HAL open science

Learning stochastic geometry models and convolutional neural networks. Application to multiple object detection in aerospatial data sets

Jules Mabon

To cite this version:

Jules Mabon. Learning stochastic geometry models and convolutional neural networks. Application to multiple object detection in aerospatial data sets. Computer Vision and Pattern Recognition [cs.CV]. Université Côte d'Azur, 2023. English. NNT : 2023COAZ4116 . tel-04404849v2

HAL Id: tel-04404849

https://theses.hal.science/tel-04404849v2

Submitted on 19 Jan 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

UNIVERSITÉ CÔTE D'AZUR

ÉCOLE DOCTORALE
SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION

THÈSE DE DOCTORAT

Apprentissage de modèles de géométrie stochastique et réseaux de neurones convolutifs. Application à la détection d'objets multiples dans des jeux de données aérospatiales.

Jules MABON

Ayana, Centre Inria d'Université Côte d'Azur

Présentée en vue de l'obtention du grade de docteur en AUTOMATIQUE TRAITEMENT DU SIGNAL ET DES IMAGES
d'Université Côte d'Azur

Dirigée par : Josiane ZERUBIA, Directrice de Recherche (DRCE), Centre Inria d'Université Côte d'Azur
Co-encadrée par : Mathias ORTNER, Docteur, Senior Expert satellite data analysis and processing, Airbus Defense and Space
Soutenue le : 20 décembre 2023, 10h

Devant le jury, composé de :
Xavier DESCOMBES, Directeur de Recherche (DR1), Centre Inria d'Université Côte d'Azur
Alin ACHIM, Full Professor, Université de Bristol
Pierre CHAINAIS, Professeur des Universités, Centrale Lille Institut
Yann GOUSSEAU, Professeur, Telecom ParisTech

APPRENTISSAGE DE MODÈLES DE GÉOMÉTRIE STOCHASTIQUE ET RÉSEAUX DE NEURONES CONVOLUTIFS. APPLICATION À LA DÉTECTION D'OBJETS MULTIPLES DANS DES JEUX DE DONNÉES AÉROSPATIALES.

Learning Stochastic geometry models and convolutional neural networks. Application to multiple object detection in aerospatial data sets.

Jules MABON

⋈

Jury :

Président du jury

Xavier DESCOMBES, Directeur de Recherche (DR1), Centre Inria d'Université Côte d'Azur

Rapporteurs

Alin ACHIM, Full Professor, Université de Bristol
Yann GOUSSEAU, Professeur, Telecom ParisTech

Examinateurs

Pierre CHAINAIS, Professeur des Universités, Centrale Lille Institut

Directrice de thèse

Josiane ZERUBIA, Directrice de Recherche (DRCE), Centre Inria d'Université Côte d'Azur

Co-encadrant de thèse

Mathias ORTNER, Docteur, Senior Expert satellite data analysis and processing, Airbus Defense and Space

---

Université Côte d'Azur

Jules MABON

Apprentissage de modèles de géométrie stochastique et réseaux de neurones convolutifs. Application à la détection d'objets multiples dans des jeux de données aérospatiales.

xiii+190 p.

Ce document a été préparé avec LaTeX2e et la classe `these-ISSS` version v. 2.10 (mpelleau) modifiée.

Illustrations par J. Mabon, Figures (hors référencées "From (...)") préparées avec PGF/TikZ ou matplotlib par J. Mabon.

Impression : 16/1/2024 – 11:28.

À mes parents, Mamouch, Lucie, Josette & Pompon ♡

Apprentissage de modèles de géométrie stochastique et réseaux de neurones convolutifs. Application à la détection d'objets multiples dans des jeux de données aérospatiales.

