-dimensional structuring element $h_R(x, y)$ below:

$$h_R(x, y) = 1 - u\left(\sqrt{x^2 + y^2} - R\right), \quad (5.33)$$

as illustrated in Figure 5.31. Equations (5.31), (5.32) and (5.33) define a pipeline for the approximation of a morphological erosion operation by the convolution of a function $h_R(x, y)$ that is parameterized by the radius R . It thus satisfies characteristic (i) above. However, since Equation (5.33) relies on a unit step function $u(x)$ that is not differentiable, the pipeline does not satisfy characteristic (ii). In addition, since Equation (5.32) implements a hard threshold that is not able to back-

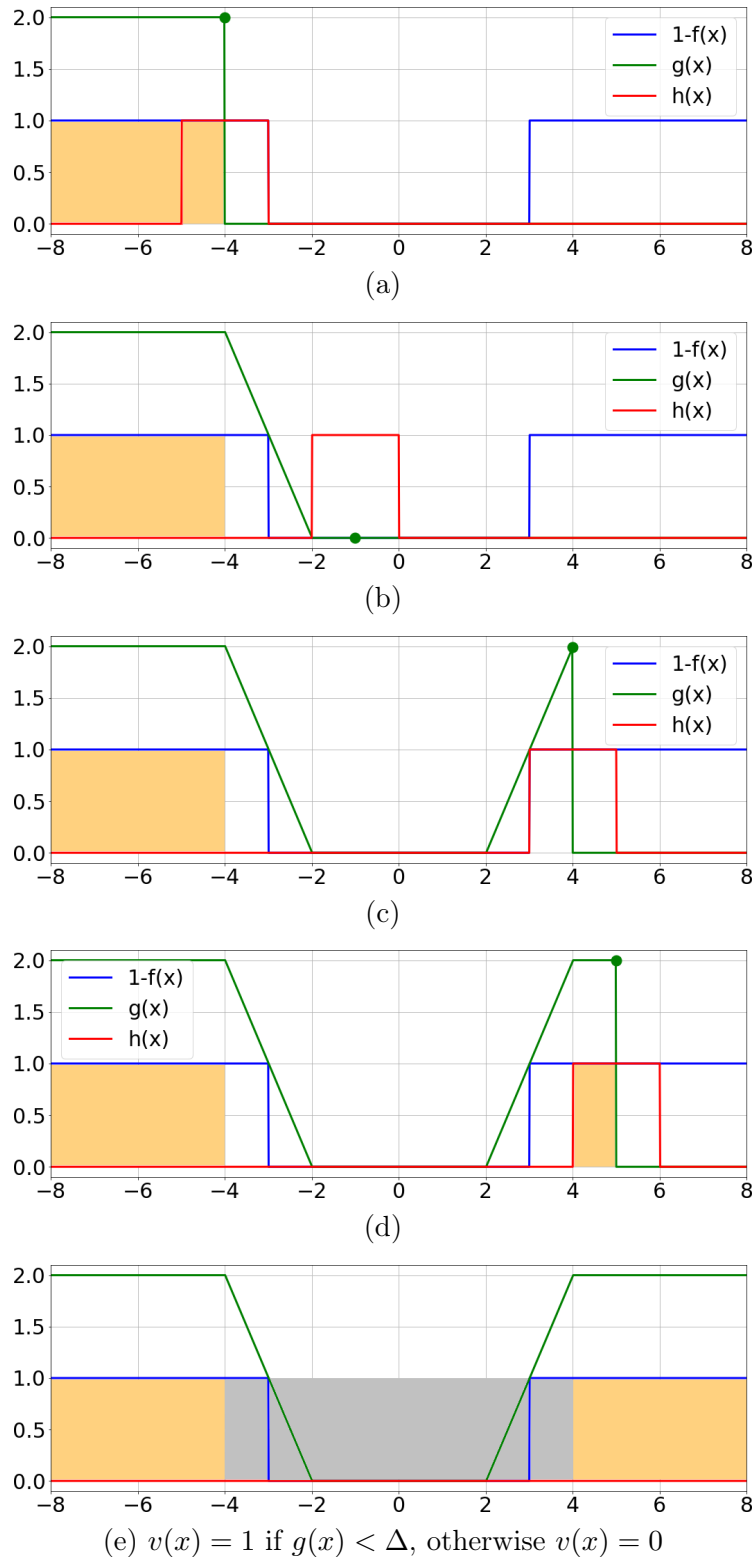


Figure 5.29: Example of convolution $g(x) = (1 - f(x)) * g(x)$ with $x_{\min} = -3$, $x_{\max} = 3$ and $\Delta = 2$. In this example, $\epsilon = 0$. The convolution is now applied in the inverted foreground $1 - f(x)$ producing $g(x)$ (green). By moving the structuring element $h(x)$ (red), the convolution result represents the intersection area between $h(x)$ and $1 - f(x)$. In the final result, shown in (e), the orange areas represent the regions where $g(x) = \Delta = 2$. The remaining area, colored in gray, represents with 1 the region where $v(x) < \Delta$, and corresponds to the dilated foreground $f(x)$.

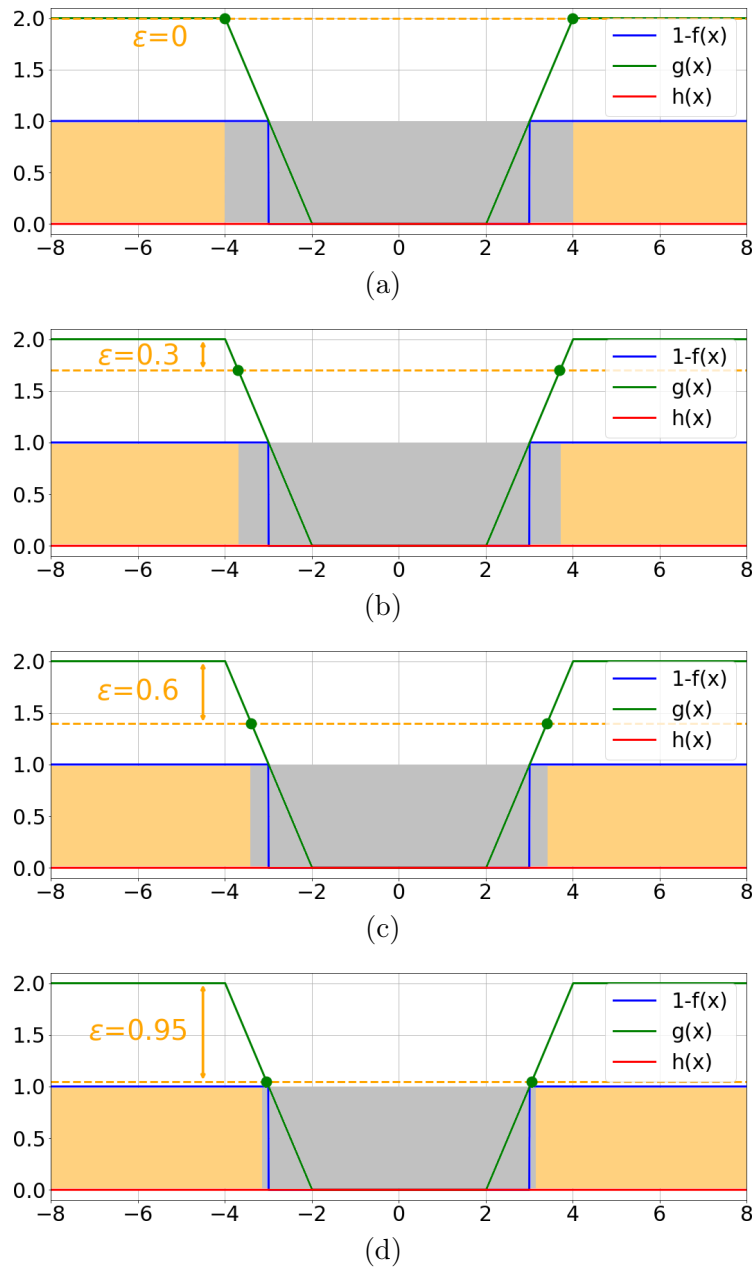


Figure 5.30: Dilation results obtained with convolution followed by thresholding with different ϵ values. By computing the convolution $g(x) = (1 - f(x)) * h(x)$ followed by a thresholding function $v(x, y)$ with Equation (5.30) using different ϵ values, the dilation of $f(x)$ is produced, and are represented by the gray regions.

propagate gradients when used in a neural network, the pipeline also does not satisfy characteristic (iii). In the sequel, we introduce a way to approximate morphological operations that also satisfies characteristics (ii) and (iii).

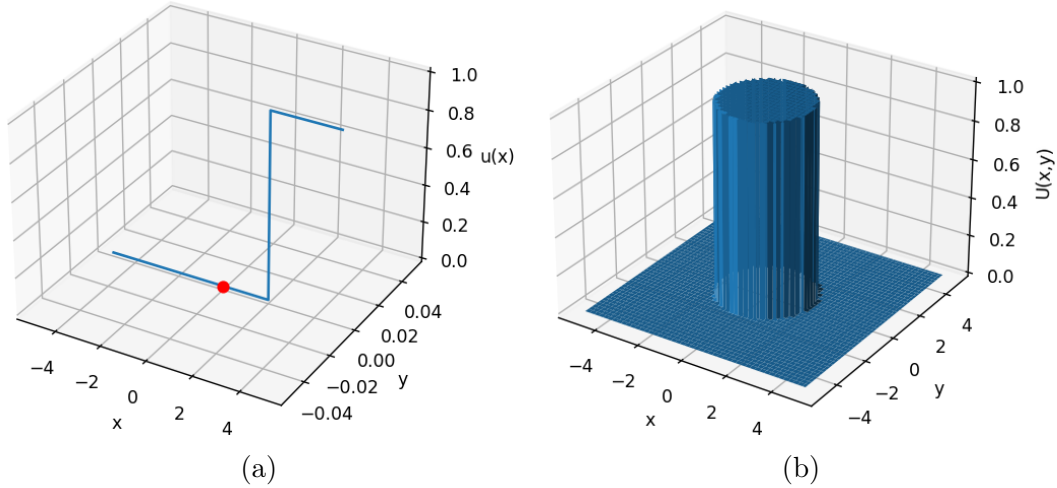


Figure 5.31: Rotation of the step function shifted to $x = R$ (a) around the point $(0, 0)$ (red dot) using Equation (5.33) produces (b).

Using Sigmoid Function to Obtain Morphological Operations that Are Differentiable with Respect to Their Radii and Can Back-Propagate Gradients

As pointed out above, the proposed solution generates an approximation of a morphological erosion that is neither differentiable with respect to its radius nor can back-propagate gradients in a neural network. This is so due to the presence of the step function in Equation (5.33) and the hard threshold in Equation (5.32), that are not differentiable. In order to generate a learnable morphological erosion structure, both the structuring element $h_R(x, y)$ and the thresholding function $v(x, y)$ should be differentiable.

In this work we employ sigmoid functions, that are differentiable everywhere, to replace the unit step function $u(x)$ in the structuring element $h_R(x, y)$ (Equation (5.33)) and the thresholding function $v(x, y)$ (Equation (5.32)).

The sigmoid function, previously illustrated in Figure 5.18 and presented in Equation (5.6), rewritten as Equation (5.34), is fully differentiable and its slant (derivative at $x = 0$) is controlled by a hyperparameter γ :

$$s(x) = \frac{1}{1 + e^{-\gamma x}}. \quad (5.34)$$

The structuring element $h_R(x, y)$ in Equation (5.33) can be differentiable with respect to R by replacing the unit step $u(x)$ by the sigmoid $s(x)$, yielding

$$h_R(x, y) = 1 - s\left(\sqrt{x^2 + y^2} - R\right). \quad (5.35)$$

Figure 5.32 illustrates structuring elements produced with different values of γ ,

which influences the slant of the sigmoid, and thus the smoothness of the circumference. The higher the γ , the closer the structuring element becomes to a solid cylinder.

To approximate the erosion of a binary image $f(x, y)$ by applying a convolution followed by a thresholding, Equation (5.31) is used to compute the convolution $g(x, y)$ considering $h(x, y) = h_R(x, y)$, the circular structuring element that is differentiable with respect to R . In order for the gradients to back-propagate if this approximation of the erosion operation is used in a neural network, the thresholding function $v(x, y)$ from Equation (5.32) must also be differentiable. To this end, we also used the sigmoid function $s(x)$ from Equation (5.34) as the thresholding function, yielding

$$v(x, y) = s(g(x, y) - (V - \epsilon)), \quad (5.36)$$

where $s(x)$ is the sigmoid function defined in Equation (5.34), V is the volume of the structuring element and ϵ is used to control how early the structuring element is totally inserted into the foreground, as in the 1D case.

The goal of the Morphology Module (MM) is to learn the radii of circular structuring elements to perform opening and closing operations in sequence to eliminate false positive pixels while keeping the shape of the anomalous object (represented by white blobs as in Figure 5.37). Although circular structuring elements produced in a 41×41 grid using very high values of γ do not change its appearance and shape, and do not influence the convolution results, the value of γ affects directly the computation of gradients concerning the radius, as seen in Equation (5.37).

$$\frac{\partial h(x, y)}{\partial R} = \gamma h(x, y)(1 - h(x, y)). \quad (5.37)$$

Note that the larger the value of γ , the better the sigmoid function $s(x)$ approximates the unit step function $u(x)$. However, a very large γ may generate numerical problems in the propagation of the gradients. A good compromise was found empirically to be $\gamma = 5$, as it avoids the problem of vanishing the local gradient and works well for kernels with resolution 41×41 , which can fit circular kernels with radius $1 < R < 20$, as shown in the examples in Figure 5.33.

The 2D convolution represented in Figure 5.34 is implemented using the differentiable approximation analyzed in this section, that uses the convolution with a sigmoidal structuring element with a binary object. It is noted that when the structuring element is completely inserted into the object, the convolution result represents the intersection volume between the structuring element and the object, being within the interval $[0, V]$, where V is the volume of the structuring element. But in practice, as the structuring element is not a complete binary structure due

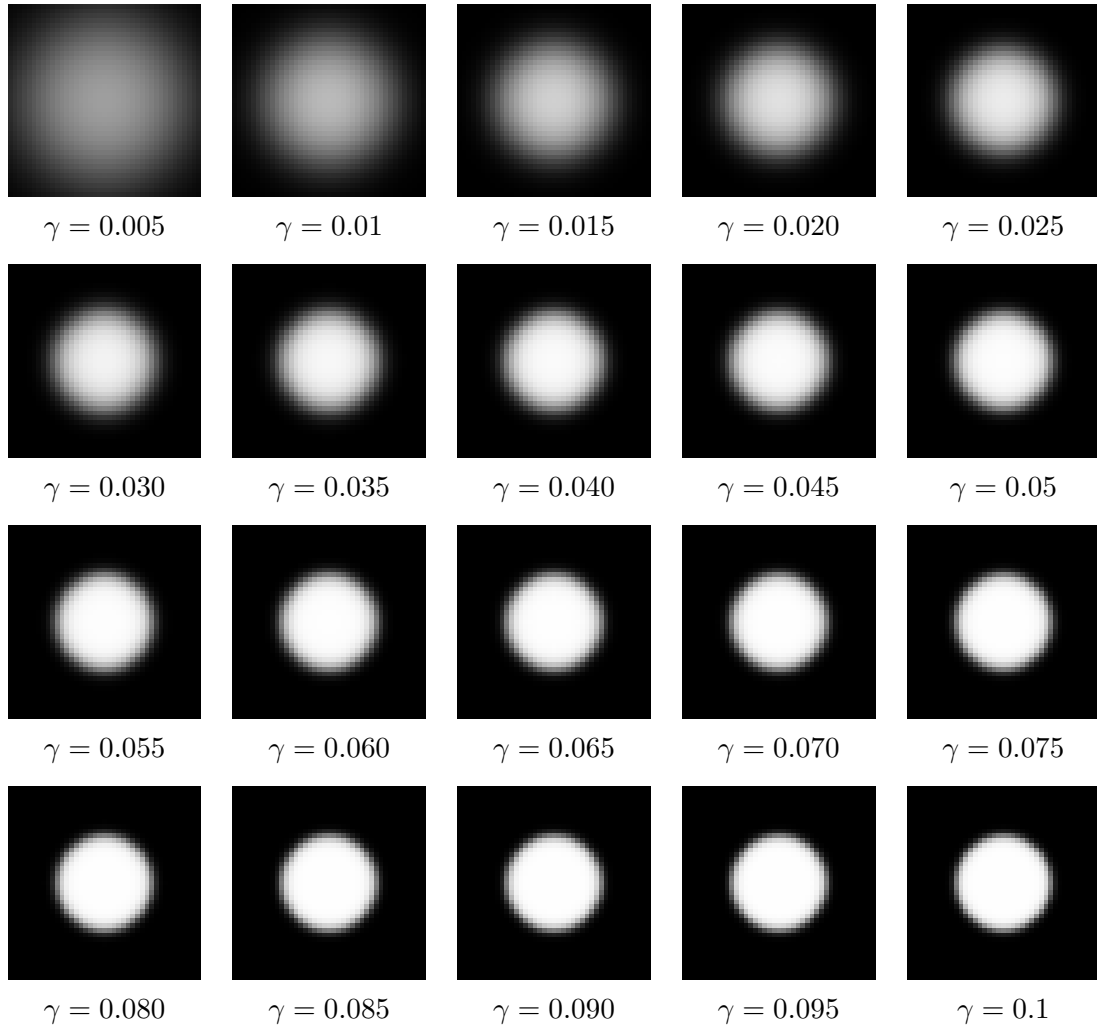


Figure 5.32: 41×41 circular kernels generated with Equations (5.6) and (5.33) with different values of γ and fixed $R = 10$.

to the selected value of $\gamma = 5$, the output of the convolution is within an interval $[0, V - \epsilon]$, where ϵ is an empirical value used to disregard 8% of the structuring element volume (see Equation (5.36)). The value $\epsilon = 0.08V$ works well for all sigmoidal structuring elements generated with $\gamma = 5$. The value of $\epsilon = 0.08V$ means that the erosion operation will output a “foreground” (high) value whenever more than 92% of the structuring element’s volume is totally inserted into the foreground. In our experiments this threshold is fixed and applied with a sigmoid function according to Equation (5.32) with $\gamma = 10$ and volume = $0.92V$. Using this approach, it is possible to obtain at the output a binary-like image whose high level pixels represent regions where 92% of the structuring element is inserted into the object. This is equivalent to a binary erosion performed with a structuring element of 8% less area than the original structuring element.

Due to the dual relation between erosion and dilation in Equation (5.21) (whose 1-D approximation is illustrated in Figure 5.30), the morphological operations dil-

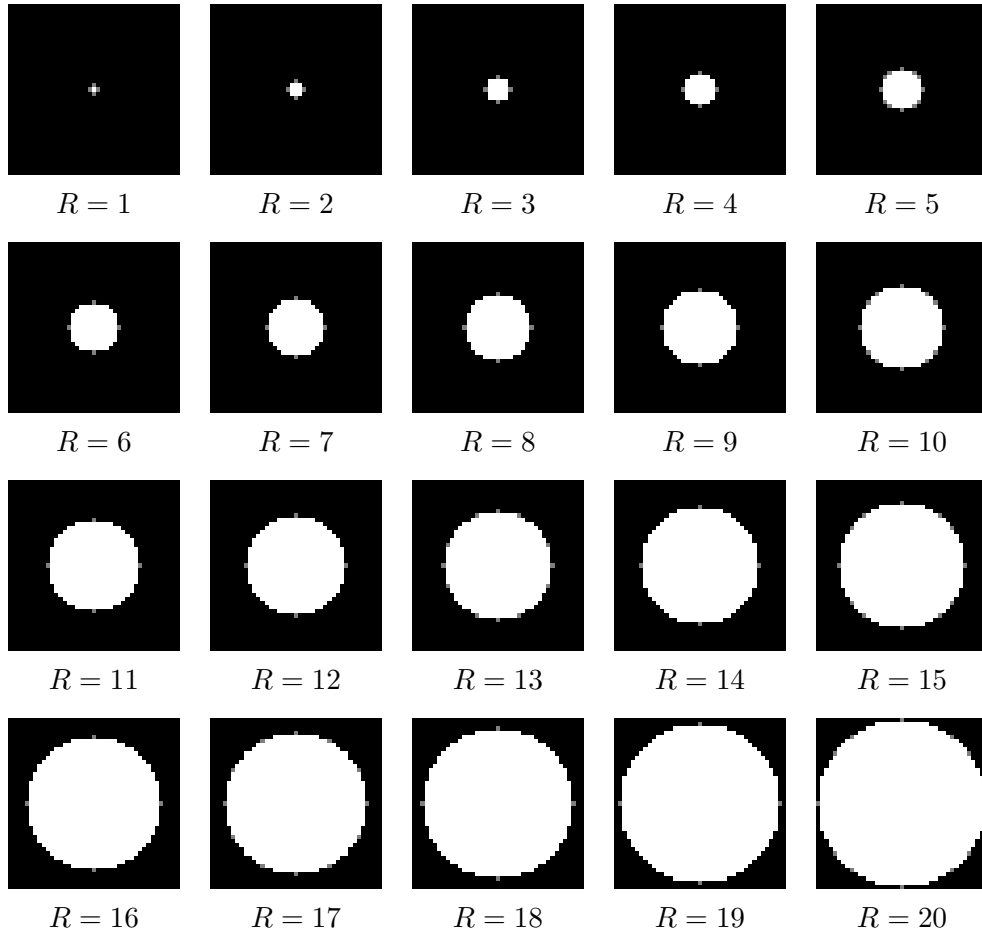


Figure 5.33: All possible 41×41 circular structuring elements to be learned by the MM generated with Equations (5.6) and (5.33) considering $\gamma = 5$ and different values of R .

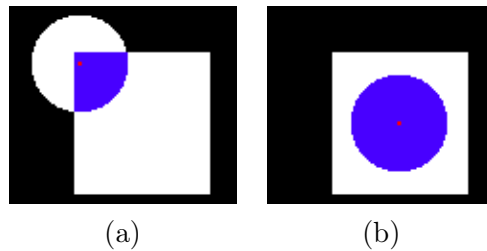


Figure 5.34: Illustration of convolutions applied to a binary image f containing a square object with a circular binary kernel h , whose center $g(x_c, y_c)$ is represented by the red pixel. The convolution is computed with Equation (5.31), and the resulting value in (x_c, y_c) represents the intersection area between the kernel h and the object, highlighted in blue. It is noted in (b) that when the kernel is completely inserted in the object, the convolution result in (x_c, y_c) represents the volume of the circular kernel.

tion, opening and closing can be executed with complement, convolution and thresholding operations. Figure 5.35 illustrates the combination of such operations when applied to a given input image. One can readily see that it produces similar results

with the ones obtained with the binary morphological operations from Figure 5.26.

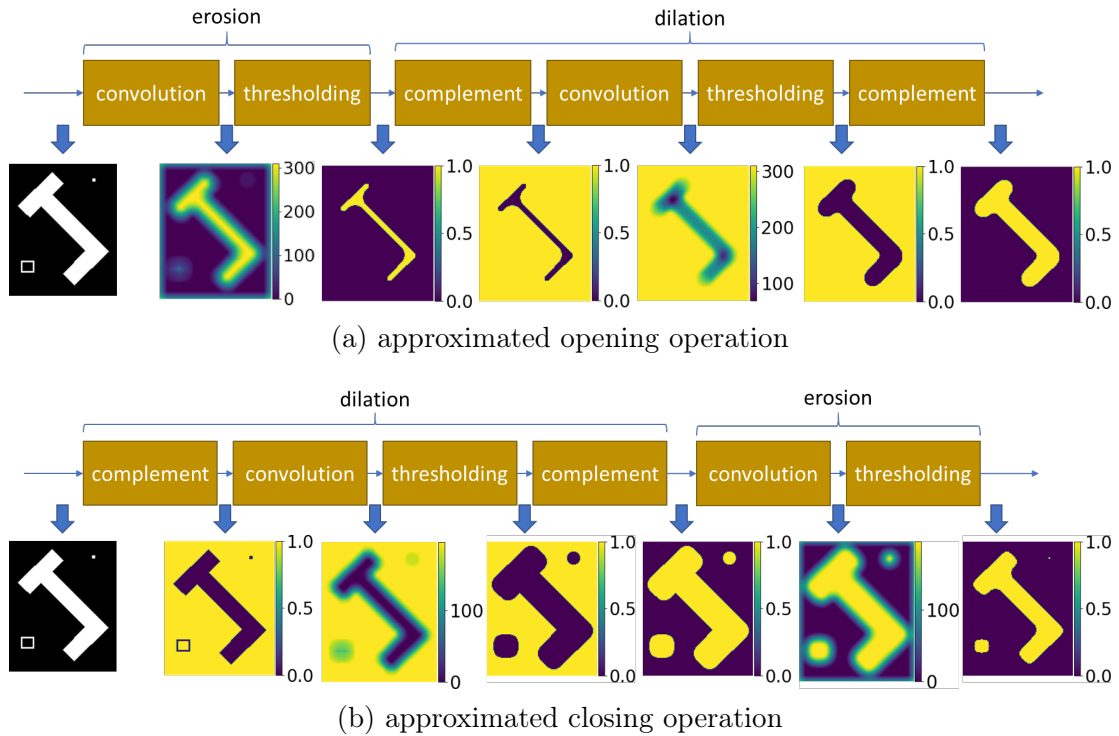


Figure 5.35: Approximated opening (a) and closing (b) operations performed by combining convolutions and thresholding operations. The convolutions were performed with a sigmoidal structuring element generated with $\gamma = 5$ and $R = 8$ (see Figure 5.33). The threshold operations were executed with sigmoid functions produced with Equation (5.6), considering $t = 0.95V$, being V the structuring element volume, $\gamma = 10$ for the erosion thresholds, and $\gamma = 5$ for the dilation thresholds.

In brief, since the erosion operation can be approximated by a convolution followed by a thresholding operation, and the opening and closing operations can be produced in terms of the erosion, using Equations 5.24 and 5.25, respectively, then opening and closing can also be approximated by convolutions and thresholding operations.

Another critical problem in designing differentiable opening and closing operations is that when binary morphological operations are used, there are cases in which the output of an opening or closing operation is independent of the structuring element radius. Supposing one could implement a differentiable erosion operation that would be exactly equivalent to the binary case, if such a situation happens during the training with a neural network, the derivatives with respect to the structuring elements would be zero, and the radius value would not be learned. Figure 5.36 illustrates binary morphological operations where the same foreground is being closed (dilation followed by erosion) with structuring elements of different radii. Note that the results produced by the dilation clearly vary according to the structuring element radius. Nevertheless, when the erosion is applied in sequence, the dilated

foreground returns to its original format. The same behavior could be observed in the opening operation. This clearly illustrates that, when using binary morphological operations, there are cases where the derivative with respect to the radius of the closing operation is zero. It is important to note that the proposed differentiable approximations of morphological operations naturally address this problem by using sigmoid functions for creating the structuring elements. Due to its smooth dome-like shape, the sigmoidal structuring element is not only differentiable regarding the equation that generated it, but also prevents opening and closing operations to produce the same results with different radii.

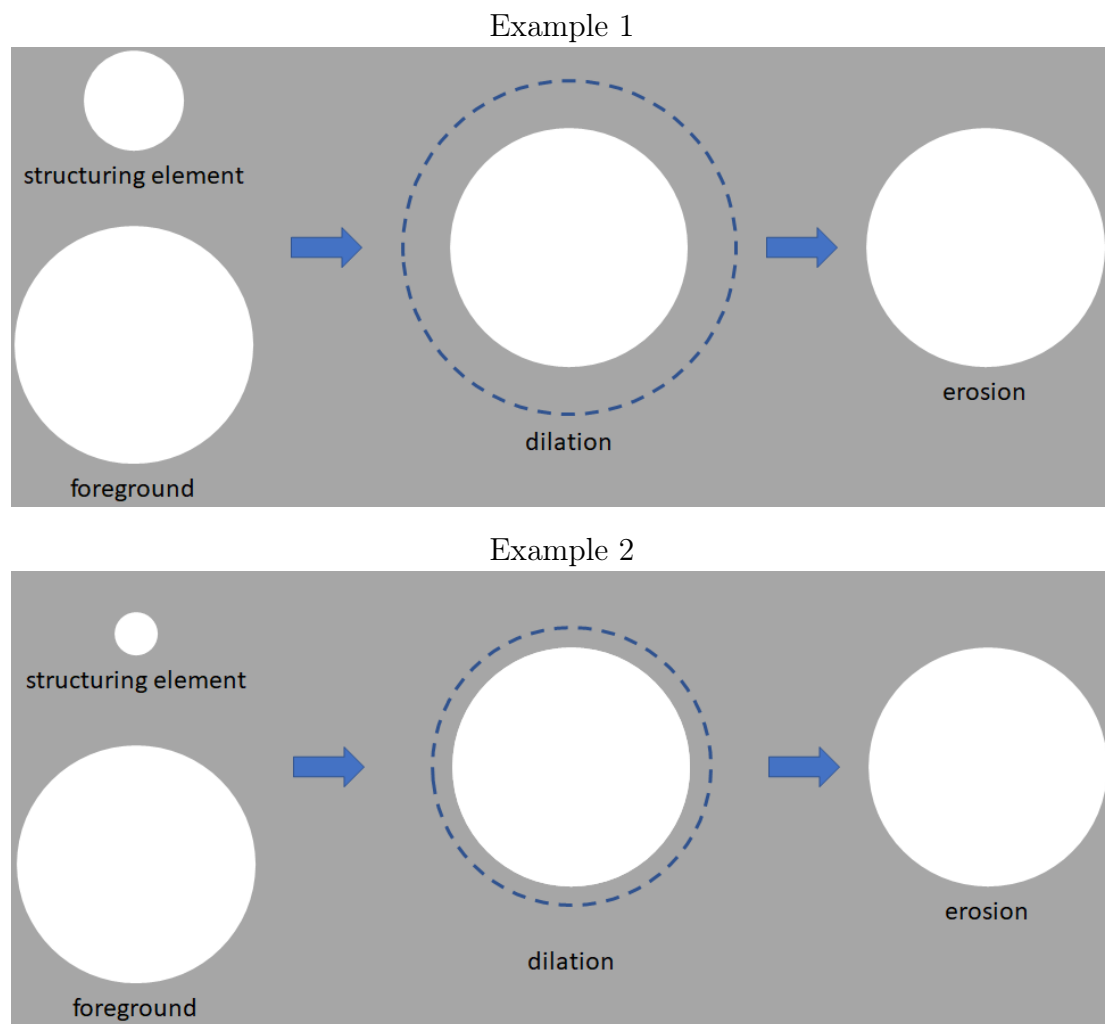


Figure 5.36: Dilation followed by erosion of a foreground with structuring elements of different radii. In Example 1, the structuring element has a larger radius than the structuring element in Example 2, and the foregrounds are identical in both examples. The dashed shape represent the dilated area with its respective structuring element. Note that eroding the dilated foreground produces the same original foreground image.

Examples of morphological operations produced by the trained differentiable morphology module (MM) can be seen in Figure 5.37. Note the small blobs in

the input images are removed with the differentiable opening operation while their shapes are sustained. The images in Figure 5.37 (b) are transformed with the differentiable closing operation and the wholes are filled. Note that as the ground-truth blobs used to train the MM are rectangular bounding boxes, the closed images tend to deform the blob, as it is not optimized based on the object original shape.

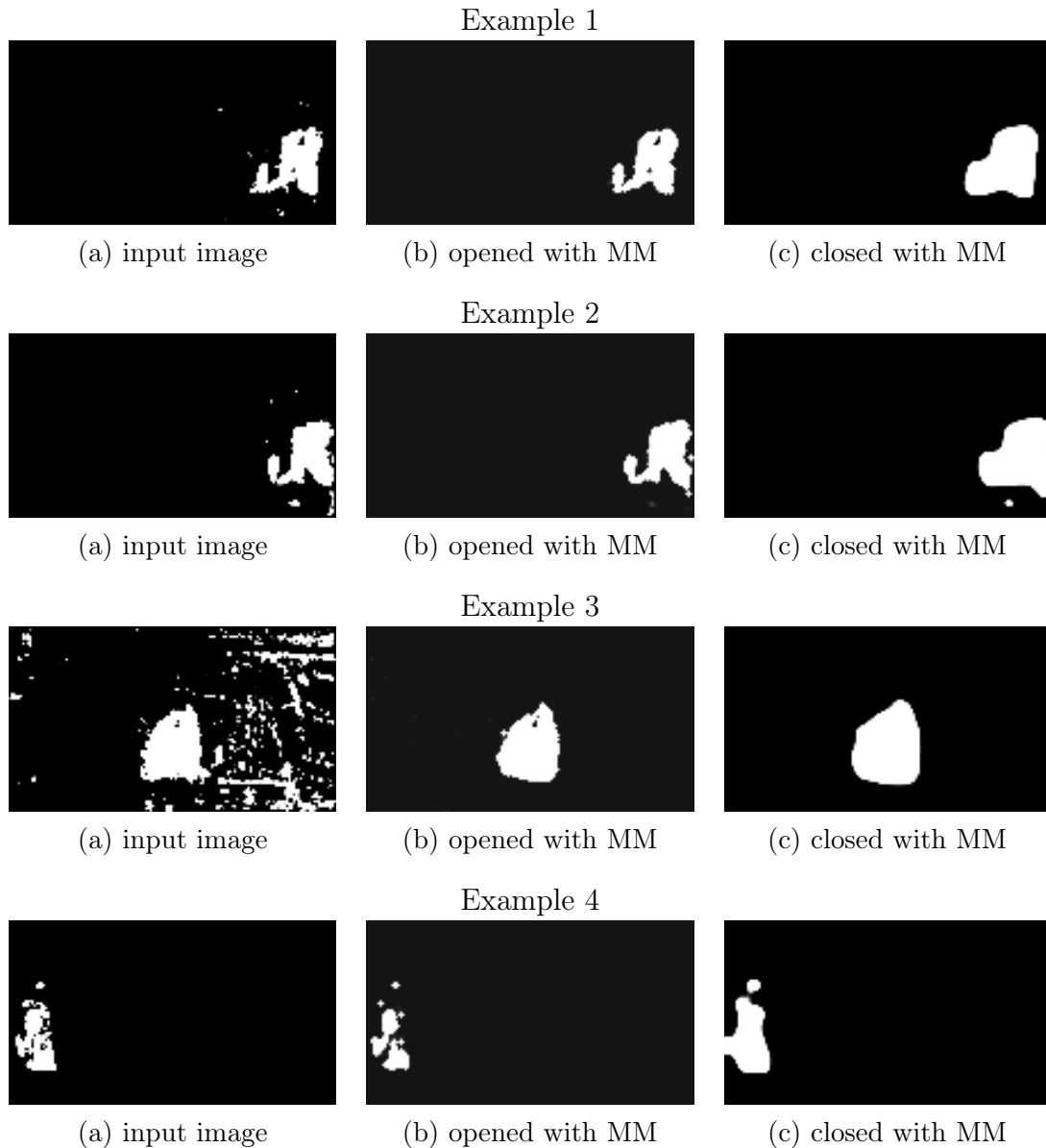


Figure 5.37: Examples of frames transformed with the differential morphology operations. Images (a) are input images. They are then transformed with the differentiable opening operation, producing images (b), which are transformed with closing differentiable operation, producing images (c). Note that most of the noise in (a) are removed with the opening operation. The holes in (b) are filled after the closing operation.

In the next section, the module responsible to classify the closed images as anomalous or non-anomalous is presented.

5.6 Classification Module (CM)

The goal of the proposed pipeline is to classify the aligned pair of reference and target frame. All previous modules (DM, TCM and MM) produce images with blobs representing the location of the anomalous objects. Along the pipeline, each module has a predefined task so that false positive regions are eliminated and false negative regions are incorporated into the blob.

The classification module (CM) considers the amount of pixels whose values are not 0 by summing all pixel values within the input image, dividing it by the amount of pixels in the image (width \times height). In other words, the CM evaluates the fraction of the image represented by blobs. Thus, an image without any blob will produce 0 and an image dominated by blobs (white pixels), it will produce 1. This fraction of the image represented by blobs is then thresholded with a sigmoid function generated with Equation (5.6), enabling the threshold value to be learned by means of the backpropagation of the computed gradients. The value $\gamma = 5,000$ was empirically chosen, producing a sigmoid very close to a step function, but still able to allow the backpropagation of the gradient. The role of the CM is similar to the thresholding function represented by Equation 5.32, but now instead of evaluating the $g(x, y)$, it considers the fraction of the image represented by blobs and the value of ϵ is set to 0.

In a perfect scenario, where the input image of the CM does not contain false positive pixels, it is expected to learn a threshold very close to 0, as it will be able to distinguish images with very small anomalous objects. In the VDAO database, with the movement of the robot, when the object is entering or leaving the scene, it usually produces very small blobs, that is the reason why the learned threshold represents a small value.

5.7 Training, Validating and Testing

The procedure to train, validate and test the network proposed in this work is described in this section.

Differently from the approaches covered in previous chapters, the training process of the network proposed in this chapter (see the pipeline in Figure 5.1) is done as the usual process of training a regular CNN. Training samples are passed by the network into batches, and after a complete training epoch, a validation set is used to measure the generalization capability of the network. The epoch in which the model produces the lowest loss value in the validation set is considered to have the best set of learned parameters, and thus, they are used to measure the network performance on the testing set.

As discussed in Section 3.5, to avoid contamination between training, validating and testing sets, we split the samples into folds. The division of the target objects in folds is shown in Table 5.1 and the number of samples used in each fold are in Table 5.2. Each fold was trained separately, and thus, for each fold a network with different parameter values is produced.

Table 5.1: Separation of objects into folds used to train, validate and test the network presented in the current chapter.

fold	black backpack	black coat	brown box	camera box	dark-blue box	pink bottle	shoe	towel	white jar
1	validation	testing	training	training	training	validation	training	training	validation
2	testing	training	validation	training	training	training	validation	training	validation
3	training	training	testing	validation	validation	training	training	validation	training
4	training	training	validation	testing	validation	training	validation	training	training
5	validation	validation	training	validation	testing	training	training	training	training
6	training	training	training	training	validation	testing	validation	training	validation
7	training	validation	validation	validation	training	training	testing	training	training
8	validation	training	training	validation	training	validation	training	testing	training
9	training	validation	training	training	training	validation	training	validation	testing

Table 5.2: Number of training, validation and testing samples per fold.

fold	training samples	validation samples	testing samples
1	11,818	4,623	1,206
2	12,498	3,618	2,010
3	12,636	3,618	1,206
4	13,616	3,618	1,206
5	10,862	4,422	1,206
6	14,862	3,618	1,407
7	12,118	3,618	1,206
8	11,680	4,623	1,206
9	12,384	3,819	1,206

During training, batches are expected to be balanced, having the same amount of anomalous and not anomalous pair of frames. Nevertheless, the selection of the best voting window performed by the TCM requires the analysis of consecutive pairs of frames, which does not guarantee balanced batches. To overcome such problem, the network was trained in parts, so each module is trained separately one epoch at a time.

Even though the γ value of the sigmoid function responsible for classification in the DM is adjusted to permit the existence of local gradients, training all modules at once produced a lot of instability. The gradient computed by means of the classification loss is vanished when back-propagated until reaching the MM. Training networks in parts or setting a training schedule for each module are solutions proposed by different works to avoid the vanishing gradient problem [154–156].

First, the DM is trained for one epoch with batches containing balanced pair of frames. The batch loss is measured with the MSE of the MCC, computed between the output image and a ground-truth binary image created with the bounding box

ground-truth coordinates, as illustrated in Figure 5.13. Next, the TCM will choose the best voting window that obtains the best MCC in consecutive frames for one epoch. For that, the DM parameters are frozen, and batches with 15 consecutive frames are passed by the network and evaluated by the TCM. Then, the MM is trained considering batches of consecutive frames for one epoch. When consecutive frames pass by the TCM, due to its voting processing, a single binary image is produced. Thus, 15 batches, each formed by consecutive frames, are input into the network, and the TCM’s outputs are accumulated until a full batch is completed. This batch is then used to train the MM. The MCC of the output image is computed, and its distance to its optimum value $MCC=1$ is measured with MSE. Finally, the CM is trained for one epoch, similarly as done with the MM: a set of 15 batches, formed by consecutive frames, are sequentially input into the network and accumulated after the TCM, which will produce one full batch of 15 samples. This batch continues its path until the CM classifies the samples. We have tried to use the binary cross entropy loss to measure the loss of the final classification, but it was very unstable. As the MSE proved to be more robust, the classification error is computed with MSE. A complete training epoch is accounted when all modules are trained considering all samples. For each fold, 100 training epochs were performed.

We used the Adam optimizer [157] for training with hyperparameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1 \times 10^{-8}$, and initial values and learning rates were adjusted according to each parameter as shown in Table 5.3.

Table 5.3: Parameters and learning rates used to train the proposed network.

modules	parameters	initial values	learning rates
DM	reference weights $(w_{r_1}, w_{r_2}, \dots, w_{r_{256}})$	$[1, 1, \dots, 1]$	1×10^{-2}
	target weights $(w_{t_1}, w_{t_2}, \dots, w_{t_{256}})$	$[1, 1, \dots, 1]$	1×10^{-2}
	bias $(b_1, b_2, \dots, b_{256})$	$[1, 1, \dots, 1] \times 10^{-2}$	2×10^{-4}
	combination weights $(w_1, w_2, \dots, w_{256})$	$[1, 1, \dots, 1]$	2×10^{-2}
	sigmoid threshold (t)	-2.15	3×10^{-2}
MM	radius opening	1	1×10^{-4}
	radius closing	1	13×10^{-3}
CM	sigmoid threshold	2×10^{-2}	1×10^{-4}

During the training, each module was able to adjust their learning parameters in order to optimize the MSE considering their respective outputs. As explained in Section 5.3, the DM contains 1,025 learning parameters. Figure 5.38 shows the final parameters learned by the DM in epoch 56, which was used in the validation and testing sets. The weights used to multiply the target feature maps in Figure 5.38(a) which have reached the highest values may be associated with the most discriminative feature maps. But it needs further investigation to be confirmed.

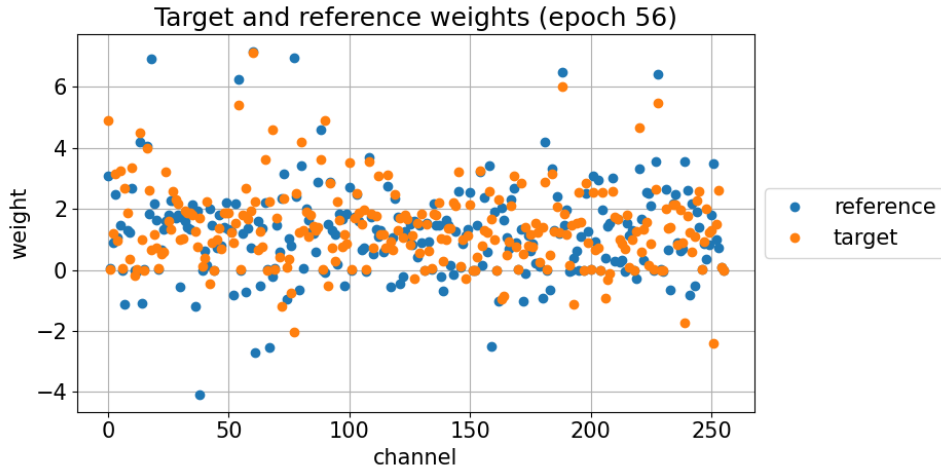
The learnable parameters of the MM are the opening and closing radii of the sigmoidal structuring elements. Figure 5.39 shows the evolution of such parameters

along the training epochs in fold 5. Due to the small false positive blobs, the opening radius stays almost constant in 1, being able to remove blobs with an area close to $3 = \pi \times R^2$. As the blobs in general have irregular shapes and the morphology is optimized considering rectangular bounding boxes, the radius of the closing structuring element tends to reach higher values to grow the blobs, so it reaches areas close to the ground-truth bounding box areas.

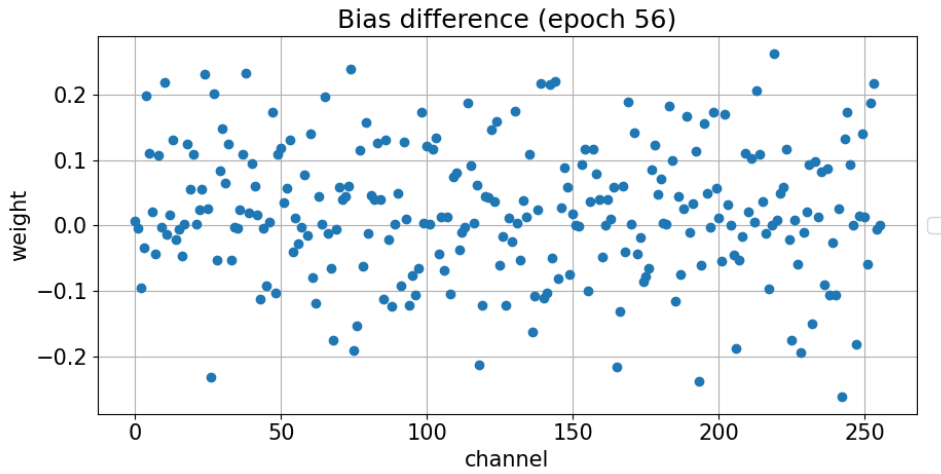
As the CM learns the threshold value responsible to classify the pair of frames as anomalous and not anomalous by verifying the percentage of the input image containing white pixels, it is expected to learn a threshold with low value so it can consider small objects as anomalous. It will indeed fail if the input image contains object blobs larger than blobs formed by residual noise not removed by previous modules. Figure 5.40 shows the classification threshold learned in fold 5 along the training. It stabilizes around 0.068%, which classifies as anomalous, input images containing more than 8 white pixels.

The comparative training losses obtained for each fold in each module are shown in Figure 5.41. It is noted that losses of all modules decrease along the training in all folds, being evident the learning ability of the proposed network. Due to the dependency of outputs of previous modules, the CM takes more epochs to stabilize than the other modules.

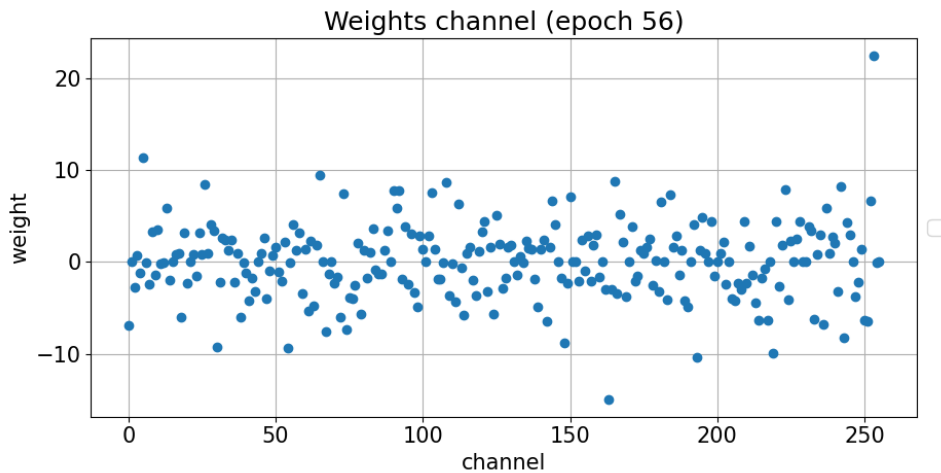
In the next section, the results obtained by each module are presented.



(a) learned weights used to multiply the reference and target feature maps



(b) learned bias ($[b_1, b_2, \dots, b_{256}]$)



(c) learned weights ($[w_1, w_2, \dots, w_{256}]$)

Figure 5.38: Parameters learned by DM during training of fold 5 in the 56th epoch, the one that obtained the best results in the validation set. These parameters are distributed in the DM as illustrated in Figure 5.17. The final parameter t achieved the value -2.86 in epoch 56.

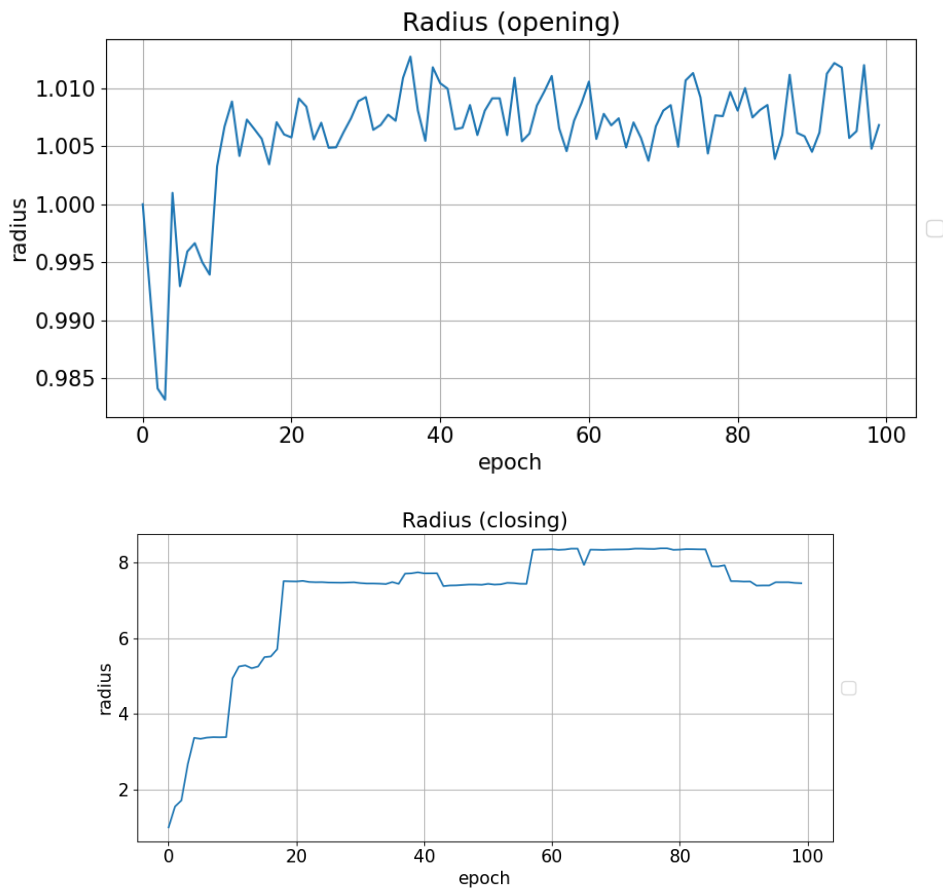


Figure 5.39: Parameters learned by MM during the training of fold 5.

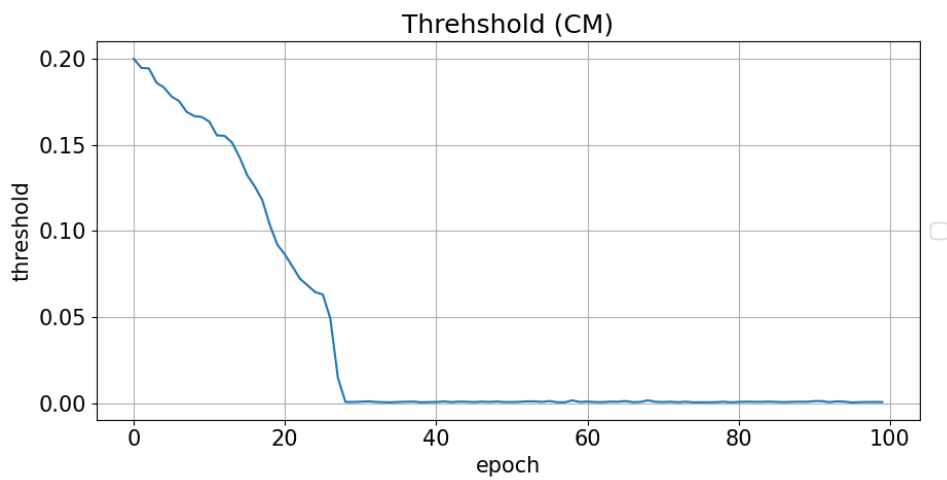


Figure 5.40: Threshold values learned by CM during training of fold 5.