Résumé

Les drones et les satellites en orbite basse, dont les CubeSats, sont de plus en plus utilisés pour la surveillance, générant d'importantes masses de données à traiter. L'acquisition d'images satellitaires est sujette aux perturbations atmosphériques, aux occlusions et à une résolution limitée. Pour détecter de petits objets, l'information visuelle est limitée. Cependant, les objets d'intérêt (comme les petits véhicules) ne sont pas uniformément répartis dans l'image, présentant des configurations spécifiques. Ces dernières années, les Réseaux de Neurones Convolutifs (CNN) ont montré des compétences remarquables pour extraire des informations, en particulier les textures. Cependant, modéliser les interactions entre objets nécessite une complexité accrue. Les CNN considèrent généralement les interactions lors d'une étape de post-traitement. En revanche, les Processus Ponctuels permettent de modéliser la vraisemblance des points par rapport à l'image et leurs interactions simultanément. La plupart des modèles stochastiques utilisent des mesures de contraste pour la correspondance à l'image ; elles sont adaptées aux objets à contraste fort et faible complexité du fond. Cependant, les petits véhicules sur les images satellitaires présentent divers niveaux de contraste et une grande variété d'objets de fond et de fausses alarmes. Cette thèse de doctorat propose d'utiliser les CNN pour l'extraction d'informations, combinées aux Processus Ponctuels pour modéliser les interactions, en utilisant les sorties CNN comme données. De plus, nous introduisons une méthode unifiée pour estimer les paramètres du modèle de Processus Ponctuel. Nos résultats montrent l'efficacité de ce modèle sur plusieurs jeux de données de télédétection, avec régularisation géométrique et robustesse accrue pour un nombre limité de paramètres.

Mots-clés : Géométrie Stochastique, Modèles à Base d'Énergies, Détection, Objets Multiples, Images Satellitaires, Très Haute Résolution.

Learning Stochastic geometry models and convolutional neural networks. Application to multiple object detection in aerospatial data sets.

Abstract

Unmanned aerial vehicles and low-orbit satellites, including CubeSats, are increasingly used for wide-area surveillance, generating substantial data for processing. Satellite imagery acquisition is susceptible to atmospheric disruptions, occlusions, and limited resolution, resulting in limited visual data for small object detection. However, the objects of interest (e.g., small vehicles) are unevenly distributed in the image: there are some priors on the structure of the configurations. In recent years, convolutional neural network (CNN) models have excelled at extracting information from images, especially texture details. Yet, modeling object interactions requires a significant increase in model complexity and parameters. CNN models generally treat interaction as a post-processing step. In contrast, Point Processes aim to simultaneously model each point's likelihood in relation to the image (data term) and their interactions (prior term). Most Point Process models rely on contrast measures (foreground vs. background) for their data terms, which work well with clearly contrasted objects and minimal background clutter. However, small vehicles in satellite images exhibit varying contrast levels and a diverse range of background and false alarm objects. In this PhD thesis, we propose harnessing CNN models information extraction abilities in combination with Point Process interaction models, using CNN outputs as data terms. Additionally, we introduce a unified method for estimating Point Process model parameters. Our model demonstrates excellent performance on multiple remote sensing datasets, providing geometric regularization and enhanced noise robustness, all with a minimal parameter footprint.

Keywords: Stochastic Geometry, Energy Based Models, Detection, Multiple objects, Satellite Images, Very High Resolution.

Remerciements

L'équipe d'encadrement et moi-même tenons tout d'abord à remercier BPI France pour le financement de cette thèse de doctorat via le projet LiChIE. Nous remercions aussi l'infrastructure OPAL de l'Université Côte d'Azur pour avoir fourni son support et les ressources de calcul.

Je tiens aussi à remercier toutes les personnes ayant contribué au bon déroulement de ma thèse : les collaborateurs d'Airbus pour leur accueil et retours utiles lors de mes visites à Toulouse ; l'équipe du SI de Sophia (tout particulièrement Francis) pour avoir sauvé mon ordinateur plus d'une fois ; nos assistantes d'équipe Nathalie N., succédant Nathalie B.

Aussi, je tiens particulièrement à remercier ma directrice de thèse Josiane Zerubia ainsi que mon co-encadrant Mathias Ortner pour leurs conseils éclairés et directions avisées.

Je remercie par ailleurs les membres du jury, Pr Alin Achim, Pr Pierre Chainais, Pr Xavier Descombes et Pr Yann Gousseau, pour s'être rendus disponibles, pour leurs questions propices à une riche discussion et leurs retours positifs.

Je tiens à exprimer ma gratitude envers ma famille et mes amis pour leur soutien à distance, mais infaillible. Enfin, mes collègues (et amis) de laboratoire furent aussi d'une grande aide : je tiens à remercier les membres de l'ADSTIC qui m'ont accueillie parmi eux ; les camarades de l'I3S Tomas & Pati, Nina, Romain, Amélie, Laetitia, Juliette, Baptiste, Marie, Margaux, François et bien d'autres ; et bien sûr les collègues de l'Inria Bilel, Camilo, Jhonatan, Louis, Yanick, Priscilla et Martina.

Merci à tous·tes ♡

Table of contents

My publications **1**

Notations & Acronyms **3**

Résumé long en français **5**

Introduction

1 Introduction **15**
- 1.1 Computer vision in remote sensing . . . 16
- 1.2 Satellite image acquisition . . . 17
- 1.3 Object detection . . . 20
  - 1.3.1 Image probability model . . . 20
  - 1.3.2 Point Process . . . 21
  - 1.3.3 Convolutional Neural Networks . . . 22
- 1.4 Proposed approach . . . 23
- 1.5 Thesis structure . . . 23

2 State of the art **25**
- 2.1 Convolutional Neural Networks for object detection . . . 28
  - 2.1.1 The rise of CNN models for object detection . . . 28
  - 2.1.2 CNN for detection in remote sensing . . . 30
  - 2.1.3 Remote sensing datasets . . . 34
- 2.2 Point Processes for object detection . . . 34
  - 2.2.1 Point Process models . . . 35
  - 2.2.2 Sampling the Point Process . . . 35
  - 2.2.3 Point Process parameter estimation . . . 36
- 2.3 Energy Based Models . . . 36
  - 2.3.1 Energy Based Models as generative models . . . 36
  - 2.3.2 Learning EBMs . . . 37
  - 2.3.3 EBM applications to computer vision . . . 37

Model foundations

3 Foundations for Point Processes and Convolutional Neural Networks **41**
- 3.1 Point Process fundamentals . . . 43
  - 3.1.1 Poisson Point Processes . . . 43
  - 3.1.2 Markov Point Processes . . . 44
  - 3.1.3 Markov marked Point Processes . . . 46

3.1.4 Stability conditions . . . 47
3.2 Point Process for object detection . . . 48
3.2.1 Energy model . . . 49
3.3 Sampling the Point Process . . . 50
3.3.1 Markov chains . . . 50
3.3.2 Reversible Jump Monte Carlo Markov chain . . . 52
3.3.3 Jump diffusion . . . 57
3.3.4 Simulated annealing . . . 58
3.3.5 Stopping conditions . . . 59
3.4 CNN fundamentals . . . 60
3.5 Conclusion . . . 62

Contributions

4 Building an Energy Based Model for object detection 67
4.1 Point Process as an energy model . . . 69
4.2 Data terms from a CNN . . . 70
4.2.1 Contrast measures . . . 71
4.2.2 CNN data terms for unmarked points . . . 73
4.2.3 Data energy on marks . . . 80
4.3 Priors on configurations . . . 83
4.3.1 Point priors . . . 83
4.3.2 Interaction priors . . . 86
4.3.3 Triplet priors . . . 88
4.4 Resulting model and discussions . . . 89
4.4.1 Model pipeline . . . 89
4.4.2 Discussions . . . 89
4.5 Conclusion . . . 92

5 Point Processes as Energy Based Models 93
5.1 Sampling . . . 95
5.1.1 Data driven kernels . . . 95
5.1.2 Parallel sampling . . . 99
5.1.3 Sampling method . . . 101
5.2 Parameters estimation . . . 102
5.2.1 Learning energy weights with local perturbations . . . 103
5.2.2 Maximum likelihood learning . . . 105
5.3 Papangelou intensity as a score . . . 112
5.3.1 Computing the detection score . . . 112
5.3.2 Contrastive divergence loss and Papangelou intensity . . . 113
5.4 Conclusion . . . 114