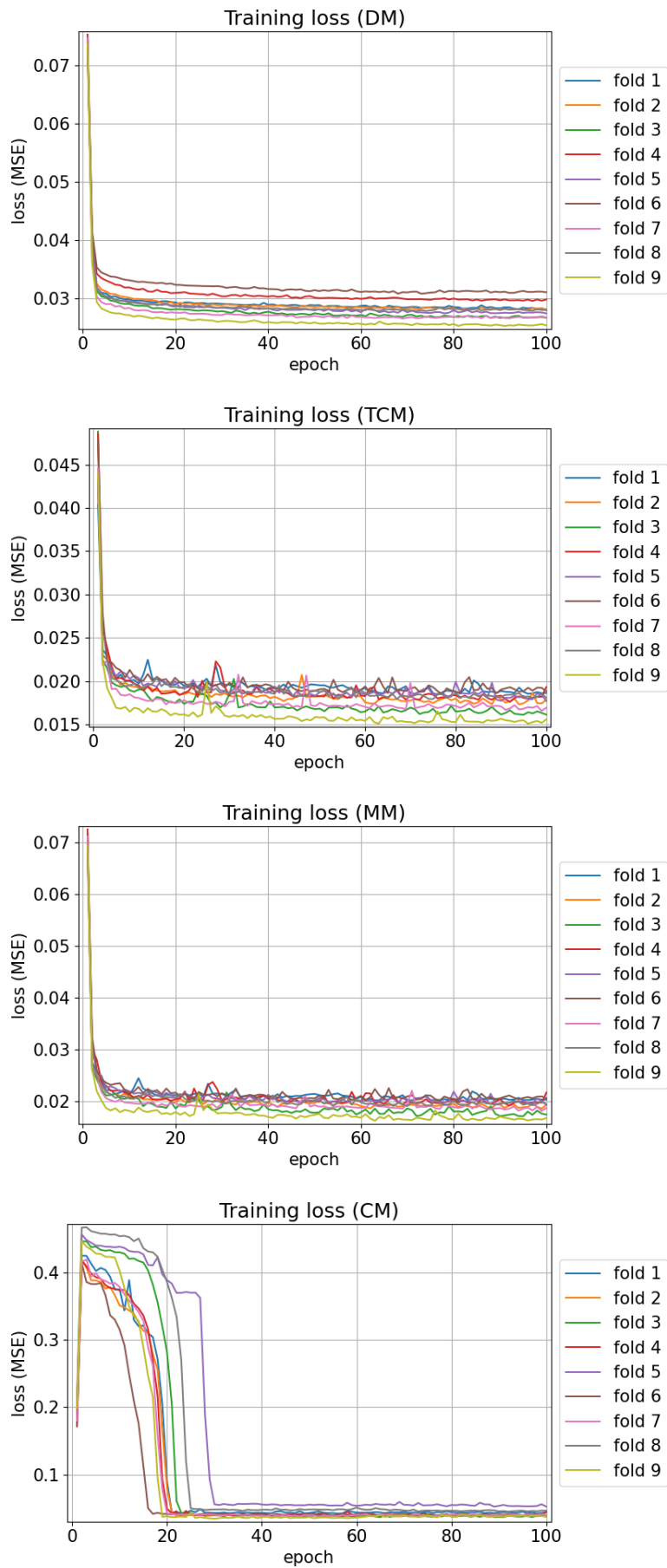


Figure 5.41: Losses obtained in all modules during training for each fold.

5.8 Results

In this section the results are presented and compared with previous works. The performance of the nine trained networks, one for each fold, was evaluated in their respective testing sets formed by videos of the VDAO-200 database.

Originally, the proposed network was built connecting the modules in sequence in the following order: DM, TCM, MM and CM. As the TCM aims to eliminate positive pixels which do not persist within a time window, we also considered having the TCM after the differentiable morphology module. Thus, another version of the network was trained considering the modules in the following order: DM, MM, TCM and CM. Results of these two versions of the network are also compared, namely *TCM MM*, the version where the TCM is applied before the MM, and *MM TCM*, the version of the network where the temporal consistency is applied after the MM.

Previous works evaluated their performances on the VDAO-200 database reporting their results with $\text{DIS}_{\text{overall}}$ (as presented in section 3.6) and the DIS averaged among all testing videos, considering two approaches: frame-level and object-level. For a better presentation of the results, both approaches are described in different subsections.

Frame-level Results

The frame-level approach evaluates the performance of a model to classify the target frame as anomalous or not anomalous. This is the same criterion that has been reported in previous chapters.

Table 5.4 presents the frame-level results with both versions of our network in comparison with other works. The DIS values in bold represent the results of our methods with equal or better performance than other approaches. By comparing our two approaches individually with the frame-level metric, the MM TCM version obtained a better average DIS than the TCM MM, being $\text{DIS}=0.31$ our best result. Both of our approaches performed better than the previous works.

By computing the $\text{DIS}_{\text{overall}}$ considering all 59 videos from Table 5.4 with Equation (3.6), TCM MM obtained $\text{TPR}=0.85$, $\text{FPR}=0.21$ and $\text{DIS}=0.26$, while MM TCM obtained $\text{TPR}=0.86$, $\text{FPR}=0.21$ and $\text{DIS}=0.25$. Therefore, our best $\text{DIS}_{\text{overall}}$ results were also reached with MM TCM.

Table 5.5 compares the results of our best model (MM TCM) to previous works considering the frame-level approach. Our MM TCM model surpasses previous works with respect to average DIS and $\text{DIS}_{\text{overall}}$ metrics.

Table 5.4: Comparative frame-level results of the proposed methods (TCM MM and MM TCM) with other works, considering all 59 videos of the VDAO-200 database. Values in bold highlight the DIS values obtained by our works that are equal or better in comparison to other works.

video	DAOMC [28]			ADAMULT [80]			MCBS [30]			mcDTSR [52]			TCM MM			MM TCM		
	TPR	FPR	DIS	TPR	FPR	DIS	TPR	FPR	DIS	TPR	FPR	DIS	TPR	FPR	DIS	TPR	FPR	DIS
1	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
2	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00	0.73	0.00	0.27	0.88	0.00	0.12	0.82	0.00	0.18
3	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
4	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
5	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
6	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
7	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
8	0.76	0.71	0.75	0.57	0.00	0.43	1.00	0.99	0.99	1.00	0.38	0.38	0.73	0.00	0.27	0.44	0.00	0.56
9	0.67	0.00	0.33	0.60	0.00	0.40	1.00	1.00	1.00	0.97	0.00	0.03	0.99	0.00	0.01	0.98	0.00	0.02
10	0.89	0.00	0.11	0.69	0.00	0.31	1.00	1.00	1.00	0.96	0.00	0.04	0.89	0.00	0.11	0.89	0.00	0.11
11	0.82	0.32	0.37	1.00	1.00	1.00	1.00	0.99	0.99	0.91	0.48	0.49	0.91	0.00	0.09	0.90	0.00	0.10
12	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.90	0.00	0.10	1.00	1.00	1.00	1.00	1.00	1.00
13	0.84	0.00	0.16	0.65	0.00	0.35	0.88	0.81	0.82	1.00	0.00	0.00	1.00	0.00	0.00	0.99	0.00	0.01
14	0.93	0.00	0.07	1.00	0.52	0.52	1.00	0.93	0.93	0.88	0.00	0.12	0.75	0.00	0.25	0.84	0.00	0.16
15	1.00	1.00	1.00	0.63	0.00	0.37	1.00	1.00	1.00	1.00	0.12	0.12	0.87	0.00	0.13	0.90	0.00	0.10
16	0.00	0.00	1.00	0.00	0.00	1.00	1.00	0.98	0.98	0.89	0.31	0.33	0.00	0.00	1.00	0.00	0.00	1.00
17	0.81	0.70	0.72	0.79	1.00	1.02	0.99	1.00	1.00	0.96	0.00	0.04	0.96	0.21	0.22	0.93	0.00	0.07
18	0.43	0.00	0.57	0.26	0.13	0.76	1.00	1.00	1.00	0.55	0.00	0.45	0.00	0.00	1.00	0.00	0.00	1.00
19	0.89	0.00	0.11	0.74	0.00	0.26	1.00	0.94	0.94	0.95	0.00	0.05	0.86	0.00	0.14	0.85	0.00	0.15
20	1.00	1.00	1.00	0.00	0.00	1.00	0.99	1.00	1.00	1.00	0.91	0.91	0.91	0.00	0.09	0.92	0.00	0.08
21	1.00	0.30	0.30	1.00	1.00	1.00	1.00	0.69	0.69	1.00	0.09	0.09	0.30	0.00	0.70	0.91	0.00	0.09
22	0.94	0.21	0.22	1.00	1.00	1.00	1.00	0.97	0.97	1.00	0.21	0.21	0.87	0.00	0.13	0.90	0.00	0.10
23	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.72	0.17	0.32	1.00	1.00	1.00	1.00	1.00	1.00
24	0.97	0.18	0.18	0.00	0.00	1.00	0.99	1.00	1.00	0.12	0.00	0.88	0.00	0.00	1.00	0.00	0.00	1.00
25	0.58	0.00	0.42	1.00	0.00	0.00	1.00	0.00	0.00	0.54	0.00	0.46	0.00	0.00	1.00	0.16	0.00	0.84
26	0.93	0.00	0.07	0.68	0.00	0.32	1.00	0.98	0.98	1.00	0.23	0.23	1.00	0.10	0.10	1.00	0.16	0.16
27	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.00	0.00	1.00	0.00	0.00
28	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00	0.26	0.00	0.74	1.00	0.00	0.00	1.00	0.00	0.00
29	0.76	0.00	0.24	0.69	0.00	0.31	1.00	1.00	1.00	0.00	0.00	1.00	0.69	0.00	0.31	0.70	0.00	0.30
30	0.80	0.00	0.20	0.56	0.00	0.44	1.00	0.98	0.98	1.00	0.16	0.16	0.95	0.00	0.05	0.93	0.00	0.07
31	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
32	0.83	0.00	0.17	0.32	0.00	0.68	1.00	1.00	1.00	0.99	0.06	0.06	0.98	0.00	0.02	0.94	0.00	0.06
33	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
34	0.70	0.00	0.30	0.56	0.00	0.44	1.00	0.98	0.98	0.97	0.09	0.10	0.81	0.00	0.19	0.76	0.00	0.24
35	1.00	0.20	0.20	0.63	0.00	0.37	1.00	0.93	0.93	0.96	0.00	0.04	0.93	0.00	0.07	0.93	0.00	0.07
36	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.95	0.27	0.27	1.00	1.00	1.00	1.00	1.00	1.00
37	0.93	0.00	0.07	0.93	0.00	0.07	1.00	1.00	1.00	0.97	0.00	0.03	0.88	0.00	0.12	0.84	0.00	0.16
38	0.76	0.13	0.28	0.47	0.00	0.53	1.00	0.97	0.97	0.76	0.03	0.25	0.96	0.00	0.04	0.95	0.00	0.05
39	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
40	1.00	1.00	1.00	1.00	0.59	0.59	1.00	1.00	1.00	1.00	1.00	1.00	0.94	0.00	0.06	0.94	0.00	0.06
41	1.00	0.95	0.95	1.00	1.00	1.00	1.00	0.97	0.97	1.00	0.09	0.09	1.00	0.40	0.40	1.00	0.28	0.28
42	1.00	1.00	1.00	0.50	0.00	0.50	1.00	0.97	0.97	0.99	0.00	0.01	0.94	0.00	0.06	0.92	0.00	0.08
43	0.14	0.00	0.86	0.00	0.00	1.00	1.00	1.00	1.00	0.93	0.26	0.27	0.00	0.00	1.00	0.00	0.00	1.00
44	0.78	0.38	0.44	0.63	0.00	0.37	1.00	1.00	1.00	0.95	0.00	0.05	0.86	0.00	0.14	0.85	0.00	0.15
45	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.37	1.00	1.18	1.00	1.00	1.00	1.00	1.00	1.00
46	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.90	0.00	0.10	0.90	0.00	0.10	0.90	0.00	0.10
47	0.93	0.00	0.07	0.91	0.00	0.09	1.00	1.00	1.00	0.97	0.00	0.03	0.80	0.00	0.20	0.82	0.00	0.18
48	0.72	0.00	0.28	0.42	0.00	0.58	1.00	1.00	1.00	0.98	0.00	0.02	0.94	0.00	0.06	0.94	0.00	0.06
49	1.00	0.20	0.20	0.93	0.00	0.07	1.00	1.00	1.00	1.00	1.00	1.00	0.95	0.00	0.05	0.97	0.00	0.03
50	0.97	0.00	0.03	1.00	1.00	1.00	1.00	1.00	1.00	0.97	0.00	0.03	0.95	0.00	0.05	0.96	0.00	0.04
51	0.96	0.86	0.86	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
52	0.84	0.82	0.83	1.00	1.00	1.00	1.00	1.00	1.00	0.74	0.00	0.26	1.00	1.00	1.00	1.00	1.00	1.00
53	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.99	0.41	0.41	1.00	0.22	0.22
54	0.85	0.00	0.15	0.50	0.00	0.50	1.00	1.00	1.00	1.00	0.00	0.00	0.93	0.00	0.07	0.93	0.00	0.07
55	0.79	0.67	0.70	0.50	0.00	0.50	1.00	1.00	1.00	0.71	0.00	0.29	0.62	0.00	0.38	0.84	0.00	0.16
56	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.64	0.64	1.00	0.91	0.91
57	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.96	0.17	0.18	0.98	0.24	0.24
58	0.62	0.00	0.38	0.19	0.00	0.81	1.00	1.00	1.00	0.88	0.00	0.12	0.90	0.00	0.10	0.90	0.00	0.10
59	1.00	0.49	0.49	0.54	0.00	0.46	1.00	1.00	1.00	0.62	0.01	0.38	0.91	0.00	0.09	0.91	0.00	0.09
Average	0.88	0.39	0.49	0.76	0.36	0.59	1.00	0.83	0.83	0.88	0.25	0.36	0.84	0.19	0.34	0.85	0.18	0.33

Object-level Results

The object-level approach (also referred to as pixel-level in some works) evaluates the classification of frames based on the intersection of the output image and the

Table 5.5: Average DIS and $\text{DIS}_{\text{overall}}$ frame-level results in the VDAO-200 of our best proposed method compared to other works. The best results of each metric are in bold.

	average			overall		
	TPR	FPR	average DIS	TPR	FPR	$\text{DIS}_{\text{overall}}$
DAOMC [28]	0.88	0.39	0.49	0.89	0.42	0.43
ADMULT [80]	0.76	0.36	0.59	0.78	0.39	0.44
MCBS [30]	1.00	0.83	0.83	1.00	0.98	0.98
mcDTSR [52]	0.88	0.25	0.36	0.88	0.28	0.30
MM TCM	0.85	0.18	0.33	0.86	0.21	0.25

ground-truth bounding box. In this approach, a true positive is accounted when the blob in the output image has a non-empty intersection with the ground-truth bounding box, and a false positive occurs when the blob and the ground-truth bounding box are disjoint.

The final output of our network provided by the CM is a classification of the target frame as anomalous or not anomalous. So, to evaluate our results with object-level approach, we disregard the classification provided by the CM, and perform the object-level evaluation based on the input image used by the CM.

Table 5.6 shows the DIS results of all testing videos of VDAO-200 considering the object-level approach. Our models obtained better average DIS in comparison to previous works. The average $\text{DIS}=0.35$ achieved by MM TCM is the best object-level result among all works.

Comparing both versions of our network using the $\text{DIS}_{\text{overall}}$ in the object-level approach, the TCM MM obtained an $\text{DIS}_{\text{overall}}=0.27$, while the MM TCM obtained a slightly better $\text{DIS}_{\text{overall}}=0.26$. Table 5.7 presents the average DIS and $\text{DIS}_{\text{overall}}$ computed with Equation (3.6) comparing our best network to previous works.

Figure 5.42 shows examples comparing frames output by our proposed MM TCM network in comparison with other works. Note that in Examples 1, 2 and 3 the shape of the blob produced by our method is closer to the object presented in the target frame. Example 3 of the second best method (MCBS) contains false positive on the left side of the frame, that was eliminated by our MM. In example 4 there is no anomaly present in the target frame, as correctly noticed by our approach. The darker part in the target frame, which was detected as a larger anomalous region by the MCBS, is in fact a shadow made by an object that is not in the scene. This is an example showing the robustness of our approach in some situations where shadows and differences of illumination are present.

All results including videos and frames output by our proposed network using the geometric alignment can be viewed at <https://bit.ly/3husYqb>.

Table 5.6: Comparative object-level results of the proposed methods (TCM MM and MM TCM) with other works, considering all 59 videos of the VDAO-200 database. Values in bold highlight the DIS values obtained by our works that are equal or better in comparison to other works.

video	DAOMC			ADAMULT			MCBS			mcDTSR			TCM MM			MM TCM		
	TPR	FPR	DIS	TPR	FPR	DIS	TPR	FPR	DIS	TPR	FPR	DIS	TPR	FPR	DIS	TPR	FPR	DIS
1	1.00	1.00	1.00	1.00	0.00	0.00	1.00	0.10	0.10	1.00	1.00	1.00	1.00	0.36	0.36	1.00	0.00	0.00
2	1.00	0.00	0.00	0.71	0.97	1.01	1.00	0.90	0.90	0.73	0.00	0.27	0.89	0.00	0.11	0.85	0.00	0.15
3	1.00	0.04	0.04	1.00	0.00	0.00	1.00	0.28	0.28	1.00	0.73	0.73	1.00	0.00	0.00	1.00	0.00	0.00
4	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
5	1.00	0.09	0.09	1.00	0.00	0.00	1.00	0.07	0.07	1.00	0.59	0.59	1.00	0.00	0.00	1.00	0.00	0.00
6	1.00	0.10	0.10	1.00	0.63	0.63	1.00	1.00	1.00	1.00	0.79	0.79	1.00	0.00	0.00	1.00	0.00	0.00
7	1.00	1.00	1.00	1.00	0.00	0.00	1.00	0.96	0.96	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
8	0.76	0.30	0.39	0.51	0.03	0.49	1.00	0.87	0.87	1.00	0.16	0.16	0.73	0.00	0.27	0.44	0.00	0.56
9	0.67	0.18	0.37	0.52	0.06	0.48	0.95	1.00	1.00	0.97	0.00	0.03	0.99	0.10	0.10	0.98	0.12	0.13
10	0.89	0.10	0.15	0.69	0.00	0.31	1.00	0.97	0.97	0.96	0.72	0.72	0.89	0.00	0.11	0.89	0.00	0.11
11	0.82	0.32	0.37	0.82	1.00	1.02	0.97	0.98	0.98	0.91	0.22	0.24	0.92	0.00	0.08	0.90	0.00	0.10
12	0.87	1.00	1.01	1.00	0.22	0.22	0.94	0.48	0.48	0.90	0.00	0.10	0.90	1.00	1.01	0.91	1.00	1.00
13	0.84	0.00	0.16	0.65	0.19	0.40	0.86	0.71	0.72	1.00	0.00	0.00	1.00	0.00	0.00	0.99	0.00	0.01
14	0.92	0.00	0.08	1.00	0.14	0.14	1.00	0.74	0.74	0.88	0.00	0.12	0.75	0.00	0.25	0.85	0.00	0.15
15	0.89	1.00	1.01	0.58	0.04	0.43	1.00	1.00	1.00	1.00	0.02	0.02	0.86	0.03	0.14	0.92	0.25	0.27
16	0.00	0.00	1.00	0.00	0.00	1.00	0.77	1.00	1.02	0.89	0.08	0.13	0.00	0.00	1.00	0.00	0.00	1.00
17	0.80	0.11	0.23	0.62	0.29	0.48	0.96	0.45	0.45	0.96	0.00	0.04	0.93	0.27	0.28	0.93	0.00	0.07
18	0.43	0.00	0.57	0.00	0.22	1.02	0.75	0.99	1.02	0.55	0.00	0.45	0.00	0.00	1.00	0.00	0.00	1.00
19	0.89	0.00	0.11	0.54	0.15	0.49	1.00	0.67	0.67	0.95	0.00	0.05	0.87	0.00	0.13	0.87	0.00	0.13
20	0.66	1.00	1.06	0.00	0.00	1.00	0.26	1.00	1.24	0.78	0.98	1.00	0.91	0.00	0.09	0.93	0.00	0.07
21	0.95	0.60	0.60	0.97	0.71	0.71	0.97	0.62	0.62	1.00	0.03	0.03	0.30	0.00	0.70	0.92	0.00	0.08
22	0.92	0.05	0.10	0.68	0.74	0.81	1.00	0.90	0.90	1.00	0.03	0.03	0.87	0.00	0.13	0.90	0.00	0.10
23	0.99	1.00	1.00	1.00	1.00	1.00	0.93	1.00	1.00	0.72	0.04	0.28	0.95	1.00	1.00	0.95	1.00	1.00
24	0.00	0.73	1.24	0.00	0.00	1.00	0.00	1.00	1.41	0.12	0.00	0.88	0.00	0.01	1.00	0.00	0.00	1.00
25	0.58	0.00	0.42	0.56	0.54	0.70	1.00	0.90	0.90	0.53	0.00	0.47	0.00	0.00	1.00	0.21	0.00	0.79
26	0.90	0.05	0.11	0.66	0.01	0.34	1.00	0.56	0.56	1.00	0.53	0.53	0.99	0.04	0.05	0.98	0.41	0.41
27	1.00	1.00	1.00	1.00	0.18	0.18	1.00	0.75	0.75	1.00	0.09	0.09	1.00	0.00	0.00	1.00	0.01	0.01
28	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.90	0.90	0.26	0.00	0.74	1.00	0.01	0.01	1.00	0.01	0.01
29	0.76	0.01	0.24	0.68	0.00	0.32	0.91	0.98	0.98	0.00	0.00	1.00	0.69	0.08	0.32	0.70	0.03	0.31
30	0.80	0.49	0.53	0.56	0.00	0.44	1.00	0.97	0.97	1.00	0.55	0.55	0.95	0.00	0.05	0.96	0.00	0.04
31	0.87	0.81	0.82	0.61	0.54	0.67	1.00	0.61	0.61	0.95	1.00	1.00	1.00	0.92	0.92	1.00	1.00	1.00
32	0.83	0.00	0.17	0.32	0.00	0.68	1.00	0.78	0.78	0.99	0.02	0.02	0.98	0.00	0.02	1.00	0.00	0.00
33	1.00	1.00	1.00	1.00	1.00	1.00	0.83	1.00	1.01	0.92	1.00	1.00	0.97	1.00	1.00	0.99	1.00	1.00
34	0.70	0.00	0.30	0.56	0.00	0.44	1.00	0.69	0.69	0.97	0.03	0.04	0.81	0.00	0.19	0.79	0.00	0.21
35	0.87	0.82	0.83	0.62	0.00	0.38	0.97	0.61	0.61	0.96	0.00	0.04	0.93	0.00	0.07	0.93	0.00	0.07
36	1.00	1.00	1.00	1.00	1.00	1.00	0.02	1.00	1.40	0.95	0.07	0.09	1.00	1.00	1.00	1.00	1.00	1.00
37	0.93	0.00	0.07	0.93	0.00	0.07	0.99	0.96	0.96	0.97	0.00	0.03	0.88	0.00	0.12	0.84	0.00	0.16
38	0.76	0.05	0.25	0.47	0.00	0.53	1.00	0.99	0.99	0.76	0.01	0.24	0.96	0.00	0.04	0.95	0.00	0.05
39	0.84	0.93	0.94	1.00	0.25	0.25	0.91	1.00	1.00	0.92	1.00	1.00	0.94	1.00	1.00	0.94	1.00	1.00
40	1.00	0.56	0.56	1.00	0.13	0.13	1.00	0.95	0.95	1.00	1.00	1.00	0.94	0.00	0.06	0.94	0.00	0.06
41	0.88	0.87	0.87	0.87	1.00	1.01	0.63	0.99	1.06	1.00	0.03	0.03	1.00	0.81	0.81	1.00	0.81	0.81
42	0.88	0.90	0.91	0.50	0.00	0.50	0.96	0.96	0.96	0.99	0.00	0.01	0.94	0.00	0.06	0.92	0.00	0.08
43	0.14	0.00	0.86	0.00	0.00	1.00	0.72	1.00	1.04	0.93	0.11	0.14	0.00	0.00	1.00	0.00	0.00	1.00
44	0.73	0.13	0.30	0.63	0.00	0.37	0.96	1.00	1.00	0.95	0.04	0.06	0.86	0.00	0.14	0.85	0.00	0.15
45	0.82	1.00	1.02	1.00	1.00	1.00	0.00	1.00	1.41	0.35	0.41	0.77	0.80	1.00	1.02	0.93	1.00	1.00
46	0.94	0.79	0.79	0.98	0.14	0.15	0.93	0.97	0.97	0.90	0.00	0.10	0.90	0.00	0.10	0.90	0.01	0.10
47	0.93	0.00	0.07	0.91	0.26	0.28	1.00	1.00	1.00	0.97	0.26	0.26	0.80	0.00	0.20	0.83	0.00	0.17
48	0.72	0.15	0.32	0.42	0.00	0.58	0.96	0.97	0.97	0.98	0.00	0.02	0.94	0.00	0.06	0.95	0.00	0.05
49	1.00	0.05	0.05	0.93	0.00	0.07	1.00	0.99	0.99	1.00	0.76	0.76	0.95	0.00	0.05	0.99	0.00	0.01
50	0.86	0.14	0.20	0.18	0.89	1.21	1.00	0.77	0.77	0.97	0.02	0.04	0.95	0.00	0.05	0.96	0.00	0.04
51	0.85	0.66	0.68	1.00	1.00	1.00	0.97	0.92	0.92	0.81	1.00	1.02	0.98	1.00	1.00	0.91	1.00	1.00
52	0.64	0.79	0.87	0.85	1.00	1.01	0.39	1.00	1.17	0.74	0.55	0.61	0.92	1.00	1.00	0.93	1.00	1.00
53	0.69	1.00	1.05	0.88	1.00	1.01	0.79	1.00	1.02	0.85	1.00	1.01	0.80	0.40	0.45	0.90	0.56	0.57
54	0.84	0.00	0.16	0.50	0.00	0.50	1.00	0.51	0.51	1.00	0.01	0.01	0.93	0.00	0.07	0.93	0.00	0.07
55	0.60	0.32	0.51	0.50	0.00	0.50	0.87	1.00	1.01	0.71	0.00	0.29	0.61	0.00	0.39	0.84	0.00	0.16
56	1.00	1.00	1.00	1.00	0.38	0.38	1.00	1.00	1.00	0.81	1.00	1.02	0.61	0.76	0.85	0.68	0.96	1.01
57	1.00	0.67	0.67	1.00	0.21	0.21	0.96	0.92	0.92	0.95	1.00	1.00	0.96	0.26	0.27	0.98	0.37	0.37
58	0.62	0.00	0.38	0.19	0.00	0.81	0.97	0.80	0.80	0.88	0.00	0.12	0.90	0.00	0.10	0.91	0.00	0.09
59	0.74	0.79	0.83	0.54	0.00	0.46	1.00	1.00	1.00	0.62	0.00	0.38	0.91	0.00	0.09	0.91	0.00	0.09
Average	0.81	0.42	0.53	0.70	0.29	0.54	0.88	0.83	0.86	0.86	0.29	0.39	0.82	0.20	0.36	0.84	0.21	0.35

Table 5.7: $\text{DIS}_{\text{overall}}$ object-level results in the VDAO-200 of our best proposed method compared with other works. The best results of each metric are in bold.

	average			overall		
	TPR	FPR	average DIS	TPR	FPR	overall
DAOMC [28]	0.81	0.42	0.53	0.82	0.42	0.45
ADMULT [80]	0.70	0.29	0.54	0.72	0.29	0.40
MCBS [30]	0.88	0.83	0.86	0.89	0.83	0.84
mcDTSR [52]	0.86	0.29	0.39	0.86	0.29	0.32
MM TCM	0.84	0.21	0.35	0.85	0.21	0.26



Figure 5.42: Comparative images output by layer TCM of our proposed method (MM TCM) and images produced by the second best approach.

5.9 Conclusions

In this chapter the anomaly detection problem was approached from another perspective. We start by proposing a geometrical alignment technique to improve the quality of the previous temporal alignment. The geometrical alignment estimates a transformation composed of rotation and translation based on the motion vectors between the reference and target frames. Also, differently from previous chapters, a network was proposed to distinguish the differences between pre-aligned pair of frames, providing the classification of the target frame as *anomalous* or *not anomalous*.

Our proposed network contains 1,028 trainable parameters distributed into four modules: dissimilarity module (DM), differentiable morphology module (MM), temporal consistency module (TCM), and classification module (CM). Each module was designed to perform specific operations, which when applied in sequence, aim to contribute to the correct classification.

The main contribution of this chapter is a novel differentiable morphology technique, which enables the automatic adjustment of the radius of a circular structuring element through the learning process. The differentiable morphology contains a single learnable parameter not causing an overhead in terms of computational cost.

Important conclusions can be pointed out as:

- Separating the training, validation and testing samples into nine folds prevents data contamination, but it makes the training process more time consuming.
- Although the proposed geometrical alignment is more computationally costly as it involves the computation of multiple transformations between the target frame and reference frames, the geometrical alignment eliminated the false positive pixels output by the DM.
- The attempt to use CLAHE in the reference and target frames to eliminate illumination differences did not improve the robustness of the geometrical alignment.
- Measuring the binary outputs produced by the DM, MM and TCM was only possible with the Matthews Correlation Coefficient (MCC). Due to the unbalanced amount of the two-class pixels (black and white) in the produced images, the MCC proved to be a good approach to measure their differences.
- The silhouette shapes of the anomalous objects were compared to rectangular ground-truth bounding boxes (as shown in Figure 5.13), causing a small deformation of the object shape output by the MM due to the closing operation. Silhouette ground-truth annotations [86] could improve the achieved results.

- As the DM performs only pixel-wise operations along the channel axis, it alone cannot differ spatially misaligned features of the target and reference feature tensors from features of the anomalous objects in the target frame such as borders and corners. Nevertheless, the geometrical alignment and subsequent modules help to eliminate artifacts produced by such misaligned regions.
- The DM proved to be robust to the illumination problem in most of the frames and specially in videos where the illumination varies a lot.
- Sigmoid functions using Equation (5.6) were successfully used to produce binary outputs. They were used in the last step of the dissimilarity module to produce binary images, and in the classification module to produce a binary classification. For both cases, the slant was adjusted using the γ value to provide the closest binary outputs, without compromising the gradients.
- The proposed differentiable morphology applying the opening and closing operations proved to be effective in approximating the traditional morphology operations, with the advantage of learning the radius of the structuring element.
- Placing the temporal consistency module after the differentiable morphology module led to better results than the ones obtained with it placed before the morphology module.
- The results obtained by our models MM TCM surpassed the previous works considering the average DIS and the DIS_{overall} metrics using the frame-level and object-level evaluation modes.

By the aforementioned points and the presented results, we conclude that the proposed network surpassed the state-of-the-art models in the anomaly detection task using the VDAO database.

The content found in the next chapter has no direct connection with the approaches seen in the current chapter to detect the anomalies represented by objects in the target videos. As next chapter covers metrics used to evaluate object detection tasks, which is a different approach than the ones used so far, the reader may read it independently.

Chapter 6

A Comparative Analysis of Object Detection Metrics with an Open-Source Tool

In the first attempts to detect the anomalies represented by objects in the VDAO database, techniques of object detection were explored but without any success. During our initial investigations, we noted a lack of consensus in different works and implementations concerning the evaluation metrics of the object detection problem, which motivated us to investigate this problem.

The evaluation of object detection is a topic that attracts much attention of the research community and industry. The need to gather and formalize the different metrics in a single work inspired us to develop an open-source tool to evaluate the object detection with different metrics, compatible with different file formats.

Seen that, this chapter aims to present our contributions done in the metrics applied to object detection tasks. By that, we start presenting the object detection problem, followed by an overview of selected works of the area. Then, the bounding box formats used by different datasets and output by object detectors are described. The following section explains the performance metrics used to evaluate object detectors and a numerical example illustrates the application of the metrics. A section is dedicated to depict the differences given by the most used object detection metrics. The developed tool is explained in the sequence and a practical example is offered showing results obtained in a large dataset. Finally, our conclusions are made in the last section of this chapter.

6.1 Introduction

The human visual system can effectively distinguish objects in different environments and contexts, even under a variety of constraints such as low illumination [158], color differences [159], and occlusions [11, 160]. In addition, objects are key to the understanding of a scene’s context, which lends paramount importance to the estimation of their precise location and classification. This has led computer vision researchers to explore automatic object detection for decades [161], reaching impressive results particularly in the last few years [25, 26, 162, 163].

Object detection algorithms attempt to locate general occurrences of one or more predefined classes of objects. In a system designed to detect pedestrians, for instance, an algorithm tries to locate all pedestrians that appear within an image or a video [160, 164, 165]. In the identification task, however, an algorithm tries to recognize a specific instance of a given class of objects. In the pedestrian example, an identification algorithm wants to determine the identity of each pedestrian previously detected.

Initially, real-time object detection applications were limited to only one object type [166] at a time, mostly due to hardware limitations. Later on, advancements in object detection techniques led to their increasing adoption in areas that included the manufacturing industry with optical inspections [22], video surveillance [167], forensics [23, 168], medical image analysis [18, 19, 169], autonomous vehicles [15], and traffic monitoring [17]. In the last decade, the use of deep neural networks (DNNs) has completely changed the landscape of the computer vision field [170]. DNNs have allowed for drastic improvements in image classification, image segmentation, anomaly detection, optical character recognition (OCR), action recognition, image generation, and object detection [161].

The field of object detection has yielded significant improvements in both efficiency and accuracy. To validate such improvements, new techniques must be assessed against current state-of-the-art approaches, preferably over widely available datasets. However, benchmark datasets and evaluation metrics differ from work to work, often making their comparative assessment confusing and misleading. We identified two main reasons for such confusion in comparative assessments:

- There are often differences in bounding box representation formats among different detectors. Boxes could be represented, for instance by their upper-left corner coordinates (x, y) and their absolute dimensions $(width, height)$ in pixels, or by their relative coordinates (x_{rel}, y_{rel}) and dimensions $(width_{rel}, height_{rel})$, with the values normalized by the image size, among others;

- Each performance assessment tool implements a set of different metrics, requiring specific formats for the ground-truth and detected bounding boxes.

Even though many tools have been developed to convert the annotated boxes from one format to another, the quality assessment of the final detections still lacks a tool compatible with different bounding box formats and multiple metrics. Our previous work [171] contributed to the research community in this direction, by presenting a tool which reads ground-truth and detected bounding boxes in a closed format and evaluates the detections using the average precision (AP) and mean average precision (mAP) metrics, as required in the PASCAL Challenge [172]. In this work that contribution is significantly expanded by incorporating 12 other metrics, as well as by supporting additional annotation formats into the developed open-source toolbox. The new evaluation tool is available at https://github.com/rafaelpadilla/review_object_detection_metrics. We believe that our work significantly simplifies the task of evaluating object detection algorithms.

This work intends to explain in detail the computation of the most popular metrics used as benchmarks by the research community, particularly in online challenges and competitions, providing their mathematical foundations and a practical example to illustrate their applicability. In order to do so, after a brief contextualization of the object-detection field in Section 6.2, the most common annotation formats and assessment metrics are examined in Sections 6.3 and 6.4, respectively. A numerical example is provided in Section 6.5 illustrating the previous concepts from a practical perspective. Popular metrics are further addressed in Section 6.6. Section 6.7 presents an open-source and freely distributed toolkit that implements all discussed concepts in a unified and validated way, as verified in Section 6.8. Finally, Section 6.9 concludes the paper by summarizing its main technical contributions.

6.2 An Overview of Selected Works on Object Detection

Back in the mid-50s and 60s the first attempts to recognize simple patterns in images were published [173, 174]. These works identified primitive shapes and convex polygons based on contours. In the mid-80s, more complex shapes started gaining meaning, such as in [175], which described an automated process to construct a three-dimensional geometric description of an airplane.

To describe more complex objects, instead of characterizing them by their shapes, automated feature extraction methods were developed. Different methods attempted to find important feature points that when combined could describe

objects broadly. Robust feature points are represented by distinctive pixels, whose neighborhood describe the same object irrespective of changes in pose, rotation, and illumination. The Harris detector [176] finds such points in the object corners based on local intensity changes. A local search algorithm using gradients was devised in [177] to solve the image registration problem, which later was expanded to a tracking algorithm [178] for identifying important points in videos.

More robust methods were able to identify characteristic pixel points and represent them as feature vectors. The so-called scale invariant feature transform (SIFT) [179], for instance, applied the difference of Gaussians in several scales coupled with histograms of gradients, yielding characteristic points with features that are robust to scale changes and rotation. Another popular feature detector and descriptor, the speed up robust features (SURF) [180], was claimed to be faster and more robust than SIFT, and uses a blob detector based on the Hessian matrix for interest point detection and wavelet responses for feature representations.

Feature-point representation methods alone are not able to perform object detection, but can help in extracting a group of keypoints that are used to represent them. In [181], the SIFT keypoints and features are used to detect humans in images, and in [182] SIFT was combined with color histograms to classify regions of interest across frames to track objects in videos. Another powerful feature extractor widely applied for object detection is the histogram of oriented gradients (HOG) [183], which is computed for several image small cells. The histograms of each cell are combined to form the object descriptor, which, associated to a classifier, can perform the object detection task [183, 184].

The Viola–Jones object detection framework was described in the path-breaking work of [166]. It could detect a single class object at a rate of 15 frames per second. The proposed algorithm employed a cascade of weak classifiers to process image patches of different sizes, being able to associate bounding boxes to the target object. The Viola–Jones method was first applied to face detection and required extensive training to automatically select a group of Haar-features to represent the target object, thus detecting one class of objects at a time. This framework has been extended to detect other object classes such as pedestrians [164, 165] and cars [16].

More recently, with the growth and popularization of deep learning in computer vision problems [4, 26, 162, 185, 186], object detection algorithms have started to develop from a new perspective [187, 188]. The traditional feature extraction [179, 180, 183] phase is performed by convolutional neural networks (CNNs), which are dominating computer vision research in many fields. Due to their spatial invariance, convolutions perform feature extraction spatially and can be combined into layers to produce the desired feature maps. The network end is usually composed of fully connected (FC) layers that can perform classification and regression tasks.

The output is then compared to a desired result and the network parameters are adjusted to minimize a given loss function. The advantage of using DNNs in object detection tasks is the fact that their architectures can extract features and predict bounding boxes in the same pipeline, allowing efficient end-to-end training. The more layers a network has, the more complex features it is able to extract, but the more parameters it needs to learn, demanding more computer processing power and data.

When it is not feasible to acquire more real data, data augmentation techniques are used to generate artificial but realistic data. Color and geometric operations and changes inside the target object area are the main actions performed by data augmentation methods for object detection tasks [189]. The work in [190] applied generative adversarial networks (GANs) to increase by 10 times the amount of medical chest images to detect patients with COVID-19. In [191], the number of images was increased by applying filters in astronomy images so as to improve the performance of galaxy detectors.

The CNN-based object detectors may be cataloged as single-shot or region-based detectors, also known as one- or two-stage detectors, respectively. The single-shot detectors work by splitting the images into a grid of cells. For each cell, they make bounding-box guesses of different scales and aspect ratios. This type of detector prioritizes speed rather than accuracy, aiming to predict both bounding box and class simultaneously. Overfeat [192] was one of the first single-shot detectors, followed by the single shot multiBox detector (SSD) [193], and all versions of you only look once (YOLO) [27, 40, 41, 163, 194]. The region-based detectors perform the detection in two steps. First, they generate a sparse set of region proposals in the image where the objects are supposed to be. The second stage classifies each object proposal and refines its estimated position. The region-based convolutional neural network (R-CNN) [38] was a pioneer employing CNNs in this last stage, achieving significant gains in accuracy. Later works such as Fast R-CNN [39], Faster R-CNN [25], and region-based fully convolutional networks (R-FCN) [195] suggest changes in R-CNN to improve its speed. The aforementioned detectors have some heuristic and hand-crafted steps such as region feature extraction or non-maximum suppression to remove duplicate detections. In this context, graph neural networks (GNNs) are employed to compute region of interest features in a more efficient way and process the objects simultaneously by modeling them according to their appearance feature and geometry [196, 197].

Hybrid solutions combining different approaches have been proposed lately and have proved to be more robust in various object-detection applications. The work in [198] proposes a hybrid solution involving a genetic algorithm and CNNs to classify small objects (structures) presented in microscopy images. Feature descriptors

coupled with a cuckoo search algorithm were applied by the authors of [20] to detect vessels in a marine environment using synthetic aperture radar (SAR) images. This approach was compared to genetic algorithms and neural network models individually, improving precision to nearly 96.2%. Vision-based autonomous vehicles can also benefit from hybrid models as shown in [199], where a system integrating different approaches was developed to detect and identify pedestrians and to predict their movements. In the context of detecting objects using depth information, the work in [200] proposes a hybrid attention neural network that incorporates depth and high-level RGB features to produce an attention map to remove background information.

Other works aim to detect the most important region of interest and segment relevant objects using salient object detectors. The work in [201] proposes a pipeline to separate an input image into a pair of images using content-preserving transforms. Then, each resulting image is passed by an interweaved convolutional neural network, which extracts complementary information of the image pairs and fuses them into the final salient map.

As medical images are acquired with special equipment, they form a very specific type of image [19]. To detect lesions, organs, and other structures of interest can be crucial for a precise diagnostic. However, most object detection systems are designed for general applications and usually do not perform well in medical images without adaptations [202]. Detecting anomalies such as glaucoma, breast, and lung lesions, for instance, have been explored from the medical object-detection perspective in [203, 204]. In the medical field, training and testing data usually have significant differences due to data scarcity and privacy. In order to address this issue, a domain adaptation framework, referred to as clustering CNNs (CLU-CNNs) [205], has been proposed to improve the generalization capability without specific domain training.

With new object detection methods being constantly released, it is highly desirable that a consensual evaluation procedure is established. To do so, the most common bounding box formats used by public datasets and competitions are revised in the next section.

6.3 Bounding Box Formats

Given the large, ever growing number of object detectors based on supervised methods in different areas, specific datasets have been built to train these systems. Popular competitions such as the common objects in context (COCO) [206], PASCAL visual object classes (VOC) [6], and Open Images Dataset [5] offer annotated datasets so that participants can train and evaluate their models before submission. Apart

from using available datasets, collecting data for object detection tasks can be quite challenging, as labeling images and videos is often a manual and quite demanding exercise. The medical field provides a good example of this fact: Developing a database of X-ray, electroencephalography (EEG), magnetoencephalography (MEG), or electrocardiography (ECG) images involves not only high costs for capturing such signals, but also requires expert knowledge to interpret and annotate them.

To ease the object annotation process in images and videos, many tools have been developed to annotate different datasets. Such tools basically offer the same features, such as bounding-box annotations and polygon-like silhouettes, as shown in Figure 6.1.

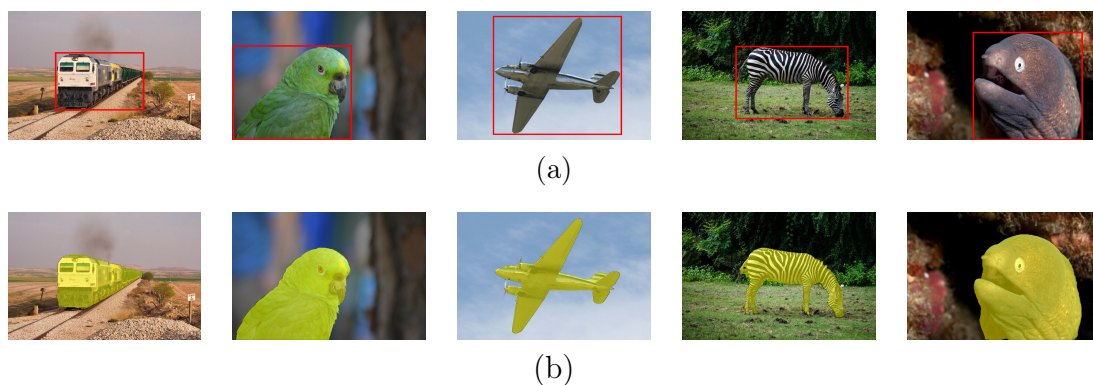


Figure 6.1: Two different types of annotated images from OpenImage [5]: (a) Bounding box annotations; (b) Silhouette annotations (in yellowish-green), also referred to as segmentation and pixel-level annotations.

A vast amount of annotation tools are freely available. Table 6.1 lists the most popular ones with their respective bounding box output formats.

Table 6.1: Popular free annotation tools and their supported output formats.

Annotation Tool	Annotation Types	Output Formats
LabelMe [207]	Bounding boxes and polygons	LabelMe, but provides conversion to COCO and PASCAL VOC
LabelIMG [208]	Bounding boxes	PASCAL VOC and YOLO
Microsoft VoTT [209]	Bounding boxes and polygons	PASCAL VOC, TFRecords, specific CSV, Azure Custom Vision Service, Microsoft Cognitive Toolkit (CNTK), VoTT
Computer Vision Annotation Tool (CVAT) [210]	Bounding boxes and polygons	COCO, CVAT, LabelMe, PASCAL VOC, TFRecord, YOLO, and others
VGG Image Annotation Tool (VIA) [211]	Bounding boxes and polygons	COCO and specific CSV and JSON

Some datasets introduced new formats to represent their annotations, which are usually named after the datasets themselves. The PASCAL VOC dataset [6] established the PASCAL VOC XML format and the COCO dataset [206] represents their annotations in the COCO format, embodied in a JSON file. Annotation tools

also brought further formats. For example, CVAT [210], a popular annotation tool, outputs bounding boxes in multiple formats, including its own specific XML-based one, named a CVAT format. The most popular bounding box formats shown in Table 6.1 are described in more detail. Note that whenever we refer to absolute coordinates, we mean coordinates that are expressed on the image coordinate frame, as opposed to coordinates that are normalized by the image width or image height.

1. **PASCAL VOC**: It consists of one XML file for each image containing none, one or multiple bounding boxes. The upper-left and bottom-right pixel coordinates are absolute. Each bounding box also contains a tag representing the class of the object. Extra information about the labeled object can be provided, such as whether, the object extends beyond the bounding box or it is partially occluded. The annotations in the ImageNet [82] and PASCAL VOC [6] datasets are provided using the PASCAL VOC format;
2. **COCO**: It is represented by a single JSON file containing all bounding boxes of a given dataset. The classes of the objects are listed separately in the *categories* tag and identified by an *id*. The image file corresponding to an annotation is also indicated in a separate element (*images*) that contains its file name and is referenced by an *id*. The bounding boxes and their object classes are listed in a different element (*annotations*), with their top-left (x, y) coordinates being absolute, and with explicit values of width and height;
3. **LabelMe**: The bounding-box annotations in this format are inserted in a single JSON file for each image, containing a list of boxes represented by their absolute upper-left and bottom-right coordinates. Besides the class of the object, this format also contains the image data encoded in base64 type, thus making the LabelMe format to consume more storage space than others;
4. **YOLO**: One TXT file per image is used in this representation. Each line of the file contains the class id and the bounding box coordinates. An extra file is needed to map the class id to the class name. The bounding box coordinates are not absolute, being represented by the format $(\frac{x_{center}}{image\ width}, \frac{y_{center}}{image\ height}, \frac{box\ width}{image\ width}, \frac{height}{image\ height})$. The advantage of representing the boxes in this format is that, if the image dimensions are scaled, the bounding box coordinates do not change, and thus the annotation file does not have to be altered. This type of format is the one preferred by those who annotate images in one resolution and need to scale their dimensions to fulfill the input shape requirement of a specific CNN. The YOLO object detector needs bounding boxes in this format to execute training;

5. **VoTT**: This representation of the bounding boxes coordinates and object class is made in a JSON file (one file per image) and the coordinates are expressed as the width, height and upper-left (x, y) pixel position in absolute coordinates. The Visual Object Tagging Tool (VoTT) produces annotations in this format;
6. **CVAT**: It consists of a unique XML file with all bounding boxes in the dataset represented by the upper-left and bottom-right pixel absolute coordinates. This format has been created with the CVAT annotation tool;
7. **TFRecord**: This is a serialized representation of the whole dataset containing all images and annotations in a single file. This format is recognized by the Tensorflow library [212];
8. **Tensorflow Object Detection**: This is a CSV file containing all labeled bounding boxes of the dataset. The bounding box format is represented by the upper-left and bottom-right pixel absolute coordinates. This is also a widely used format employed by the Tensorflow library;
9. **Open Images Dataset**: This format is associated with the Open Images Dataset [5] to annotate its ground-truth bounding boxes. All annotations are written in a unique CSV file listing the name of the images and labels, as well as upper-left and bottom-right absolute coordinates of the bounding boxes. Extra information about the labeled object is conveyed by other tags such as, for example, *IsOcclude*, *IsGroupOf*, and *IsTruncated*.

As each dataset is annotated using a specific format, works tend to employ the evaluation tools provided along with the datasets to assess their performance. Therefore, their results are dependent on the specific metric implementation associated with the used dataset. For example, the PASCAL VOC dataset employs the PASCAL VOC annotation format, which provides a MATLAB code implementing the metrics AP and mAP (intersection over union (IOU)=.50). This tends to inhibit the use of other metrics to report results obtained for this particular dataset. Table 6.2 lists popular object detection methods along with the datasets and the 14 different metrics used to report their results, namely: AP@[.5:.05:.95], AP@.50, AP@.75, AP_S, AP_M, AP_L, AR₁, AR₁₀, AR₁₀₀, AR_S, AR_M, AR_L, mAP (IOU=.50), and AP.

As the evaluation metrics are directly associated with a given annotation format, almost all works report their results only for the metrics implemented for the benchmarking dataset. For example, mAP (IOU=.50) is reported when the PASCAL VOC dataset is used, while AP@[.5:.05:.95] is applied to report results on the COCO dataset. If a work uses the COCO dataset to train a model and wants to

evaluate their results with the PASCAL VOC tool, it will be necessary to convert the ground-truth COCO JSON format to the PASCAL VOC XML format. This scenario discourages the use of such cross-dataset assessments, which have become quite rare in the object detection literature.

Table 6.2: Popular object detection methods along with the datasets and metrics used to report their results.

Method	Benchmark Dataset	Metrics
CornerNet [213]	COCO	AP@[.5:.05:.95]; AP@.50; AP@.75; AP _S ; AP _M ; AP _L
EfficientDet [214]	COCO	AP@[.5:.05:.95]; AP@.50; AP@.75
Fast R-CNN [39]	PASCAL VOC 2007, 2010, 2012	AP; mAP (IOU=.50)
Faster R-CNN [25]	PASCAL VOC 2007, 2012	AP; mAP (IOU=.50)
Faster R-CNN [25]	COCO	AP@[.5:.05:.95]; AP@.50
R-CNN [38]	PASCAL VOC 2007, 2010, 2012	AP; mAP (IOU=.50)
RFB Net [215]	PASCAL VOC 2007	mAP (IOU=.50)
RFB Net [215]	COCO	AP@[.5:.05:.95]; AP@.50; AP@.75; AP _S ; AP _M ; AP _L
RefineDet [216]	PASCAL VOC 2007, 2012	mAP (IOU=.50)
RefineDet [216]	COCO	AP@[.5:.05:.95]; AP@.50; AP@.75; AP _S ; AP _M ; AP _L
RetinaNet [217]	COCO	AP@[.5:.05:.95]; AP@.50; AP@.75; AP _S ; AP _M ; AP _L
R-FCN [195]	PASCAL VOC 2007, 2012	mAP (IOU=.50)
R-FCN [195]	COCO	AP@[.5:.05:.95]; AP@.50; AP _S ; AP _M ; AP _L
SSD [193]	PASCAL VOC 2007, 2012	mAP (IOU=.50)
SSD [193]	COCO	AP@[.5:.05:.95]; AP@.50; AP@.75; AP _S ; AP _M ; AP _L ; AR ₁ ; AR ₁₀ ; AR ₁₀₀ ; AR _S ; AR _M ; AR _L
SSD [193]	ImageNet	mAP (IOU=.50)
Yolo v1 [40]	PASCAL VOC 2007, 2012; Picasso; People-Art	AP; mAP (IOU=.50)
Yolo v2 [41]	PASCAL VOC 2007, 2012	AP; mAP (IOU=.50)
Yolo v2 [41]	COCO	AP@[.5:.05:.95]; AP@.50; AP@.75; AP _S ; AP _M ; AP _L ; AR ₁ ; AR ₁₀ ; AR ₁₀₀ ; AR _S ; AR _M ; AR _L
Yolo v3 [27]	COCO	AP@[.5:.05:.95]; AP@.50; AP@.75; AP _S ; AP _M ; AP _L ; AR ₁ ; AR ₁₀ ; AR ₁₀₀ ; AR _S ; AR _M ; AR _L
Yolo v4 [194]	COCO	AP@[.5:.05:.95]; AP@.50; AP@.75; AP _S ; AP _M ; AP _L
Yolo v5 [163]	COCO	AP@[.5:.05:.95]; AP@.50

An example of confusions that may arise in such a scenario is given by the fact that some works affirm that the metrics AP@.50 and mAP (IOU=.50) are the same [195], which may not always be true. The origins of such misunderstandings are the differences in how each tool computes the corresponding metrics. The next section deals with this problem by detailing the implementations of the several object detection metrics and pointing out their differences.

6.4 Performance Metrics

Challenges and online competitions have pushed forward the frontier of the object detection field, improving results for specific datasets in every new edition. To validate the submitted results, each competition applies a specific metric to rank the submitted detections. These assessment criteria have also been used by the research community to report and compare object detection methods using different datasets as illustrated in Table 6.2. Among the popular metrics to report the results, this section will cover those used by the most popular competitions, namely Open Images RVC [218], COCO Detection Challenge [219], VOC Challenge [172], Datalab Cup [220], Google AI Open Images challenge [221], Lyft 3D Object Detection for Autonomous Vehicles [222], and City Intelligence Hackathon [223]. Object detectors aim to predict the location of objects of a given class in an image or video with a high confidence. They do so by placing bounding boxes to identify the positions

of the objects. Therefore, a detection is represented by a set of three attributes: The object class, the corresponding bounding box, and the confidence score, usually given by a value between 0 and 1 showing how confident the detector is about that prediction. The assessment is done based on:

- A set of ground-truth bounding boxes representing the rectangular areas of an image containing objects of the class to be detected, and
- a set of detections predicted by a model, each one consisting of a bounding box, a class, and a confidence value.

Detection evaluation metrics are used to quantify the performance of detection algorithms in different areas and fields [224, 225]. In the case of object detection, the employed evaluation metrics measure how close the detected bounding boxes are to the ground-truth bounding boxes. This measurement is done independently for each object class, by assessing the amount of overlap of the predicted and ground-truth areas.

Consider a target object to be detected represented by a ground-truth bounding box B_{gt} and the detected area represented by a predicted bounding box B_p . Without taking into account a confidence level, a perfect match is considered when the area and location of the predicted and ground-truth boxes are the same. These two conditions are assessed by the intersection over union (IOU), a measurement based on the Jaccard Index, a coefficient of similarity for two sets of data [226]. In the object detection scope, the IOU is equal to the area of the overlap (intersection) between the predicted bounding box B_p and the ground-truth bounding box B_{gt} divided by the area of their union, that is:

$$J(B_p, B_{gt}) = \text{IOU} = \frac{\text{area}(B_p \cap B_{gt})}{\text{area}(B_p \cup B_{gt})}, \quad (6.1)$$

as illustrated in Figure 6.2.

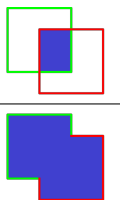
$$\text{IOU} = \frac{\text{area of overlap}}{\text{area of union}} = \frac{\text{area of overlap}}{\text{area of union}}$$


Figure 6.2: Illustration of the intersection over union (IOU).

A perfect match occurs when $\text{IOU} = 1$ and, if both bounding boxes do not intercept each other, $\text{IOU} = 0$. The closer to 1 the IOU gets, the better the detection is considered. As object detectors also perform the classification of each bounding

box, only ground-truth and detected boxes of the same class are comparable through the IOU.

By setting an IOU threshold, a metric can be more or less restrictive on considering detections as correct or incorrect. An IOU threshold closer to 1 is more restrictive as it requires almost-perfect detections, while an IOU threshold closer to, but different than 0 is more flexible, considering as detections even small overlaps between B_p and B_{gt} . IOU values are usually expressed in percentages, and the most used threshold values are 50% and 75%. In Sections 6.4.1, 6.4.2, and 6.4.4 the IOU is used to define the metrics that are most relevant to object detection.

6.4.1 Precision and Recall

Let us consider a detector that assumes that every possible rectangular region of the image contains a target object (this would be done by placing bounding boxes of all possible sizes centered in every image pixel). If there is one object to be detected, the detector would correctly find it by one of the many predicted bounding boxes. That is not an efficient way to detect objects, as many wrong predictions are made as well. Conversely, a detector which never generates any bounding box, will never have a miss-detection. These extreme examples highlight two important concepts, referred as precision and recall, are further explained below.

Precision is the ability of a model to identify only relevant objects. It is the percentage of correct positive predictions. Recall is the ability of a model to find all relevant cases (all ground-truth bounding boxes). It is the percentage of correct positive predictions among all given ground truths. To calculate the precision and recall values, each detected bounding box must first be classified as:

- **True positive (TP)**: A correct detection of a ground-truth bounding box;
- **False positive (FP)**: An incorrect detection of a non-existing object or a misplaced detection of an existing object;
- **False negative (FN)**: An undetected ground-truth bounding box.

Assuming there is a dataset with G ground-truths and a model that outputs N detections, of which S are correct ($S \leq G$), the concepts of precision and recall can

be formally expressed as:

$$Pr = \frac{\sum_{n=1}^S TP_n}{\sum_{n=1}^S TP_n + \sum_{n=1}^{N-S} FP_n} = \frac{\sum_{n=1}^S TP_n}{\text{all detections}}, \quad (6.2)$$

$$Rc = \frac{\sum_{n=1}^S TP_n}{\sum_{n=1}^S TP_n + \sum_{n=1}^{G-S} FN_n} = \frac{\sum_{n=1}^S TP_n}{\text{all ground truths}}. \quad (6.3)$$

6.4.2 Average Precision

As discussed above, the output of an object detector is characterized by a bounding box, a class, and a confidence interval. The confidence level can be taken into account in the precision and recall calculations by considering as positive detections only those whose confidence is larger than a τ . confidence threshold τ . The detections whose confidence level is smaller than τ are considered as negatives. By doing so, one may rewrite Equations (6.2) and (6.3) to consider this dependence on the confidence threshold τ as:

$$Pr(\tau) = \frac{\sum_{n=1}^S TP_n(\tau)}{\sum_{n=1}^S TP_n(\tau) + \sum_{n=1}^{N-S} FP_n(\tau)} = \frac{\sum_{n=1}^S TP_n(\tau)}{\text{all detections}(\tau)}, \quad (6.4)$$

$$Rc(\tau) = \frac{\sum_{n=1}^S TP_n(\tau)}{\sum_{n=1}^S TP_n(\tau) + \sum_{n=1}^{G-S} FN_n(\tau)} = \frac{\sum_{n=1}^S TP_n(\tau)}{\text{all ground truths}}. \quad (6.5)$$

Both $TP(\tau)$ and $FP(\tau)$ are decreasing functions of τ , as a larger τ reduces the number of positive detections. Conversely, $FN(\tau)$ is an increasing function of τ , since less positive detections imply a larger number of negative detections. In addition, $\sum TP(\tau) + \sum FN(\tau)$ does not depend on τ and is a constant equal to the number of all the ground truths. Therefore, from Equation (6.5), the recall $Rc(\tau)$ is a decreasing function of τ . On the other hand, nothing can be said a priori about the precision $Pr(\tau)$, since both the numerator and denominator of Equation (6.4) are decreasing functions of τ , and indeed the graph of $Pr(\tau) \times Rc(\tau)$ tends to exhibit a zig-zag behavior in practical cases, as later illustrated in Section 6.5.

In practice, a good object detector should find all ground-truth objects (FN = 0 \equiv high recall), while identifying only relevant objects (FP = 0 \equiv high precision). Therefore, a particular object detector can be considered good if, when the confidence threshold decreases, its precision remains high as its recall increases. Hence, a large area under the curve (AUC) tends to indicate both high precision and high recall. Unfortunately, in practical cases, the precision \times recall plot is often not monotonic, being zigzag-like instead, which poses challenges to an accurate measurement of its AUC.

The average precision (AP) is a metric based on the area under a $Pr \times Rc$ curve that has been pre-processed to eliminate the zig-zag behavior. It summarizes this precision-recall trade-off dictated by confidence levels of the predicted bounding boxes.

To compute the AP, one starts by ordering the K different confidence values $\tau(k)$ output by the object detector as:

$$\tau(k), \quad k = 1, 2, \dots, K \quad \text{such that} \quad \tau(i) > \tau(j) \quad \text{for} \quad i > j. \quad (6.6)$$

Since the Rc values also have a one-to-one, monotonic correspondence with τ , which has a one-to-one, monotonic, correspondence with the index k , then the $Pr \times Rc$ curve is not continuous but sampled at the discrete points $Rc(\tau(k))$, leading to the set of pairs $(Pr(\tau(k)), Rc(\tau(k)))$ indexed by k .

Now one defines an ordered set of reference recall values $R_r(n)$,

$$R_r(n), \quad n = 1, 2, \dots, N \quad \text{such that} \quad R_r(m) < R_r(n) \quad \text{for} \quad m > n. \quad (6.7)$$

The AP is computed using the two ordered sets in Equations (6.6) and (6.7). But before computing AP, the precision \times recall pairs have to be interpolated such that the resulting precision \times recall curve is monotonic. The resulting interpolated curve is defined by a continuous function $Pr_{\text{interp}}(R)$, where R is a real value contained in the interval $[0, 1]$, defined as:

$$Pr_{\text{interp}}(R) = \max_{k | Rc(\tau(k)) \geq R} \{Pr(\tau(k))\}, \quad (6.8)$$

where $\tau(k)$ is defined in Equation (6.6) and $Rc(\tau(k))$ is the recall value for the confidence $\tau(k)$, computed according to Equation (6.5). The precision value interpolated at recall R corresponds to the maximum precision $Pr_{\text{interp}}(k)$ whose corresponding recall value is greater than or equal to R . Note that an interpolation using a polynomial fitting would not be convenient in this case, since a polynomial interpolation cannot guarantee that the resulting interpolated curve is monotonic.

Now one is ready to compute AP by sampling $Pr_{\text{interp}}(R)$ at the N reference

recall values R_r defined in Equation (6.7). The AP is the area under the $Pr \times Rc$ curve calculated by a Riemann integral of $Pr_{\text{interp}}(R)$ using the K recall values from the set $R_r(k)$ in Equation (6.7) as sampling points, that is,

$$\text{AP} = \sum_{k=0}^K (R_r(k) - R_r(k+1)) Pr_{\text{interp}}(R_r(k)), \quad (6.9)$$

where $Pr_{\text{interp}}(R)$ is defined in Equation (6.8) and $R_r(k)$ is given by Equation (6.12), with $R_r(0) = 1$ and $R_r(K+1) = 0$.

There are basically two approaches to compute this Riemann integral: The N -point interpolation and the all-point interpolation, as detailed below.

N -point Interpolation

In the N -point interpolation, the set of reference recall values $R_r(n)$ for the computation of the Riemann integral in Equation (6.9) are equally spaced in the interval $[0, 1]$, that is,

$$R_r(n) = \frac{N-n}{N-1}, \quad n = 1, 2, \dots, N. \quad (6.10)$$

and thus the expression for AP becomes:

$$\text{AP} = \frac{1}{N} \sum_{n=1}^N Pr_{\text{interp}}(R_r(n)). \quad (6.11)$$

Actually the N -point interpolation as defined by Equation (6.11) computes an AP value which is equal to the value computed by the Riemann integral in Equation (6.9) multiplied by $\frac{N-1}{N}$.

Popular applications of this interpolation method use $N = 101$ as in the competition [219] and $N = 11$ as initially adopted by the competition [172], which was later changed to the all-point interpolation method.

All-Point Interpolation

For the computation of AP using the so-called all-point interpolation, here referred to as AP_{all} , as the set values $R_r(n)$ used to compute the Riemann integral in Equation (6.9) corresponds exactly to the set of recall values computed considering all K confidence levels $\tau(k)$ in Equation (6.6), with the confidences $\tau(0) = 0$ and $\tau(K+1) = 1$ added so that the points $R_r(0) = 1$ and $R_r(K+1) = 0$ are considered

in Equation (6.9). More precisely,

$$\begin{aligned} R_r(0) &= 1, \\ R_r(k) &= Rc(\tau(k)), \quad k = 1, 2, \dots, K, \\ R_r(K + 1) &= 0. \end{aligned} \tag{6.12}$$

where $Rc(\tau(k))$ is given by Equation (6.5) with $Rc(\tau(0)) = 1$ and $Rc(\tau(K + 1)) = 0$.

Using this definition of $R_r(k)$ in Equation (6.12), AP_{all} is computed using Equation (6.9). In the all-point interpolation, instead of using the precision observed at only a few points, the AP is obtained by interpolating the precision at each recall level. The Pascal Challenge [172] adopts the all-point interpolation method to compute the average precision.

6.4.3 Mean Average Precision

Regardless of the interpolation method, AP is obtained individually for each class. In large datasets with many classes, it is useful to have a unique metric that is able to represent the exactness of the detections among all classes. For such cases, the mean average precision (mAP) is computed, which is simply the average AP over all classes [25, 193], that is,

$$\text{mAP} = \frac{1}{C} \sum_{i=1}^C AP_i, \tag{6.13}$$

where AP_i is the AP value for the i -th class and C is the total number of classes being evaluated.

6.4.4 Average Recall

The average recall (AR) [227] is another evaluation metric used to measure the assertiveness of object detectors for a given class. Unlike the average precision, the confidences of the estimated detections are not taken into account in AR computation. This turns all detections into positive ones, which is equivalent to setting the confidence threshold as $\tau = 0$ in Equations (6.4) and (6.5).

The AR metric makes an evaluation at a large range of IOU thresholds, by taking into account all recall values obtained for IOU thresholds in the interval $[0.5, 1]$. An IOU of 0.5 can be interpreted as a rough localization of an object and is the least acceptable IOU by most of the metrics, and an IOU equal to 1 is equivalent to the perfect location of the detected object. Therefore, by averaging recall values in the interval $[0.5, 1]$, the model is evaluated on the condition of the object location being considerably accurate.

Let o be the IOU overlap between a ground truth and a detected bounding box as computed by Equation (6.1), and $Rc_{IOU}(o)$ a function that retrieves the recall for a given IOU o . The AR is defined as twice the area under the $Rc_{IOU}(o) \times o$ curve for the IOU interval $[0.5, 1]$, that is,

$$AR = 2 \int_{0.5}^1 Rc_{IOU}(o) do. \quad (6.14)$$

The authors in [227] also give a straightforward equation for the computation of the above integral from the discrete sample set, as twice the average of the excess IOU for all the ground-truths, that is,

$$AR = \frac{2}{G} \sum_{i=1}^G \max(\text{IOU}_i - 0.5, 0), \quad (6.15)$$

where IOU_i denotes the best IOU obtained for a given ground truth i and G is the total number of ground-truths.

Interestingly, COCO also reports the AR, although its definition does not match exactly that in Equation (6.15). Instead, what is reported as the COCO AR is the average of the maximum obtained recall across several IOU thresholds. To do so one first defines a set of O IOU thresholds:

$$t(o), \quad o = 1, 2, \dots, O. \quad (6.16)$$

Then, letting $Pr_{t(o)}(\tau(k))$, $Rc_{t(o)}(\tau(k))$ be the precision \times recall points for a confidence $\tau(k)$, given the IOU threshold $t(o)$, the COCO AR is computed as:

$$AR = \frac{1}{O} \sum_{o=1}^O \max_{k | Pr_{t(o)}(\tau(k)) > 0} \{Rc_{t(o)}(\tau(k))\}, \quad (6.17)$$

that is, the average of the largest recall values such that the precision is greater than zero for each IOU threshold, and $\tau(k)$ as defined in Equation (6.6). Effectively, this yields a coarse approximation of the original integral in Equation (6.14), provided that the IOU threshold set $t(o)$ covers an adequate range of overlaps.

6.4.5 Mean Average Recall

As the AR is calculated individually for each class, similarly to what is done to compute mAP, a unique AR value can be obtained considering the mean AR among all classes, that is:

$$\text{mAR} = \frac{1}{C} \sum_{i=1}^C AR_i. \quad (6.18)$$

In the sequel, a practical example illustrates the differences reflected in the final result depending on the chosen method.

6.5 A Numerical Example

Considering the set of 12 images in Figure 6.3, each image, except (a), (g), and (j), has at least one target object of the class *cat*, whose ground-truth locations are delimited by the green rectangles. There is a total of 12 target objects limited by the green boxes. Images (b), (e), and (f) have each two ground-truth samples of the target class. An object detector predicted 12 objects represented by the red rectangles (labeled with letters ‘A’ to ‘L’) with their associated confidence levels being represented as percentages also shown close to the corresponding boxes. From the above, images (a), (g), and (j) are expected to have no detection, and images (b), (e), and (f) are expected to have two detections each.

All things considered, to evaluate the precision and recall of the 12 detections it is necessary to establish an IOU threshold t , which will classify each detection as TP or FP. In this example, let us first consider as TP the detections with $\text{IOU} > 50\%$, that is $t = 0.5$.

As stated before, AP is a metric that integrates precision and recall in different confidence values. Thus, it is necessary to count the amount of TP and FP classifications given the different confidence levels. Table 6.3 presents each detection from our example sorted by their confidence levels. In this table, columns $\sum \text{TP}(\tau)$ and $\sum \text{FP}(\tau)$ are the accumulated TPs and FPs, respectively, whose corresponding confidence levels are larger than or equal to the confidence τ specified in the second column of the table. Precision ($Pr(\tau)$) and recall ($Rc(\tau)$) values are calculated based on Equations (6.4) and (6.5), respectively. In this example a detection is considered as a TP only if its IOU is larger than 50%, and in this case the column ‘IOU > 0.5?’ is marked as ‘Yes’, otherwise it is marked as ‘No’ and is considered an FP. In this example, all detections overlap some ground-truth with $\text{IOU} > 0.5$, except detection ‘J’, which is not overlapping any ground-truth, so there is no IOU to be computed in this case.

Some detectors can output one detection overlapping multiple ground truths, as seen in the image from Figure 6.3b with detections ‘A’ and ‘B’. As detection ‘A’ has a higher confidence than ‘B’ ($89\% > 82\%$), ‘A’ has the preference over ‘B’ to match the ground-truth, so ‘A’ is associated with the ground truth which gives the highest IOU. Figure 6.4c,d show the two possible associations that ‘A’ can have, ending up with the first one, which presents a higher IOU. Detection ‘B’ is left with the remaining ground truth in Figure 6.4f. Another similar situation where one detection could be associated with more than one ground truth is faced by detection

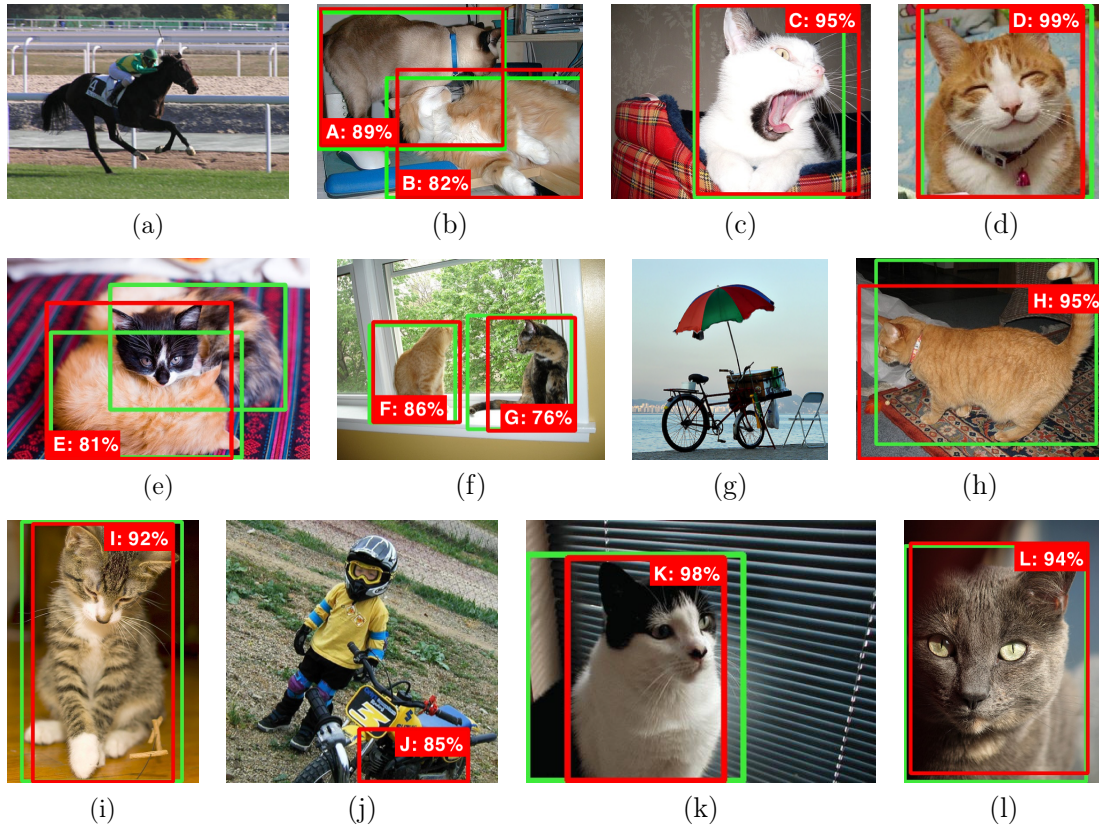


Figure 6.3: Samples of 12 images from the PASCAL VOC 2012 dataset [6] with ground-truth objects of the class *cat* in green boxes, and the detections performed by [163] in red boxes along with their respected confidence levels. In samples (b), (c), (d), (f), (h), (i), (k) and (l) the amount of the ground-truth and detected objects is the same. In samples of images (a), (g), and (j) no ground-truth object should be detected but one false detection occurred in image (j). In sample (e) there are two target objects to be detected, but the detector missed one of them.

‘E’ in Figure 6.3e. The application of the same rule results in matching detection ‘E’ with the ground truth whose IOU is the highest, represented by the fairer cat, at the bottom of the image.

By choosing a more restrictive IOU threshold, different precision $Pr(\tau)$ and recall $Rc(\tau)$ values can be obtained. Table 6.4 computes the precision and recall values with a more strict IOU threshold of $t = 0.75$. By that, it is noticeable the occurrence of more FP detections and less TP detections, thus reducing both the precision $Pr(\tau)$ and recall $Rc(\tau)$ values.

Graphical representations of the $Pr(\tau) \times Rc(\tau)$ values presented in Tables 6.3 and 6.4 can be seen in Figure 6.5. By comparing both curves, one may note that for this example:

- With a less restrictive IOU threshold ($t = 0.5$), higher recall values can be obtained with the highest precision. In other words, the detector can retrieve about 66.5% of the total ground truths without any miss detection.

Table 6.3: Precision and recall values for detections in Figure 6.3, that contain a total of 12 ground truths, considering an IOU threshold $t = 0.5$.

Bounding Box	Confidence(τ)	IOU	IOU > 0.5?	$\sum TP(\tau)$	$\sum FP(\tau)$	$Pr(\tau)$	$Rc(\tau)$
D	99%	0.91	Yes	1	0	1.0000	0.0833
K	98%	0.70	Yes	2	0	1.0000	0.1667
C	95%	0.86	Yes	3	0	1.0000	0.2500
H	95%	0.72	Yes	4	0	1.0000	0.3333
L	94%	0.91	Yes	5	0	1.0000	0.4167
I	92%	0.86	Yes	6	0	1.0000	0.5000
A	89%	0.92	Yes	7	0	1.0000	0.5833
F	86%	0.87	Yes	8	0	1.0000	0.6667
J	85%	-	No	8	1	0.8889	0.6667
B	82%	0.84	Yes	9	1	0.9000	0.7500
E	81%	0.74	Yes	10	1	0.9091	0.8333
G	76%	0.76	Yes	11	1	0.9167	0.9167

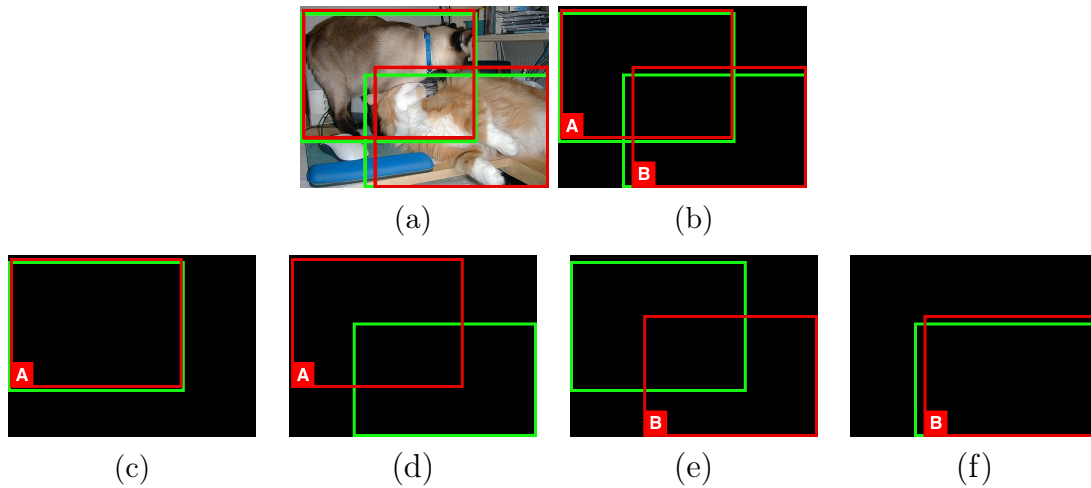


Figure 6.4: Particular cases showing detected bounding boxes overlapping multiple ground truths. (a) Original image with predicted (red) and ground-truth (green) bounding boxes. (b) Bounding boxes only. (c, d) Possible overlaps of the first ground truth. (c) Detection ‘A’ overlapping the first ground truth with IOU = 0.92. (d) Detection ‘A’ overlapping the second ground truth with IOU = 0.20. (e, f) Possible overlaps of the second ground truth. (e) Detection ‘B’ overlapping the first ground truth with IOU = 0.19. (f) Detection ‘B’ overlapping the second ground truth with IOU = 0.84.

- Using $t = 0.75$, the detector is more sensitive to different confidence values τ . This is explained by the more accentuated monotonic behavior for this IOU threshold.
- Regardless the IOU threshold applied, this detector can never retrieve 100% of the ground truths ($Pr(\tau) = 1$) for any confidence value τ . This is due to the fact that the algorithm failed to output any bounding box

Note that Figure 6.5 suggests that an IOU threshold of $t = 0.5$ is less affected by different confidence levels. The graph for the lowest IOU threshold ($t = 0.5$) shows

Table 6.4: Precision and recall values for detections in Figure 6.3, that contain a total of 12 ground truths, considering an IOU threshold $t = 0.75$.

Bounding Box	Confidence (τ)	IOU	IOU > 0.75?	$\sum\text{TP}(\tau)$	$\sum\text{FP}(\tau)$	$Pr(\tau)$	$Rc(\tau)$
D	99%	0.91	Yes	1	0	1.0000	0.0833
K	98%	0.70	No	1	1	0.5000	0.0833
C	95%	0.86	Yes	2	1	0.6667	0.1667
H	95%	0.72	No	2	2	0.5000	0.1667
L	94%	0.91	Yes	3	2	0.6000	0.2500
I	92%	0.86	Yes	4	2	0.6667	0.3333
A	89%	0.92	Yes	5	2	0.7143	0.4167
F	86%	0.87	Yes	6	2	0.7500	0.5000
J	85%	-	No	6	3	0.6667	0.5000
B	82%	0.84	Yes	7	3	0.7000	0.5833
E	81%	0.74	No	7	4	0.6364	0.5833
G	76%	0.76	Yes	8	4	0.6667	0.6667

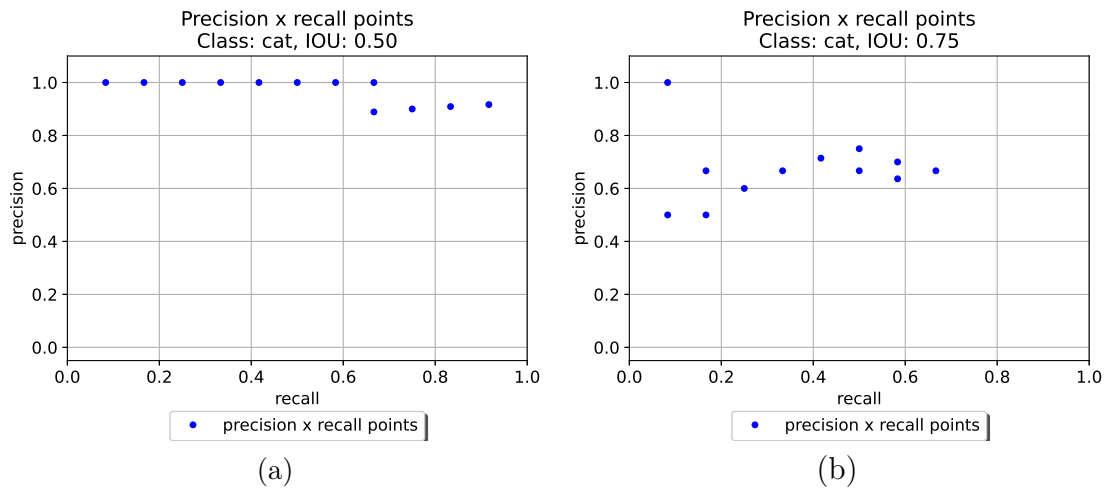


Figure 6.5: Precision \times Recall points with values calculated for: (a) Results provided in Table 6.3. (b) Results provided in Table 6.4.

that when confidence levels τ are high, the precision $Pr(\tau)$ does not vary, being equal to the maximum (1.0) for most of confidence values τ . However, in order to detect more objects (increasing the recall $Rc(\tau)$), it is necessary to set a lower confidence threshold τ , which reduces the precision at most by 12%. On the other hand, considering the highest IOU threshold ($t = 0.75$), the detector can retrieve half of the target objects (recall=0.5) with a precision of 0.75.

As previously explained, different methods can be applied to estimate the average precision, that is, the area under the precision \times recall curve. To obtain AP using the N -point interpolation in Equation (6.11) with $N = 11$ points, the area under the $Pr \times Rc$ curve is computed as the average of the interpolated precision $Pr_{\text{interp}}(R)$ (Equation (6.9)) samples considering the sampling recall points R at $R_r(n)$ in the set $\{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$ (Equation (6.10)). On the other hand, to obtain AP using the all-point interpolation approach, the area under the $Pr \times Rc$ curve is computed by the Riemann integral in Equation (6.9), sampling the

recall points R at $R_r(n)$ coincident with the $R_c(\tau)$ values given by the last column of Table 6.3 or of Table 6.4. The results can be seen in Figure 6.6. When an IOU threshold $t = 0.5$ was applied, the 11-point interpolation method obtained $AP = 88.64\%$ while the all-point interpolation method resulted in a slightly higher AP, reaching $AP = 89.58\%$. Similarly, for an IOU threshold $t = 0.75$, the 11-point interpolation method obtained $AP = 49.24\%$ and the all-point interpolation obtained $AP = 50.97\%$.

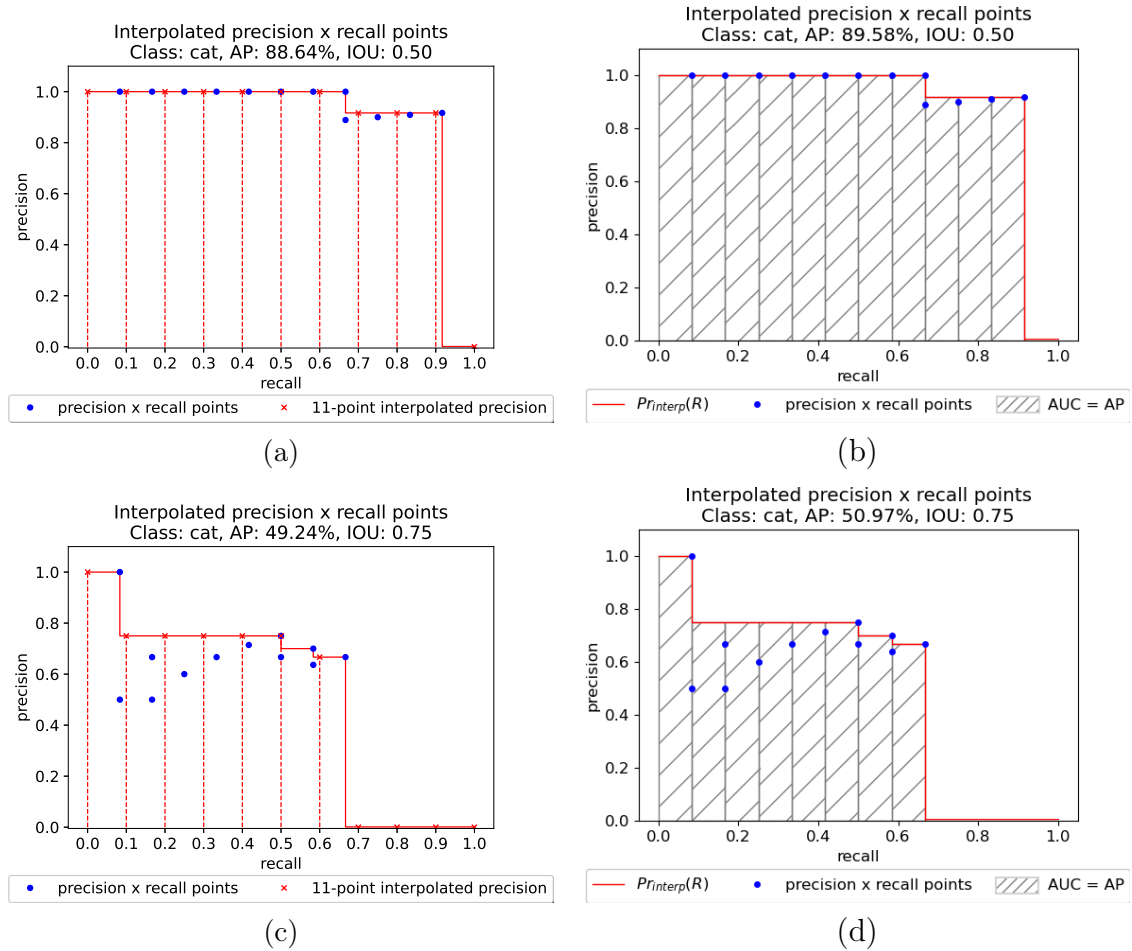


Figure 6.6: Results of different approaches to compute the AP metric. (a) 11-point interpolation with IOU threshold $t = 0.5$. (b) All-point interpolation with IOU threshold $t = 0.5$. (c) 11-point interpolation with IOU threshold $t = 0.75$. (d) All-point interpolation with IOU threshold $t = 0.75$.

When a lower IOU threshold t was considered ($t = 0.5$ as opposed to $t = 0.75$), the AP was considerably increased in both interpolation approaches. This is caused by the increase in the TP detections, due to a lower IOU threshold.

If focus is shifted towards how well localized the detections are, irrespective of their confidence values, it is sensible to consult the AR metrics (Equations (6.14)–(6.18)). Computing twice the average excess IOU for the samples in this practical example as in Equation (6.15), yields $AR = 60\%$, while computing the average max

recall across the standard COCO IOU thresholds, that is $t \in \{0.50, 0.55, \dots, 0.95\}$, as in Equation (6.17), yields $\text{AR} = 66\%$. As the latter computation effectively does a coarser quantization of the IOU space, the two AR figures differ slightly. The next section enlists and briefly describes which variations of the metrics based on AP and AR are more frequently employed in the literature. In most cases they are the result of combinations of different IOU thresholds and interpolation methods.

6.6 Most Employed Metrics Based on AP and AR

As previously presented, there are different ways to evaluate the area under the precision \times recall and recall \times IOU curves. Nonetheless, besides such combinations of different IOU thresholds and interpolation points, that are other variations that result in different metric values. Some methods limit the evaluation by object scales and detections per image. This section overviews the distinctions behind all the metrics shown in Table 6.2.

6.6.1 AP with IOU Threshold $t = 0.5$

This AP metric is widely used to evaluate detections in the PASCAL VOC dataset [6]. Its official implementation is in MATLAB and it is available in the PASCAL VOC toolkit. It measures the AP of each class individually by computing the area under the precision \times recall curve interpolating all points as presented in Equation (6.9). In order to classify detections as TP or FP the IOU threshold is set to $t = 0.5$.

6.6.2 mAP with IOU Threshold $t = 0.5$

This metric is also used by the PASCAL VOC dataset and is also available in their MATLAB toolkit. It is calculated as the AP with IOU $t = 0.5$, but the result obtained by each class is averaged as given in Equation (6.13).

6.6.3 AP@.5 and AP@.75

These two metrics evaluate the precision \times recall curve differently than the PASCAL VOC metrics. In this method, the interpolation is performed in $N = 101$ recall points, as given in Equation (6.11). Then, the computed results for each class are summed up and divided by the number of classes, as in Equation (6.13).

The only difference between AP@.5 and AP@.75 regards the applied IOU thresholds. AP@.5 uses $t = 0.5$ whereas AP@.75 applies $t = 0.75$. These metrics are com-

monly used to report detections performed in the COCO dataset and are officially available in their official evaluation tool.

6.6.4 AP@[.5:.05:.95]

This metric expands the AP@.5 and AP@.75 metrics by computing the AP@ with 10 different IOU thresholds ($t = [0.5, 0.55, \dots, 0.95]$) and taking the average among all computed results.

6.6.5 AP_S, AP_M, and AP_L

These three metrics, also referred to as AP Across Scales, apply the AP@[.5,.05:.95] from Section 6.6.1 taking into consideration the area of the ground-truth object:

- AP_S only evaluates small ground-truth objects (area < 32² pixels);
- AP_M only evaluates medium-sized ground-truth objects (32² < area < 96² pixels);
- AP_L only evaluates large ground-truth objects (area > 96²).

When evaluating objects of a given size, objects of the other sizes (both ground-truth and predicted) are not considered in the evaluation. This metric is also part of the COCO evaluation dataset.

6.6.6 AR₁, AR₁₀, and AR₁₀₀

These AR variations apply Equation (6.14) limiting the number of detections per image, that is, they calculate the AR given a fixed amount of detections per image, averaged over all classes and IOUs. The IOUs used to measure the recall values are the same as in AP@[.5,.05:.95].

AR₁ considers up to one detection per image, while AR₁₀ and AR₁₀₀ consider at most 10 and 100 objects per image, respectively.

6.6.7 AR_S, AR_M and AR_L

Similarly to the AR variations with limited number of detections per image, these metrics evaluate detections considering the same areas as the AP across scales. As the metrics based on AR are implemented in the COCO official evaluation tool, they are regularly reported with the COCO dataset.

6.6.8 F1-Score

The F1-score is defined as the harmonic mean of the precision (Pr) and recall (Rc) of a given detector, that is:

$$F_1 = 2 \frac{Pr \cdot Rc}{Pr + Rc} = \frac{TP}{TP + \frac{FN+FP}{2}}. \quad (6.19)$$

The F1-score is limited to the interval $[0, 1]$, being 0 if precision or recall (or both) are 0, and 1 when both precision and recall are 1.

As the F1-score does not take into account different confidence values, it is only used to compare object detectors in a fixed confidence threshold level τ .

6.6.9 Other Metrics

Other less popular metrics have also been proposed to evaluate object detections. They are mainly designed to be applied with particular datasets. The Open Images Object Detection Metric, for example, is similar to mAP (IOU=.50), being specifically designed to consider special ground-truth annotations of the Open Images dataset [5]. This dataset groups into a single annotation five or more objects of the same class that somehow are occluding each other, such as *a group of flowers* or *a group of people*. This metric simply ignores a detection if it overlaps a ground-truth box tagged as *group of*, whose area of intersection between the detection and ground-truth boxes divided by the area of the detection is greater than 0.5. This way, it does not penalize detections matching a group of very close ground-truth objects.

The localization recall-precision (LRP) error, a new metric suggested in [228], intends to consider the accuracy of the detected bounding box localization and equitably evaluate situations where the AP is unable to distinguish very different precision \times recall curves.

6.6.10 Comparisons among Metrics

In practice, the COCO's AP@[.5:.05:.95] and PASCAL mAP metrics are the most popular ones used as benchmarks. However, as COCO's AP@[.5:.05:.95] is affected by different IOUs, it is not possible to evaluate the effectiveness of the detector with a more or less restrictive IOU with this metric. For a more strict evaluation with respect to the likeness of the ground truth and detection bounding boxes, the AP@.75 metric should be applied. In datasets where the objects appear to have relatively different sizes, AP metrics concerning their areas should be employed. By that, the assertiveness of objects with similar relative sizes can be compared. As shown in

this work, the interpolation methods applied by the AP metrics try to remove the non-monotonic behavior of the $Pr(\tau) \times Rc(\tau)$ curve before calculating its AUC. In an N -point interpolation, a greater N leads to a better AUC approximation. Therefore, the 101-point interpolation approach used by COCO’s AP metrics provides a better AUC approximation than the 11-point interpolation approach. On the other hand, PASCAL VOC uses the all-point interpolation, which is an even better approximation of the AUC. In cases where the detector is expected to detect at least a certain amount of objects in a given image (e.g., detecting one bird in a flock of birds should be sufficient), AR metrics regarding detections or sizes are more appropriate.

6.7 An Open-Source Toolbox

This work focuses on explaining and comparing the different metrics and formats currently used in object detection, detailing the specifications and pointing out the particularities of each metric variation. The existing tools provided by popular competitions [172, 218–222] are not adequate to evaluate metrics using annotations in formats that are different from their native ones. Thus, to complement the analysis of the metrics presented here, the authors have developed and released an open-source toolkit as a reliable source of object detection metrics for the academic community and researchers.

With more than 3100 stars and 740 forks, our previously available tool for object detection assessment [229] has received positive feedback from the community and researchers. It has also been used as the official tool in competition [223], adopted in 3rd-party libraries such as [230], and parts of our code have been used by many other works such as in YoloV5 [163]. Besides the significant acceptance by the community, we have received many requests to expand the tool in order to support new metrics and bounding box formats. Such demands motivated us to offer more evaluation metrics, and to accept more bounding box formats.

This tool implements the same metrics used by the most popular competitions and object-detection benchmark researches. This implementation does not require modifications of the detection model to match complicated input formats, avoiding conversions to XML, JSON, CSV, or other file types. It supports more than eight different kinds of annotation formats, including the ones presented in Table 6.1. To ensure the accuracy of the results, the implementation strictly followed the metric definitions and the output results were carefully validated against the ones of the official implementations.

Developed in Python and supporting 14 object detection metrics for images, the tool can also be adapted and expanded to support new metrics and formats. The expanded project can be accessed at <https://github.com/rafaelpadilla/>

review_object_detection_metrics [231].

6.8 Metrics Evaluation in a Practical Example

In this section, we use different object detection metrics to evaluate YOLOv5 model [163]. The chosen model was trained with the COCO dataset and was applied in the training/validation PASCAL VOC 2012 dataset [6]. Intentionally different datasets were used to train and evaluate the model to evidence the potential of our tool to deal with different ground-truth and detected bounding-box formats. For this experiment, the annotations of the ground-truth boxes are in PASCAL VOC format containing 20 classes of objects, while the model was trained with COCO dataset and was able to detect objects in 80 classes, predicting detections in text files in the YOLO format.

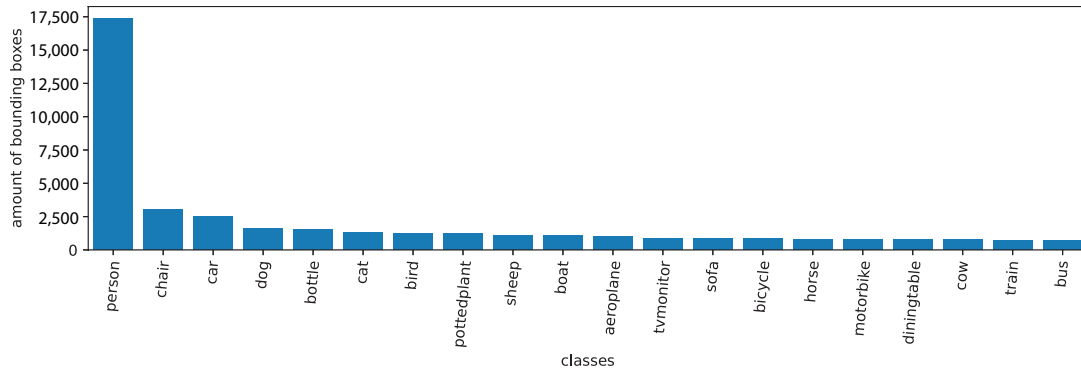
By using our tool, one can quickly obtain 14 different metrics without the necessity to convert files to specific formats. As some classes of the ground-truth dataset are tagged differently by the detector (e.g., PASCAL VOC class *tvmonitor* is referred to as *tv* in COCO dataset), the only required work is to provide a text file listing the names of the classes in the ground-truth format. This way the evaluation tool can recognize that the detected object *airplane* should be evaluated as *aeroplane*.

A total of 17,125 of images from the train/val PASCAL VOC 2012 dataset containing 40,138 objects of 20 classes were evaluated by the YOLOV5 model to detect objects in 80 different classes. A total of 74,752 detections were detected by the model. Figure 6.7 compares the distribution of ground-truth and detected objects per class. Due to the difference of classes in the training and testing datasets, many predicted classes are not in the ground-truth set, so detections of the extra classes are ignored by the metrics.

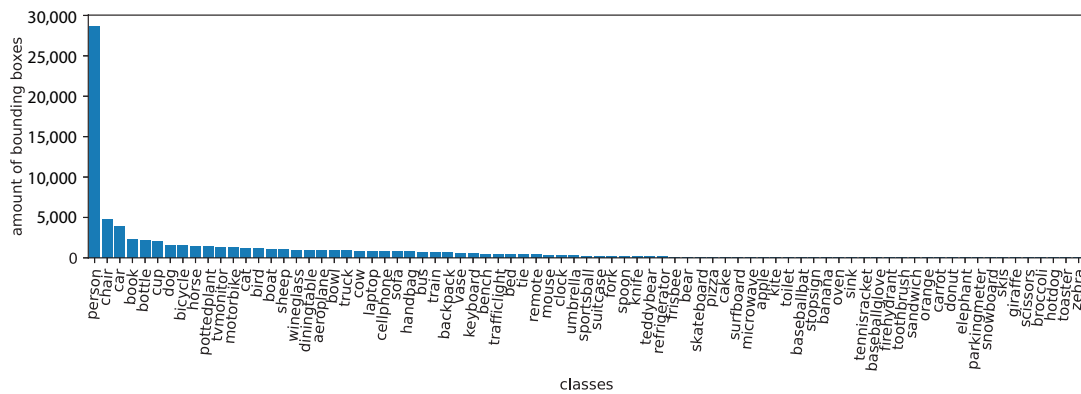
The AP results for each class are presented in Table 6.5. The highest AP values over all classes were obtained when the AUC was measured with the 11-point interpolation method and an IOU threshold of $t = 0.5$, resulting in $mAP = 0.58$. As expected for all cases, a more rigorous IOU threshold ($t = 0.75$) resulted in a smaller AP. Comparing the individual AP results among all classes, the most difficult object for all interpolation methods was the *potted plant*, having an AP not higher than 0.37 for an IOU threshold of $t = 0.5$ and an AP not higher than 0.22 with an IOU threshold of $t = 0.75$.

The results obtained by the variations which apply AP and AR with different sizes and quantity of objects per image are summarized in Table 6.6.

Even if the same interpolation technique is applied, the results may vary depending on the IOU threshold. Similarly, different interpolations with the same IOU threshold may also lead to distinct results.



(a) Class distribution in the ground-truth dataset.



(b) Class distribution of the detected objects.

Figure 6.7: Class distributions: (a) Ground-truth bounding boxes. (b) Detected bounding boxes.

The metrics considering objects in different scales are useful to compare the assertiveness of detections in datasets containing objects of different scales. In the COCO dataset, for instance, roughly 42% of the objects are considered small (area $< 32^2$ pixels), 34% are considered medium ($32^2 < \text{area} < 96^2$ pixels), and 24% are considered large (area $> 96^2$ pixels). This explains the vast amount of works using this dataset to report their results.

Table 6.5: AP results obtained with different interpolation methods and IOU thresholds.

Class	IOU threshold = 0.5			IOU threshold = 0.75		
	101-point	11-point	all-point	101-point	11-point	all-point
aeroplane	0.76	0.79	0.77	0.57	0.58	0.58
bicycle	0.41	0.43	0.41	0.31	0.33	0.31
bird	0.66	0.67	0.66	0.49	0.48	0.50
boat	0.47	0.46	0.47	0.28	0.29	0.29
bottle	0.45	0.47	0.45	0.32	0.34	0.33
bus	0.79	0.78	0.80	0.74	0.69	0.74
car	0.52	0.53	0.53	0.39	0.39	0.39
cat	0.73	0.74	0.73	0.54	0.53	0.54
chair	0.41	0.40	0.41	0.30	0.32	0.30
cow	0.74	0.69	0.74	0.59	0.58	0.60
diningtable	0.44	0.46	0.44	0.28	0.31	0.28
dog	0.66	0.64	0.65	0.53	0.53	0.53
horse	0.42	0.43	0.43	0.35	0.36	0.35
motorbike	0.51	0.53	0.51	0.38	0.39	0.38
person	0.67	0.65	0.68	0.53	0.54	0.53
pottedplant	0.37	0.39	0.37	0.22	0.25	0.22
sheep	0.68	0.68	0.68	0.56	0.58	0.57
sofa	0.44	0.45	0.44	0.37	0.38	0.37
train	0.75	0.77	0.76	0.65	0.66	0.65
tvmonitor	0.54	0.55	0.54	0.43	0.45	0.43
average	0.57	0.58	0.57	0.44	0.45	0.44

Table 6.6: Values of AP and average recall (AR) variations for different object sizes and number of detections per image.

Metric	Result
AP _S	0.13
AP _M	0.33
AP _L	0.46
AR ₁	0.39
AR ₁₀	0.53
AR ₁₀₀	0.53
AR _S	0.23
AR _M	0.47
AR _L	0.58

6.9 Conclusions

This work analyzed the formats of bounding boxes used to represent the objects in popular datasets and demonstrated the most common benchmark object detection metrics. The similarities and inconsistencies of each metric were examined, and our

results revealed their dissimilarities by evaluating the predictions of a pre-trained object detector in a largely used dataset.

A toolkit implementing all described metrics in a way compatible with the most data-annotation formats in use was presented and validated. Such results may facilitate direct and unified comparisons among most algorithms being proposed in the field of object detection. For future work, we intend to perform a regular survey update through its companion website, incorporating newly proposed metrics and annotation formats, and extend it to the problem of object tracking.

Chapter 7

Conclusions

In this thesis, we proposed a technique for real-time anomaly detection and frame classification in a cluttered environment using the VDAO database. Given two videos, being the reference video recorded in the expected scenario without anomalous objects and the target video, where anomalous objects may be presented, our goal was to develop a technique able to classify each frame as anomalous or non-anomalous, and identify regions of the anomalous frames containing such anomalous objects.

Supervised object detection techniques were first evaluated. As the variety of anomalous objects is limited in the VDAO database, and the nature of the problem does not allow us to know beforehand the class of the anomalous object in the scene, this approach was discarded. Nevertheless, our initial studies on object detection metrics allowed us to identify a gap in existing works, which allowed us to contribute with one conference [171] and one journal paper [232] on this topic.

As the perspective was changed to the anomaly detection point of view, we were motivated by the work [2], which used deep features extracted from both target and reference frames. We started developing a simple temporal alignment technique between both frames, which later was incremented with a more robust geometrical alignment.

Our initial attempts applied Bayesian optimization to find the best hyperparameters to train a Random Forest classifier using deep features extracted from the temporally aligned pair of frames. Our results did not show a significant improvement in the approach without hyperparameter selection. Still using the Random Forest classifier, we replaced the original feature reduction operation (average pooling) by selecting features based on their energy criterion using PCA. The computation burden was extremely high and the results were still not satisfactory. The incapacity of improving the previous results motivated us to explore the problem from another perspective.

A pipeline formed by well-defined operations allowed the processing of the fea-

ture tensors in sequential modules. The combination of these modules formed a network that produces an image containing blobs representing anomalous regions of the target frame, which is classified by the last module, the classification module (CM). The differentiable morphology module (MM) stands out from the other modules not only due to its contribution in eliminating false positive pixels while maintaining the true positive ones, but also due to its capacity to approximate morphological operations in a differentiable way. Results produced by our proposed network surpass those of previous works, showing the great potential of our model.

7.1 Future Work

In this section, we present the main ideas for future works that can be done in areas addressed in this thesis.

7.1.1 Input Features

The feature extraction reported in Chapter 3 uses the pretrained Resnet-50 backbone, which produces the features for all models explored in our work. It is known that shallower layers of a convolutional network produce low-level features such as borders, edges, and simple textures. As the 4th convolutional layer, the residual 3, obtained the best results, only features extracted by this layer were used in the network MM TCM proposed in Chapter 5. Nevertheless, the DM presented in Section 5.3 could be used to combine feature maps from different layers in an attempt to improve the classification results. Its structure could be replicated multiple times, so that other feature maps are weighed with more sets of values, and possibly the anomalous objects can be more distinguishable in the output image with fewer false positive regions.

7.1.2 Retrain the Network

As reported in Chapter 5, the modules of the proposed network were trained individually. The reason for that is due the lower gradient values computed in the dissimilarity module, resulting in the vanishing gradient problem, hindering the learning process of the parameters of the previous modules.

The work [156] affirms that the correct initialization of the network parameters may prevent the network to get stuck in poor solutions. As a possible solution for the vanishing gradient problem observed when the MM and CM modules are trained together, may be an alternative to use the already trained parameters as initial values. This solution would use only the classification loss to update the radii of morphology operations and the classification threshold.

7.1.3 Apply Differentiable Morphology in Other Problems

The attempt to create differentiable morphology operations has been explored by different works [151–153]. However, the simplicity and efficiency of our proposed differentiable morphology could contribute to other tasks, such as noise suppression, background removal, image segmentation, etc.

The radius of the circular structuring element of our proposed differentiable morphological operations is the only learnable parameter, being able to produce results similar to the traditional non-differentiable operations.

7.1.4 Replace the Geometric Alignment by the Temporal Alignment

The results in Chapter 5 were obtained with the geometrical alignment. Although this technique shows better results than previous works, the geometrical alignment is time consuming and cannot be applied in real time.

As the temporal alignment is less time consuming and can be applied in real time, a further experiment is required to evaluate the performance of the network with the temporal alignment.

Appendix A

Published and Submitted Papers

During the elaboration of this thesis, some works were published as listed further below. We highlight the works **J1.** and **C1.** for achieving an expressive number of citations in a relative short period of time. According to google scholar platform, by the conclusion of this thesis, the work **C1.** has 121 citations and its associated github repository has 3,741 stars, and the work **J1.** has 26 citations and its associated github repository has 350 stars.

A.1 Journal Papers

- J1.** Padilla, R., Passos, W. L., Dias, Thadeu L. B., Netto, S. L., da Silva, E. A. B., A Comparative Analysis of Object Detection Metrics with a Companion Open-Source Toolkit, *Electronics, Special Issue on Deep Learning Based Object Detection*, vol. 10, January, 2021.
- J2.** Cinelli, L. P., de Oliveira, J. F., L., de Pinho, V., Passos, W. L., Padilla, R., Braz, P., Galves, B., Dalvi, D., Lewenfus, G., Ferreira, J., Ji, A., de Oliveira, F., Gonçalves, C., Netto, S. L., da Silva, E. A., B., de Campos, M. L. R., Automatic Event Identification and Extraction from Daily Drilling Reports Using an Expert System and Artificial Intelligence, vol. 205, *Journal of Petroleum Science and Engineering*, October, 2021.

A.2 Conference Papers

- C1.** R. Padilla, S. L. Netto and E. A. B. da Silva, A Survey on Performance Metrics for Object-Detection Algorithms, *International Conference on Systems, Signals and Image Processing*, Niterói, Brazil, July, 2020.
- C2.** Dias, T. L. B., Tavares, L. G., Padilla, R., da Silva, A. F., Thomaz, L. A.,

Netto, S. L., da Silva, E. A. B., Siamese Networks for Bounding-Box to Silhouette Annotation of Video Databases, *XXXVIII Simpósio Brasileiro de Telecomunicações e Processamento de Sinais*, Brazil, November, 2020.

- C3.** Barros, B., Passos, W. L., Padilla, R., da Silva, Netto, S. L., E. A. B., Araújo, G. M., Acerca de Técnicas de Aumento de Dados para a Detecção Automática de Focos de Mosquito, *XXXVII Simpósio Brasileiro de Telecomunicações e Processamento de Sinais*, Petrópolis, Brazil, September, 2019.

References

- [1] “VDAO: Video Database of Abandoned Objects in a Cluttered Industrial Environment”. 2014. Disponível em: <http://www.smt.ufrj.br/~tvdigital/database/objects>. Accessed: 2021-06-28.
- [2] AFONSO, B. M., CINELLI, L. P., THOMAZ, L. A., et al. “Moving-Camera Video Surveillance in Cluttered Environments Using Deep Features”. In: *Proceedings of the IEEE International Conference on Image Processing*, pp. 2296–2300, 2018.
- [3] BERNDT, D. J., CLIFFORD, J. “Using Dynamic Time Warping to Find Patterns in Time Series”. In: *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pp. 359–370, 1994.
- [4] HE, K., ZHANG, X., REN, S., et al. “Deep Residual Learning for Image Recognition”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3350–3355, Las Vegas, NV, USA, 27–30 June 2016. IEEE.
- [5] KUZNETSOVA, A., ROM, H., ALLDRIN, N., et al. “The Open Images Dataset V4: Unified Image Classification, Object Detection, and Visual Relationship Detection at Scale”, *International Journal of Computer Vision*, v. 128, pp. 1956–1981, 2020.
- [6] EVERINGHAM, M., VAN GOOL, L., WILLIAMS, C. K. I., et al. “The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results”. <http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html>, 2011.
- [7] LAMA, R. K., GWAK, J., PARK, J.-S., et al. “Diagnosis of Alzheimer’s Disease Based on Structural MRI Images Using a Regularized Extreme Learning Machine and PCA Features”, *Journal of healthcare engineering*, v. 2017, 2017.
- [8] SIROVICH, L., KIRBY, M. “Low-dimensional Procedure for the Characterization of Human Faces”, *Josa a*, v. 4, n. 3, pp. 519–524, 1987.

- [9] CARDENAS, J., SOTO, J. E., FIGUEROA, M. “Multimodal Registration of Visible, SWIR and LWIR Images in a Distributed Smart Camera System”, *Microprocessors and Microsystems*, v. 73, pp. 102987, 2020.
- [10] SLANEY, M., COVELL, M., LASSITER, B. “Automatic Audio Morphing”. In: *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, v. 2, pp. 1001–1004, 1996.
- [11] GAO, T., PACKER, B., KOLLER, D. “A Segmentation-Aware Object Detection Model with Occlusion Handling”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1361–1368, Colorado Springs, CO, USA, 20–25 June 2011. IEEE.
- [12] SHEEN, D. M., MCKIM, D. L., HALL, T. E. “Three-Dimensional Millimeter-Wave Imaging for Concealed Weapon Detection”, *IEEE Transactions on Microwave Theory and Techniques*, v. 49, n. 9, pp. 1581–1592, 2001.
- [13] OTOOM, A. F., GUNES, H., PICCARDI, M. “Feature Extraction Techniques For Abandoned Object Classification in Video Surveillance”. In: *IEEE International Conference on Image Processing*, pp. 1368–1371, 2008.
- [14] SHIN, H.-C., LEE, J.-Y. “Pedestrian Video Data Abstraction and Classification for Surveillance System”. In: *Proceedings of the IEEE International Conference on Information and Communication Technology Convergence*, pp. 1476–1478, 2018.
- [15] JANAI, J., GÜNEY, F., BEHL, A., et al. *Computer Vision for Autonomous Vehicles: Problems, Datasets and State of the Art*. Now Publishers, 2020.
- [16] SUN, Z., BEBIS, G., MILLER, R. “On-Road Vehicle Detection: A Review”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 28, n. 5, pp. 694–711, 2006.
- [17] BUCH, N., VELASTIN, S. A., ORWELL, J. “A Review of Computer Vision Techniques for the Analysis of Urban Traffic”, *IEEE Transactions on Intelligent Transportation Systems*, v. 12, n. 3, pp. 920–939, 2011.
- [18] GANSTER, H., PINZ, P., ROHRER, R., et al. “Automated Melanoma Recognition”, *IEEE Transactions on Medical Imaging*, v. 20, n. 3, pp. 233–239, 2001.

- [19] OLABARRIAGA, S. D., SMEULDERS, A. W. M. “Interaction in the Segmentation of Medical Images: A Survey”, *Medical Image Analysis*, v. 5, n. 2, pp. 127–142, 2001.
- [20] S, I. T. J., SASIKALA, J., JULIET, D. S. “Optimized Vessel Detection in Marine Environment Using Hybrid Adaptive Cuckoo Search Algorithm”, *Computers & Electrical Engineering*, v. 78, pp. 482–492, 2019.
- [21] LEPETIT, V., LAGGER, P., FUA, P. “Randomized Trees for Real-Time Keypoint Recognition”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, v. 2, pp. 775–781, 2005.
- [22] HIRANO, Y., GARCIA, C., SUKTHANKAR, R., et al. “Industry and Object Recognition: Applications, Applied Research and Challenges”, *Toward Category-Level Object Recognition*, v. 4170, pp. 49–64, 2006.
- [23] FRANKE, K., SRIHARI, S. N. “Computational Forensics: An Overview”. In: *International Workshop on Computational Forensics (IWCF)*, pp. 1–10, Washington, DC, USA, 7–8 August 2008. Springer.
- [24] VISHWAKARMA, S., AGRAWAL, A. “A Survey on Activity Recognition and Behavior Understanding in Video Surveillance”, *The Visual Computer*, v. 29, n. 10, pp. 983–1009, 2013.
- [25] REN, S., HE, K., GIRSHICK, R., et al. “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 39, n. 6, pp. 1137–1149, 2017.
- [26] KRIZHEVSKY, A., SUTSKEVER, I., HINTON, G. E. “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Proceedings of the International Conference on Neural Information Processing Systems (NeurIPS)*, pp. 1097–1105, Lake Tahoe, Nevada, USA, 2012. Curran Associates.
- [27] REDMON, J., FARHADI, A. “YOLOv3: An Incremental Improvement”, *Technical Report*, 2018.
- [28] KONG, H., AUDIBERT, J.-Y., PONCE, J. “Detecting Abandoned Objects with a Moving Camera”, *IEEE Transactions on Image Processing*, v. 19, n. 8, pp. 2201–2210, 2010.

- [29] ADAM, A., RIVLIN, E., SHIMSHONI, I., et al. “Robust Real-Time Unusual Event Detection Using Multiple Fixed-Location Monitors”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 30, n. 3, pp. 555–560, 2008.
- [30] MUKOJIMA, H., DEGUCHI, D., KAWANISHI, Y., et al. “Moving Camera Background-Subtraction for Obstacle Detection on Railway Tracks”. In: *Proceedings of the IEEE International Conference on Image Processing*, pp. 3967–3971, 2016.
- [31] THOMAZ, L. A., DA SILVA, A. F., DA SILVA, E. A. B., et al. “Detection of Abandoned Objects Using Robust Subspace Recovery with Intrinsic Video Alignment”. In: *IEEE International Symposium on Circuits and Systems*, pp. 1–4, 2017.
- [32] HU, W.-C., CHEN, C.-H., CHEN, T.-Y., et al. “Moving Object Detection and Tracking from Video Captured by Moving Camera”, *Journal of Visual Communication and Image Representation*, v. 30, pp. 164–180, 2015.
- [33] DA SILVA, A. F., THOMAZ, L. A., CARVALHO, G., et al. “An Annotated Video Database for Abandoned-Object Detection in a Cluttered Environment”. In: *Proceedings of the International Telecommunications Symposium*, pp. 1–5, 2014.
- [34] SKEAT, W. W. *An Etymological Dictionary of the English Language*. Clarendon Press, 1968.
- [35] CHANDOLA, V., BANERJEE, A., KUMAR, V. “Anomaly Detection: A Survey”, *Journal of ACM Computing Surveys*, pp. 1–58, 2009.
- [36] RÄTY, T. D. “Survey on Contemporary Remote Surveillance Systems for Public Safety”, *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, v. 40, n. 5, pp. 493–515, 2010.
- [37] TIAN, Y., FERIS, R. S., LIU, H., et al. “Robust Detection of Abandoned and Removed Objects in Complex Surveillance Videos”, *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, v. 41, n. 5, pp. 565–576, 2010.
- [38] GIRSHICK, R., DONAHUE, J., DARRELL, T., et al. “Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 580–587, Columbus, OH, USA, 2014. IEEE.

- [39] GIRSHICK, R. “Fast R-CNN”. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 1440–1448, Santiago, Chile, 7–13 December 2015. IEEE.
- [40] REDMON, J., DIVVALA, S., GIRSHICK, R., et al. “You Only Look Once: Unified, Real-Time Object Detection”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 779–788, Las Vegas, NV, USA, 27–30 June 2016. IEEE.
- [41] REDMON, J., FARHADI, A. “YOLO9000: Better, Faster, Stronger”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6517–6525, Honolulu, HI, USA, 21–26 July 2017. IEEE.
- [42] MAHADEVAN, V., LI, W., BHALODIA, V., et al. “Anomaly Detection in Crowded Scenes”. In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 1975–1981, 2010.
- [43] DORE, A., SOTO, M., REGAZZONI, C. S. “Bayesian Tracking for Video Analytics”, *IEEE Signal Processing Magazine*, v. 27, n. 5, pp. 46–55, 2010.
- [44] SUBUDHI, B. N., NANDA, P. K., GHOSH, A. “A Change Information Based Fast Algorithm for Video Object Detection and Tracking”, *IEEE Transactions on Circuits and Systems for Video Technology*, v. 21, 2011.
- [45] SALIGRAMA, V., KONRAD, J., JODOIN, P.-M. “Video Anomaly Identification”, *IEEE Signal Processing Magazine*, v. 27, n. 5, pp. 18–33, 2010.
- [46] LIN, Y., TONG, Y., CAO, Y., et al. “Visual-Attention-Based Background Modeling for Detecting Infrequently Moving Objects”, *IEEE Transactions on Circuits and Systems for Video Technology*, v. 27, n. 6, pp. 1208–1221, 2016.
- [47] CHANG, L., ZHAO, H., ZHAI, S., et al. “Robust Abandoned Object Detection and Analysis Based on Online Learning”. In: *Proceedings of the IEEE International Conference on Robotics and Biomimetics*, pp. 940–945, 2013.
- [48] CHENG, L., GONG, M., SCHUURMANS, D., et al. “Real-Time Discriminative Background Subtraction”, *IEEE Transactions on Image Processing*, v. 20, n. 5, pp. 1401–1414, 2010.
- [49] JODOIN, P.-M., SALIGRAMA, V., KONRAD, J. “Behavior Subtraction”, *IEEE Transactions on Image Processing*, v. 21, n. 9, pp. 4244–4255, 2012.

- [50] ROMANONI, A., MATTEUCCI, M., SORRENTI, D. G. “Background Subtraction by Combining Temporal and Spatio-Temporal Histograms in the Presence of Camera Movement”, *Machine Vision and Applications*, v. 25, n. 6, pp. 1573–1584, 2014.
- [51] TOMIOKA, Y., TAKARA, A., KITAZAWA, H. “Generation of an Optimum Patrol Course for Mobile Surveillance Camera”, *IEEE Transactions on Circuits and Systems for Video Technology*, v. 22, n. 2, pp. 216–224, 2011.
- [52] JARDIM, E., THOMAZ, L., DA SILVA, E. A. B., et al. “Domain-Transformable Sparse Representation for Anomaly Detection in Moving-Camera Videos”. In: *IEEE Transactions on Image Processing*, 2020.
- [53] NAYAK, R., PATI, U. C., DAS, S. K. “A Comprehensive Review on Deep Learning-Based Methods for Video Anomaly Detection”, *Image and Vision Computing*, p. 104078, 2020.
- [54] ZHAO, B., FEI-FEI, L., XING, E. P. “Online Detection of Unusual Events in Videos via Dynamic Sparse Coding”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3313–3320. IEEE, 2011.
- [55] LUO, W., LIU, W., LIAN, D., et al. “Video Anomaly Detection with Sparse Coding Inspired Deep Neural Networks”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.
- [56] LI, A., MIAO, Z., CEN, Y., et al. “Abnormal Event Detection in Surveillance Videos Based on Low-Rank and Compact Coefficient Dictionary Learning”, *Pattern Recognition*, v. 108, pp. 107355, 2020.
- [57] ZHOU, S., SHEN, W., ZENG, D., et al. “Spatial–Temporal Convolutional Neural Networks for Anomaly Detection and Localization in Crowded Scenes”, *Signal Processing: Image Communication*, v. 47, pp. 358–368, 2016.
- [58] FENG, Y., YUAN, Y., LU, X. “Learning Deep Event Models for Crowd Anomaly Detection”, *Neurocomputing*, v. 219, pp. 548–556, 2017.
- [59] LI, Y., CAI, Y., LIU, J., et al. “Spatio-Temporal Unity Networking for Video Anomaly Detection”, *IEEE Access*, v. 7, pp. 172425–172432, 2019.
- [60] DEEPAK, K., CHANDRAKALA, S., MOHAN, C. K. “Residual Spatiotemporal Autoencoder for Unsupervised Video Anomaly Detection”, *Signal, Image and Video Processing*, v. 15, n. 1, pp. 215–222, 2021.

- [61] WANG, T., QIAO, M., LIN, Z., et al. “Generative Neural Networks for Anomaly Detection in Crowded Scenes”, *IEEE Transactions on Information Forensics and Security*, v. 14, n. 5, pp. 1390–1399, 2018.
- [62] XU, D., YAN, Y., RICCI, E., et al. “Detecting Anomalous Events in Videos by Learning Deep Representations of Appearance and Motion”, *Computer Vision and Image Understanding*, v. 156, pp. 117–127, 2017.
- [63] SULTANI, W., CHEN, C., SHAH, M. “Real-World Anomaly Detection in Surveillance Videos”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6479–6488, 2018.
- [64] “UMN: Unusual Crowd Activity Dataset”. 2006. Disponível em: <http://mha.cs.umn.edu/proj_events.shtml#crowd>. Accessed: 2021-06-28.
- [65] LI, W., MAHADEVAN, V., VASCONCELOS, N. “Anomaly Detection and Localization in Crowded Scenes”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 36, n. 1, pp. 18–32, 2013.
- [66] LU, C., SHI, J., JIA, J. “Abnormal Event Detection at 150 FPS in Matlab”. In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2720–2727, 2013.
- [67] GOYETTE, N., JODOIN, P.-M., PORIKLI, F., et al. “Changetection.net: A New Change Detection Benchmark Dataset”. In: *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pp. 1–8, 2012.
- [68] LI, L., LUO, R., MA, R., et al. “Evaluation of an IVS System for Abandoned Object Detection on PETS 2006 Datasets”. In: *Proceedings of the IEEE Workshop PETS*, pp. 91–98, 2006.
- [69] BRANCH, H. O. S. D. “Imagery Library for Intelligent Detection Systems (i-LIDS)”. In: *Proceedings of the Conference on Crime and Security*, pp. 445–448, 2006.
- [70] NAKAHATA, M. T., THOMAZ, L. A., DA SILVA, A. F., et al. “Anomaly Detection with a Moving Camera Using Spatio-Temporal Codebooks”, *Journal of Multidimensional Systems and Signal Processing*, v. 29, n. 3, pp. 1025–1054, 2018.
- [71] THOMAZ, L. A., JARDIM, E., DA SILVA, A. F., et al. “Anomaly Detection in Moving-Camera Video Sequences Using Principal Subspace Analysis”,

IEEE Transactions on Circuits and Systems, v. 65, n. 3, pp. 1003–1015, 2017.

- [72] JARDIM, E., BIAN, X., DA SILVA, E. A. B., et al. “On the Detection of Abandoned Objects with a Moving Camera Using Robust Subspace Recovery and Sparse Representation”. In: *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 1295–1299, 2015.
- [73] “VDAO-200: 200-Frame Excerpts from VDAO Database”. 2017. Disponível em: <http://www.smt.ufrj.br/~tvdigital/database/research>. Accessed: 2021-06-28.
- [74] ROSHTKHARI, M. J., LEVINE, M. D. “An On-Line, Real-Time Learning Method for Detecting Anomalies in Videos Using Spatio-Temporal Compositions”, *Journal of Computer Vision and Image Understanding*, v. 117, n. 10, pp. 1436–1452, 2013.
- [75] KEOGH, E. “Exact Indexing of Dynamic Time Warping”. In: *Proceedings of the International Conference on Very Large Data Bases*, pp. 406–417, 2002.
- [76] BREIMAN, L. “Random Forests”, *Journal of Machine Learning*, v. 45, n. 1, pp. 5–32, 2001.
- [77] BLACKBURN, J., RIBEIRO, E. “Human Motion Recognition Using Isomap and Dynamic Time Warping”. In: *Workshop on Human Motion*, pp. 285–298. Springer, 2007.
- [78] ZHAO, J., ITTI, L. “ShapeDTW: Shape Dynamic Time Warping”, *Journal of Pattern Recognition*, v. 74, pp. 171–184, 2018.
- [79] KEOGH, E. J., PAZZANI, M. J. “Derivative Dynamic Time Warping”. In: *Proceedings of the 2001 SIAM International Conference on Data Mining*, pp. 1–11. SIAM, 2001.
- [80] CARVALHO, G. H. F., THOMAZ, L. A., DA SILVA, A. F., et al. “Anomaly Detection with a Moving Camera Using Multiscale Video Analysis”, *Multidimensional Systems and Signal Processing*, v. 30, n. 1, pp. 311–342, 2019.
- [81] PADILLA, R. “Aligned VDAO frames”. 2019. Disponível em: https://github.com/rafaelpadilla/DeepLearning-VDAO/blob/master/VDAO_Alignment.md.

- [82] DENG, J., DONG, W., SOCHER, R., et al. “Imagenet: A Large-Scale Hierarchical Image Database”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009.
- [83] TAN, M., LE, Q. V. “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks”, *arXiv preprint arXiv: 1905.11946*, 2019.
- [84] ZAGORUYKO, S., KOMODAKIS, N. “Wide Residual Networks”, *arXiv preprint arXiv: 1605.07146*, 2016.
- [85] XIE, S., GIRSHICK, R., DOLLÁR, P., et al. “Aggregated Residual Transformations for Deep Neural Networks”. In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 1492–1500, 2017.
- [86] DIAS, T. L. B., TAVARES, L. G., PADILLA, R., et al. “Siamese Networks for Bounding-Box to Silhouette Annotation of Video Databases”, *XXXIII Simpósio Brasileiro de Telecomunicações*, 2020. doi: 10.14209/SBRT.2020.1570645009.
- [87] DIETTERICH, T. G. “Ensemble Learning”, *The Handbook of Brain Theory and Neural Networks Magazine*, v. 2, pp. 110–125, 2002.
- [88] FREUND, Y., SCHAPIRE, R. E. “Experiments with a New Boosting Algorithm”. In: *Proceedings of the International Conference on Machine Learning*, pp. 148–156, 1996.
- [89] DIETTERICH, T. G. “Machine Learning Research: Four Current Direction”, *Artificial Intelligence Magazine*, v. 4, pp. 97–136, 1997.
- [90] SHOTTON, J., FITZGIBBON, A., COOK, M., et al. “Real-time human Pose Recognition in Parts from Single Depth Images”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1297–1304, 2011.
- [91] MOOSMANN, F., TRIGGS, B., JURIE, F. “Fast Discriminative Visual Codebooks Using Randomized Clustering Forests”. In: *Proceedings of the IEEE Conference on Advances in Neural Information Processing Systems*, pp. 985–992, 2007.
- [92] OSHIRO, T. M., PEREZ, P. S., BARANAUSKAS, J. A. “How Many Trees in a Random Forest?” In: *International Workshop on Machine Learning and Data Mining in Pattern Recognition*, pp. 154–168. Springer, 2012.

- [93] REN, S., CAO, X., WEI, Y., et al. “Global Refinement of Random Forest”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 723–730, 2015.
- [94] THORNTON, C., HUTTER, F., HOOS, H. H., et al. “Auto-WEKA: Combined Selection and Hyperparameter Optimization of Classification Algorithms”. In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 847–855. ACM, 2013.
- [95] JIMÉNEZ, Á. B., LÁZARO, J. L., DORRONSORO, J. R. “Finding Optimal Model Parameters by Discrete Grid Search”. In: *Innovations in Hybrid Intelligent Systems*, Springer, pp. 120–127, Berlin, Heidelberg, Germany, 2007.
- [96] EISAVI, V., HOMAYOUNI, S., YAZDI, A. M., et al. “Land Cover Mapping Based on Random Forest Classification of Multitemporal Spectral and Thermal Images”, *Environmental Monitoring and Assessment*, v. 187, n. 5, pp. 291, 2015.
- [97] BERGSTRA, J., BENGIO, Y. “Random Search for Hyper-parameter Optimization”, *Journal of Machine Learning Research*, v. 13, pp. 281–305, 2012.
- [98] REIMERS, N., GUREVYCH, I. “Optimal Hyperparameters for Deep LSTM-Networks for Sequence Labeling Tasks”, *arXiv preprint arXiv: 1707.06799*, 2017.
- [99] FEURER, M., SPRINGENBERG, J. T., HUTTER, F. “Initializing Bayesian Hyperparameter Optimization via Meta-Learning”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*, 2015.
- [100] SNOEK, J., LAROCHELLE, H., ADAMS, R. P. “Practical Bayesian Optimization of Machine Learning Algorithms”. In: *Proceedings of the International Conference on Neural Information Processing Systems*, pp. 2951–2959, 2012.
- [101] RASMUSSEN, C. E. “Gaussian Processes in Machine Learning”. In: *Summer School on Machine Learning*, pp. 63–71. Springer, 2003.
- [102] NOGUEIRA, F. “Bayesian Optimization: Open source constrained global optimization tool for Python”. 2014–. Disponível em: <<https://github.com/fmfn/BayesianOptimization>>. Accessed: 2021-06-28.

- [103] CARVALHO, G. *Automatic Detection of Abandoned Objects with a Moving Camera Using Multiscale Video Analysis*. Tese de Doutorado, D. Sc. thesis, Federal University of Rio de Janeiro, Brazil, 2015.
- [104] THOMAZ, L. A., DA SILVA, A. F., DA SILVA, E. A. B., et al. “Abandoned Object Detection Using Operator-Space Pursuit”. In: *Proceedings of the IEEE International Conference on Image Processing*, pp. 1980–1984, 2015.
- [105] LI, J., CHENG, K., WANG, S., et al. “Feature Selection: A Data Perspective”, *ACM Computing Surveys (CSUR)*, v. 50, n. 6, pp. 1–45, 2017.
- [106] CHANDRASHEKAR, G., SAHIN, F. “A Survey on Feature Selection Methods”, *Computers & Electrical Engineering*, v. 40, n. 1, pp. 16–28, 2014.
- [107] BÁSCONES, D., GONZÁLEZ, C., MOZOS, D. “Hyperspectral Image Compression Using Vector Quantization, PCA and JPEG2000”, *Remote sensing*, v. 10, n. 6, pp. 907, 2018.
- [108] YANG, J., ZHANG, D., FRANGI, A. F., et al. “Two-Dimensional PCA: A New Approach to Appearance-Based Face Representation and Recognition”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 26, n. 1, pp. 131–137, 2004.
- [109] ZUO, W., ZHANG, D., YANG, J., et al. “BDPCA Plus LDA: A Novel Fast Feature Extraction Technique for Face Recognition”, *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, v. 36, n. 4, pp. 946–953, 2006.
- [110] GORBAN, A. N., KÉGL, B., WUNSCH, D. C., et al. *Principal Manifolds for Data Visualization and Dimension Reduction*, v. 58. Springer, 2008.
- [111] YIN, H. “ViSOM: A Novel Method for Multivariate Data Projection and Structure Visualization”, *IEEE Transactions on Neural Networks*, v. 13, n. 1, pp. 237–243, 2002.
- [112] ULLAH, A., MUHAMMAD, K., HUSSAIN, T., et al. “Event-Oriented 3D Convolutional Features Selection and Hash Codes Generation Using PCA for Video Retrieval”, *IEEE Access*, v. 8, pp. 196529–196540, 2020.
- [113] BELARBI, M. A., MAHMOUDI, S., BELALEM, G. “PCA as Dimensionality Reduction for Large-Scale Image Retrieval Systems”, *International Journal of Ambient Computing and Intelligence (IJACI)*, v. 8, n. 4, pp. 45–58, 2017.

- [114] KIRBY, M., SIROVICH, L. “Application of the Karhunen-Loeve Procedure for the Characterization of Human Faces”, *IEEE Transactions on Pattern analysis and Machine intelligence*, v. 12, n. 1, pp. 103–108, 1990.
- [115] TURK, M., PENTLAND, A. “Eigenfaces for Recognition”, *Journal of cognitive neuroscience*, v. 3, n. 1, pp. 71–86, 1991.
- [116] TOĞAÇAR, M., ERGEN, B., CÖMERT, Z. “Detection of Lung Cancer on Chest CT Images Using Minimum Redundancy Maximum Relevance Feature Selection Method with Convolutional Neural Networks”, *Biocybernetics and Biomedical Engineering*, v. 40, n. 1, pp. 23–39, 2020.
- [117] AHMADI, M., SHARIFI, A., JAFARIAN FARD, M., et al. “Detection of Brain Lesion Location in MRI Images Using Convolutional Neural Network and Robust PCA”, *International Journal of Neuroscience*, pp. 1–12, 2021.
- [118] GUPTA, T., GANDHI, T. K., GUPTA, R., et al. “Classification of Patients with Tumor Using MR FLAIR Images”, *Pattern Recognition Letters*, 2017.
- [119] WANG, Z., ZOU, C., CAI, W. “Small Sample Classification of Hyperspectral Remote Sensing Images Based on Sequential joint Feeping Learning Model”, *IEEE Access*, v. 8, pp. 71353–71363, 2020.
- [120] LAZCANO, R., MADROÑAL, D., FABELLO, H., et al. “Adaptation of an Iterative PCA to a Manycore Architecture for Hyperspectral Image Processing”, *Journal of Signal Processing Systems*, v. 91, n. 7, pp. 759–771, 2019.
- [121] XU, J.-L., GOWEN, A. A. “Spatial-Spectral Analysis Method Using Texture Features Combined with PCA for Information Extraction in Hyperspectral Images”, *Journal of Chemometrics*, v. 34, n. 2, pp. e3132, 2020.
- [122] MA, J., YUAN, Y. “Dimension Reduction of Image Deep Feature Using PCA”, *Journal of Visual Communication and Image Representation*, v. 63, pp. 102578, 2019.
- [123] XI, Y. T., RAMADGE, P. J. “Separable PCA for Image Classification”. In: *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 1805–1808. IEEE, 2009.
- [124] PAN, V. Y., CHEN, Z. Q. “The Complexity of the Matrix Eigenproblem”. In: *Proceedings of the Thirty-First Annual ACM Symposium on Theory*

of Computing, STOC '99, p. 507–516, New York, NY, USA, 1999. Association for Computing Machinery. ISBN: 1581130678. doi: 10.1145/301250.301389. Disponível em: <<https://doi.org/10.1145/301250.301389>>.

- [125] LIU, R., GILLIES, D. F. “Overfitting in Linear Feature Extraction for Classification of High-Dimensional Image Data”, *Pattern Recognition*, v. 53, pp. 73–86, 2016.
- [126] ZUO, W., ZHANG, D., WANG, K. “Bidirectional PCA with Assembled Matrix Distance Metric for Image Recognition”, *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, v. 36, n. 4, pp. 863–872, 2006.
- [127] ZUO, W., WANG, K., ZHANG, D. “Bi-Directional PCA with Assembled Matrix Distance Metric”. In: *Proceedings of the IEEE International Conference on Image Processing*, v. 2, pp. II–958. IEEE, 2005.
- [128] LIANG, Z., ZHANG, D., SHI, P. “The Theoretical Analysis of GLRAM and its Applications”, *Pattern Recognition*, v. 40, n. 3, pp. 1032–1041, 2007.
- [129] WOLD, H. “Estimation of Principal Components and Related Models by Iterative Least Squares”, *Multivariate Analysis*, pp. 391–420, 1966.
- [130] BISHOP, C. M. “Bayesian PCA”, *Advances in Neural Information Processing Systems*, pp. 382–388, 1999.
- [131] TROYANSKAYA, O., CANTOR, M., SHERLOCK, G., et al. “Missing Value Estimation Methods for DNA Microarrays”, *Bioinformatics*, v. 17, n. 6, pp. 520–525, 2001.
- [132] STACKLIES, W., REDESTIG, H., SCHOLZ, M., et al. “pcaMethods — A Bioconductor Package Providing PCA Methods for Incomplete Data”, *Bioinformatics*, v. 23, n. 9, pp. 1164–1167, 2007.
- [133] ANDRECUT, M. “Parallel GPU Implementation of Iterative PCA Algorithms”, *Journal of Computational Biology*, v. 16, n. 11, pp. 1593–1599, 2009.
- [134] WENG, J., ZHANG, Y., HWANG, W.-S. “A Fast Algorithm for Incremental Principal Component Analysis”. In: *International Conference on Intelligent Data Engineering and Automated Learning*, pp. 876–881. Springer, 2003.

- [135] WENG, J., ZHANG, Y., HWANG, W.-S. “Candid Covariance-Free Incremental Principal Component Analysis”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 25, n. 8, pp. 1034–1040, 2003.
- [136] WANG, K., WANG, H., WU, M., et al. “A Method for Spectral Image Registration Based on Feature Maximum Submatrix”, *EURASIP Journal on Image and Video Processing*, v. 2018, n. 1, pp. 1–11, 2018.
- [137] HORN, B. K., SCHUNCK, B. G. “Determining Optical Flow”, *Artificial Intelligence*, v. 17, n. 1-3, pp. 185–203, 1981.
- [138] DERPANIS, K. G. “Overview of the RANSAC Algorithm”, *Image Rochester NY*, v. 4, n. 1, pp. 2–3, 2010.
- [139] PIZER, S. M., AMBURN, E. P., AUSTIN, J. D., et al. “Adaptive Histogram Equalization and its Variations”, *Computer Vision, Graphics, and Image Processing*, v. 39, n. 3, pp. 355–368, 1987.
- [140] DA SILVA, A. F., THOMAZ, L. A., DA SILVA, E. A. B., et al. “Alinhamento de Sinais Obtidos em Trajetórias Fechadas Utilizando um Conjunto Genérico de Sensores (Alignment of Signals Obtained in Closed Trajectories Using a Generic Set of Sensors)”, *XXXI Simpósio Brasileiro de Telecomunicações*, pp. 329–333, 2016. doi: 10.14209/SBRT.2016.210.
- [141] ZHAO, W., NISTER, D., HSU, S. “Alignment of Continuous Video onto 3d Point Clouds”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 27, n. 8, pp. 1305–1318, 2005.
- [142] FREMONT, V., RODRIGUEZ F, S. A., BONNIFAIT, P. “Circular Targets for 3d Alignment of Video and Lidar Sensors”, *Advanced Robotics*, v. 26, n. 18, pp. 2087–2113, 2012.
- [143] BOUGHORBEL, S., JARRAY, F., EL-ANBARI, M. “Optimal Classifier for Imbalanced Data Using Matthews Correlation Coefficient Metric”, *PLoS one*, v. 12, n. 6, pp. e0177678, 2017.
- [144] CHICCO, D., JURMAN, G. “The Advantages of the Matthews Correlation Coefficient (MCC) over F1 Score and Accuracy in Binary Classification Evaluation”, *BMC genomics*, v. 21, n. 1, pp. 1–13, 2020.
- [145] CHICCO, D., TÖTSCH, N., JURMAN, G. “The Matthews Correlation Coefficient (MCC) is More Reliable than Balanced Accuracy, Bookmaker Informedness, and Markedness in Two-Class Confusion Matrix Evaluation”, *BioData mining*, v. 14, n. 1, pp. 1–22, 2021.

- [146] ABHISHEK, K., HAMARNEH, G. “Matthews Correlation Coefficient Loss for Deep Convolutional Networks: Application to Skin Lesion Segmentation”, *arXiv preprint arXiv:2010.13454*, 2020.
- [147] DOUGHERTY, E. *Mathematical Morphology in Image Processing*. CRC press, 2018.
- [148] PAL, S., CHATTERJEE, S. “Mathematical Morphology Aided Optic Disk Segmentation from Retinal Images”. In: *2017 3rd International Conference on Condition Assessment Techniques in Electrical Systems (CATCON)*, pp. 380–385. IEEE, 2017.
- [149] MISHRA, M., PANIGRAHI, R. R., ROUT, P. K. “A Combined Mathematical Morphology and Extreme Learning Machine Techniques Based Approach to Micro-Grid Protection”, *Ain Shams Engineering Journal*, v. 10, n. 2, pp. 307–318, 2019.
- [150] SOILLE, P. *Morphological Image Analysis: Principles and Applications*. Berlin, Heidelberg, Springer Science & Business Media, 2013.
- [151] SHIH, F. Y., SHEN, Y., ZHONG, X. “Development of Deep Learning Framework for Mathematical Morphology”, *International Journal of Pattern Recognition and Artificial Intelligence*, v. 33, n. 06, pp. 1954024, 2019.
- [152] MARAGOS, P. “Differential Morphology and Image Processing”, *IEEE Transactions on Image Processing*, v. 5, n. 6, pp. 922–937, 1996.
- [153] NOGUEIRA, K., CHANUSSOT, J., MURA, M. D., et al. “An Introduction to Deep Morphological Networks”, *arXiv preprint arXiv:1906.01751*, 2019.
- [154] HU, Y., HUBER, A., ANUMULA, J., et al. “Overcoming the Vanishing Gradient Problem in Plain Recurrent Networks”, *arXiv preprint arXiv:1801.06105*, 2018.
- [155] ROODSCHILD, M., SARDIÑAS, J. G., WILL, A. “A New Approach for the Vanishing Gradient Problem on Sigmoid Activation”, *Progress in Artificial Intelligence*, v. 9, n. 4, pp. 351–360, 2020.
- [156] BENGIO, Y., LAMBLIN, P., POPOVICI, D., et al. “Greedy Layer-Wise Training of Deep Networks”, *Advances in Neural Information Processing Systems*, v. 19, pp. 153, 2007.
- [157] KINGMA, D. P., BA, J. “Adam: A Method for Stochastic Optimization”, *arXiv preprint arXiv:1412.6980*, 2014.

- [158] CHENG, F.-C., HUANG, S.-C., RUAN, S.-J. “Illumination-Sensitive Background Modeling Approach for Accurate Moving Object Detection”, *IEEE Transactions on Broadcasting*, v. 57, n. 4, pp. 794–801, 2011.
- [159] KHAN, F. S., ANWER, R. M., VAN DE WEIJER, J., et al. “Color Attributes for Object Detection”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3306–3313, Providence, RI, USA, 16–21 June 2012. IEEE.
- [160] OUYANG, W., WANG, X. “A Discriminative Deep Model for Pedestrian Detection with Occlusion Handling”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3258–3265, Providence, RI, USA, 16–21 June 2012. IEEE.
- [161] ZOU, Z., SHI, Z., GUO, Y., et al. “Object Detection in 20 Years: A Survey”, *arXiv preprint arXiv:1905.05055*, 2019.
- [162] LECUN, Y., BOTTOU, L., BENGIO, Y., et al. “Gradient-Based Learning Applied to Document Recognition”, *Proceedings of the IEEE*, v. 86, n. 11, pp. 2278–2324, 1998.
- [163] JOCHER, G., STOKEN, A., BOROVEC, J., et al. “ultralytics/yolov5: v3.0”. <https://github.com/ultralytics/yolov5>, 2020.
- [164] DOLLÁR, P., WOJEK, C., SCHIELE, B., et al. “Pedestrian Detection: An Evaluation of the State of the Art”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 34, n. 4, pp. 743–761, 2012.
- [165] OHN-BAR, E., TRIVEDI, M. M. “To Boost or Not to Boost? On the Limits of Boosted Trees for Object Detection”. In: *Proceedings of the International Conference on Pattern Recognition (ICPR)*, pp. 3350–3355, Cancun, Mexico, 4–8 December 2016. IEEE.
- [166] VIOLA, P., JONES, M. “Robust Real-Time Object Detection”, *International Journal of Computer Vision*, v. 57, n. 34-47, pp. 137–154, 2004.
- [167] WANG, X. “Intelligent Multi-Camera Video Surveillance: A Review”, *Pattern Recognition Letters*, v. 34, n. 1, pp. 3–19, 2013.
- [168] BALTIERI, D., VEZZANI, R., CUCCHIARA, R. “3DPes: 3D People Dataset for Surveillance and Forensics”. In: *Proceedings of the Joint ACM Workshop on Human Gesture and Behavior Understanding*, pp. 59–64, Scottsdale, Arizona, USA, 28 November – 1 December 2011. ACM.

- [169] COOTES, T. F., TAYLOR, C. J. “Statistical Models of Appearance for Medical Image Analysis and Computer Vision”. In: *Proceedings of the Medical Imaging 2001: Image Processing*, pp. 236–248, San Diego, CA, USA, 3 July 2001. SPIE.
- [170] ZHAO, Z., ZHENG, P., XU, S., et al. “Object Detection With Deep Learning: A Review”, *IEEE Transactions on Neural Networks and Learning Systems*, v. 30, n. 11, pp. 3212–3232, 2019.
- [171] PADILLA, R., NETTO, S. L., DA SILVA, E. A. B. “A Survey on Performance Metrics for Object-Detection Algorithms”. In: *Proceedings of the International Conference on Systems, Signals and Image Processing (IWSSIP)*, pp. 237–242, Niteroi, Brazil, 1–3 July 2020. IEEE.
- [172] EVERINGHAM, M., ESLAMI, S. M. A., VAN GOOL, L., et al. “The Pascal Visual Object Classes Challenge: A Retrospective”, *International Journal of Computer Vision*, v. 111, n. 1, pp. 98–136, 2015.
- [173] ATTNEAVE, F., ARNOULT, M. D. “The Quantitative Study of Shape and Pattern Perception”, *Psychological Bulletin*, v. 53, n. 6, pp. 452–471, 1956.
- [174] ROBERTS, L. G. *Machine Perception of Three-Dimensional Solids*. Tese de Doutorado, Massachusetts Institute of Technology, 1963.
- [175] LOWE, D. G., BINFORD, T. O. “The Recovery of Three-Dimensional Structure from Image Curves”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 7, n. 3, pp. 320–326, 1985.
- [176] HARRIS, C. G., STEPHENS, M. “A Combined Corner and Edge Detector”. In: *Proceedings of the Alvey Vision Conference*, pp. 23.1–23.6, Manchester, UK, 31 August – 2 September 1988. Alvey Vision Club.
- [177] LUCAS, B. D., KANADE, T. “An Iterative Image Registration Technique with an Application to Stereo Vision”. In: *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 674–679, Vancouver, British Columbia, Canada, 24–28 August 1981. Morgan Kaufmann Publishers.
- [178] SHI, J., TOMASI. “Good Features to Track”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 593–600, Seattle, WA, USA, 21–23 June 1994. IEEE.
- [179] LOWE, D. G. “Distinctive Image Features from Scale-Invariant Keypoints”, *International Journal of Computer Vision*, v. 60, n. 2, pp. 91–110, 2004.

- [180] BAY, H., TUYTELAARS, T., VAN GOOL, L. “SURF: Speeded Up Robust Features”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 404–417, Graz, Austria, 7–13 May 2006. Springer.
- [181] NGUYEN, T., PARK, E.-A., HAN, J., et al. “Object Detection Using Scale Invariant Feature Transform”. In: *In Genetic and Evolutionary Computing*, Springer, pp. 65–72, Berlin/Heidelberg, Germany, 2014.
- [182] ZHOU, H., YUAN, Y., SHI, C. “Object Tracking Using SIFT Features and Mean Shift”, *Computer Vision and Image Understanding*, v. 113, n. 3, pp. 345–352, 2009.
- [183] DALAL, N., TRIGGS, B. “Histograms of Oriented Gradients for Human Detection”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 886–893, San Diego, CA, USA, 20–25 June 2005. IEEE.
- [184] MIZUNO, K., TERACHI, Y., TAKAGI, K., et al. “Architectural Study of HOG Feature Extraction Processor for Real-Time Object Detection”. In: *In IEEE Workshop on Signal Processing Systems*, pp. 197–202, Quebec City, QC, Canada, 17–19 October 2012. IEEE.
- [185] SZEGEDY, C., LIU, W., JIA, Y., et al. “Going Deeper with Convolutions”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–9, Boston, MA, USA, 7–12 June 2015. IEEE.
- [186] SIMONYAN, K., ZISSERMAN, A. “Very Deep Convolutional Networks for Large-Scale Image Recognition”. In: *Proceedings of the International Conference on Learning Representations (ICLR)*, pp. 1–14, San Diego, CA, USA, 7–9 May 2015. online.
- [187] HINTON, G. E., OSINDERO, S., TEH, Y.-W. “A Fast Learning Algorithm for Deep Belief Nets”, *Neural Computation*, v. 18, n. 7, pp. 1527–1554, 2006.
- [188] HINTON, G. E., SALAKHUTDINOV, R. R. “Reducing the Dimensionality of Data with Neural Networks”, *Science*, v. 313, n. 5786, pp. 504–507, 2006.
- [189] ZOPH, B., CUBUK, E. D., GHIASI, G., et al. “Learning Data Augmentation Strategies for Object Detection”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 566–583, Glasgow, UK, 23–28 August 2020. Springer.

- [190] LOEY, M., MANOGARAN, G., KHALIFA, N. E. M. “A Deep Transfer Learning Model with Classical Data Augmentation and CGAN to Detect COVID-19 from Chest CT Radiography Digital Images”, *Neural Computing and Applications*, pp. 1–13, 2020.
- [191] GONZÁLEZ, R. E., MUNOZ, R. P., HERNÁNDEZ, C. A. “Galaxy Detection and Identification Using Deep Learning and Data Augmentation”, *Astronomy and computing*, v. 25, pp. 103–109, 2018.
- [192] SERMANET, P., EIGEN, D., ZHANG, X., et al. “OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks”. In: *International Conference on Learning Representations (ICLR)*, pp. 1–16, Banff, Canada, 14–16 April 2014. online.
- [193] LIU, W., ANGUELOV, D., ERHAN, D., et al. “SSD: Single Shot MultiBox Detector”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 21–37, Amsterdam, The Netherlands, 11-14 October 2016. Springer.
- [194] BOCHKOVSKIY, A., WANG, C.-Y., LIAO, H.-Y. M. “YOLOv4: Optimal Speed and Accuracy of Object Detection”, *arXiv preprint arXiv:2004.10934*, 2020.
- [195] DAI, J., LI, Y., HE, K., et al. “R-FCN: Object Detection Via Region-Based Fully Convolutional Networks”. In: *Proceedings of the International Conference on Neural Information Processing Systems (NeurIPS)*, pp. 379–387, Barcelona, Spain, 5–10 December 2016. Curran Associates.
- [196] GU, J., HU, H., WANG, L., et al. “Learning Region Features for Object Detection”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 392–406, Munich, Germany, 8–14 September 2018. Springer.
- [197] HU, H., GU, J., ZHANG, Z., et al. “Relation Networks for Object Detection”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3588–3597, Salt Lake City, UT, USA, 18-22 June 2018. IEEE.
- [198] POŁAP, D. “An Adaptive Genetic Algorithm as a Supporting Mechanism for Microscopy Image Analysis in a Cascade of Convolution Neural Networks”, *Applied Soft Computing Journal*, v. 97, pp. 1–11, 2020.
- [199] LI, Y., WANG, H., DANG, L. M., et al. “A Deep Learning-based Hybrid Framework for Object Detection and Recognition in Autonomous Driving”, *IEEE Access*, v. 8, pp. 194228–194239, 2020.

- [200] CHEN, Y., ZHOU, W. “Hybrid-Attention Network for RGB-D Salient Object Detection”, *Applied Sciences*, v. 10, n. 17, pp. 5806, 2020.
- [201] ZHANG, P., LIU, W., LEI, Y., et al. “Hyperfusion-Net: Hyper-Densely Reflective Feature Fusion for Salient Object Detection”, *Pattern Recognition*, v. 93, pp. 521–533, 2019.
- [202] LITJENS, G., KOOL, T., BEJNORDI, B. E., et al. “A Survey on Deep Learning in Medical Image Analysis”, *Medical image analysis*, v. 42, pp. 60–88, 2017.
- [203] CAO, Z., DUAN, L., YANG, G., et al. “Breast Tumor Detection in Ultrasound Images Using Deep Learning”. In: *Proceedings of the International Workshop on Patch-based Techniques in Medical Imaging*, pp. 121–128, online, 31 August 2017. Springer.
- [204] JAEGER, P. F., KOHL, S. A. A., BICKELHAUPT, S., et al. “Retina U-Net: Embarrassingly Simple Exploitation of Segmentation Supervision for Medical Object Detection”. In: *Proceedings of Machine Learning for Health Workshop (ML4H)*, pp. 171–183, Vancouver, Canada, 13–14 December 2020. PMLR.
- [205] LI, Z., DONG, M., WEN, S., et al. “CLU-CNNs: Object detection for Medical Images”, *Neurocomputing*, v. 350, pp. 53–59, 2019.
- [206] LIN, T.-Y., MAIRE, M., BELONGIE, S., et al. “Microsoft COCO: Common Objects in Context”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 740–755, Zurich, Switzerland, 6–12 September 2014. Springer.
- [207] WADA, K. “labelme: Image Polygonal Annotation with Python”. <https://github.com/wkentaro/labelme>, 2016.
- [208] LIN, T. “LabelImg”. <https://github.com/tzutalin/labelImg>, 2015.
- [209] WADA, K. “VoTT: Visual Object Tagging Tool”. <https://github.com/Microsoft/VoTT>, 2018.
- [210] SEKACHEV, B., MANOVICH, N., ZHILTSOV, M., et al. “opencv/cvat v1.1.0”. <http://doi.org/10.5281/zenodo.4009388>, 2020.
- [211] DUTTA, A., ZISSERMAN, A. “The VIA Annotation Software for Images, Audio and Video”. In: *Proceedings of the ACM International Conference on Multimedia*, pp. 2276–2279, Nice, France, 21–25 October 2019. ACM.

- [212] ABADI, M., AGARWAL, A., BARHAM, P., et al. “TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems”. <https://www.tensorflow.org/>, 2015.
- [213] LAW, H., DENG, J. “Cornernet: Detecting Objects as Paired Keypoints”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 734–750, Munich, Germany, 8–14 September 2018. Springer.
- [214] TAN, M., PANG, R., LE, Q. V. “EfficientDet: Scalable and Efficient Object Detection”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10781–10790, Seattle, USA, 14–19 June 2020. IEEE.
- [215] LIU, S., HUANG, D., OTHERS. “Receptive Field Block Net for Accurate and Fast Object Detection”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 385–400, Munich, Germany, 8–14 September 2018. Springer.
- [216] ZHANG, S., WEN, L., BIAN, X., et al. “Single-Shot Refinement Neural Network for Object Detection”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4203–4212, Salt Lake City, Utah, USA, 18–22 June 2018. IEEE.
- [217] LIN, T.-Y., GOYAL, P., GIRSHICK, R., et al. “Focal Loss for Dense Object Detection”. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 2980–2988, Venice, Italy, 22–29 October 2017. IEEE.
- [218] “Open Images Object Detection RVC 2020 edition”. <https://www.kaggle.com/c/open-images-object-detection-rvc-2020>, 2020. Accessed: 2021-06-28.
- [219] “COCO Detection Challenge (Bounding Box)”. <https://competitions.codalab.org/competitions/20794>, 2019.
- [220] “Datalab Cup: CNN Object Detection”. <https://www.kaggle.com/c/datalabcup-cnn-object-detection>, 2017. Accessed: 2021-06-28.
- [221] “Google AI Open Images - Object Detection Track”. <https://www.kaggle.com/c/google-ai-open-images-object-detection-track>, 2018. Accessed: 2021-06-28.

- [222] “Lyft 3D Object Detection for Autonomous Vehicles”. <https://www.kaggle.com/c/3d-object-detection-for-autonomous-vehicles/>, 2019. Accessed: 2021-06-28.
- [223] “City Intelligence Hackathon”. <https://belvisionhack.ru>, 2018.
- [224] DUDCZYK, J., KAWALEC, A. “Identification of Emitter Sources in the Aspect of Their Fractal Features”, *Bulletin of the Polish Academy of Sciences Technical Sciences: Technical Sciences*, v. 61, n. 3, pp. 623–628, 2013.
- [225] RYBAK, Ł., DUDCZYK, J. “A Geometrical Divide of Data Particle in Gravitational Classification of Moons and Circles Data Sets”, *Entropy*, v. 22, n. 10, pp. 1088–1103, 2020.
- [226] JACCARD, P. “Étude Comparative de la Distribution Florale Dans Une Portion des Alpes et des Jura”, *Bulletin de la Societe Vaudoise des Sciences Naturelles*, v. 37, pp. 547–579, 1901.
- [227] HOSANG, J., BENENSON, R., DOLLÁR, P., et al. “What Makes for Effective Detection Proposals?” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 38, n. 4, pp. 814–830, 2015.
- [228] OKSUZ, K., CAN CAM, B., AKBAS, E., et al. “Localization Recall Precision (LRP): A New Performance Metric for Object Detection”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 504–519, Munich, Germany, 8–14 September 2018. Springer.
- [229] PADILLA, R., NETTO, S. L., DA SILVA, E. A. B. “Metrics for Object Detection”. <https://github.com/rafaelpadilla/Object-Detection-Metrics>, 2018.
- [230] COMPUTER RESEARCH INSTITUTE OF MONTREAL (CRIM). “thelper Package”. <https://thelper.readthedocs.io/en/latest/thelper.optim.html>, 2020.
- [231] PADILLA, R., PASSOS, W. L., DIAS, T. L. B., et al. “Evaluation Tool for Object Detection Metrics”. https://github.com/rafaelpadilla/review_object_detection_metrics, 2020.
- [232] PADILLA, R., PASSOS, W. L., DIAS, T. L. B., et al. “A Comparative Analysis of Object Detection Metrics with a Companion Open-Source Toolkit”, *Electronics*, v. 10, n. 3, 2021. ISSN: 2079-9292. Disponível em: <<https://www.mdpi.com/2079-9292/10/3/279>>.