6 Experimental results **117**
6.1 Parameters estimation with Contrastive Divergence . . . 119
6.1.1 Training example: Filtering out irrelevant terms . . . 119
6.1.2 Stable perturbation kernel . . . 121
6.2 Results on optical data . . . 121
6.2.1 COWC with non-marked Point Process . . . 121
6.2.2 Point Process of rectangles on DOTA and ADS data . . . 124
6.3 Conclusion . . . 131

7 Conclusion and Perspectives **133**
7.1 Conclusion . . . 133
7.2 Perspectives . . . 134

Bibliography **137**

List of Figures **155**

List of Tables **157**

List of Definitions **159**

List of Algorithms **161**

Appendices **163**
A Jacobian for the local transform kernel . . . 163
B Training a CNN for small object detection in remote sensing data . . . 164
C Marks data energy derivation . . . 167
D Fast computation of overlap . . . 168
E Parallelism . . . 169
E.1 Minimum cell size . . . 169
E.2 Acceptance ratio for a move in a cell . . . 169
F Buffer: size and chain length . . . 172
G Towards generic (learnable) energy terms . . . 173
H Algorithmic complexity . . . 176
H.1 Cost per energy term . . . 177
H.2 Cost per kernel . . . 178
H.3 Total cost . . . 182
I Implementation: data structures and parallelism . . . 183
I.1 Batch computation of cells . . . 183
I.2 Generic energy models . . . 187
I.3 Miscellaneous methods . . . 189

My publications

National and international conferences

Mabon, J., Ortner, M., & Zerubia, J. (2021, September). Processus ponctuels et réseaux de neurones convolutifs pour la détection de véhicules dans des images de télédétection. In *ORASIS 2021 - 18èmes Journées francophones des jeunes chercheurs en vision par ordinateur.* Saint Ferréol, France: CNRS. Retrieved from https://hal.science/hal-03339656

Mabon, J., Ortner, M., & Zerubia, J. (2022a, August). CNN-Based Energy Learning for MPP Object Detection in Satellite Images. In *2022 IEEE 32nd International Workshop on Machine Learning for Signal Processing (MLSP)* (pp. 1–6). doi: 10.1109/MLSP55214.2022.9943312

Mabon, J., Ortner, M., & Zerubia, J. (2022b, September). Point process and CNN for small object detection in satellite images. In *SPIE, Image and Signal Processing for Remote Sensing XXVIII.* doi: 10.1117/12.2635848

Mabon, J., Ortner, M., & Zerubia, J. (2022c, September). Processus ponctuels marqués et réseaux de neurones convolutifs pour la détection d'objets dans des images de télédétection. In *GRETSI 2022 - XXVIIIème Colloque Francophone de Traitement du Signal et des Images.* Nancy, France. Retrieved from https://inria.hal.science/hal-03715337

Mabon, J., Ortner, M., & Zerubia, J. (2023a, August). Apprentissage contrastif de modèles de processus ponctuels pour la détection d'objets. In *GRETSI 2023 - XXIXème Colloque Francophone de Traitement du Signal et des Images.* Grenoble, France. Retrieved from https://inria.hal.science/hal-04177141

Mabon, J., Ortner, M., & Zerubia, J. (2023b, November). Learning point process models for vehicles detection using CNNs in satellite images. In *17th International Conference on Signal-Image Technology &Internet-Based Systems (SITIS).* Retrieved from https://inria.hal.science/hal-04250535

International journal

Mabon, J., Ortner, M., & Zerubia, J. (2023, November). *Learning Point Processes and Convolutional Neural Networks for object detection in satellite images.* Retrieved from https://inria.hal.science/hal-04250540 (to be submitted to IEEE TGRS)

Seminars and presentations

- Presentation at Centre Inria d'Université Côte d'Azur *PhD seminars*, October 2021.
- Presentation at *journées du RT Geosto-MIA*, Rouen, September 2022.
- Presentation to the *Airbus Defense and Space* teams, Toulouse, September 2022.
- Presentation to the *CNES data Campus* team visiting Centre Inria d'Université Côte d'Azur, September 2022.

Other activities

- Update and maintenance of the Ayana team website (2020-2023).
- Helping in the editing of the yearly Ayana team activity report (2020-2023).
- Organizing member (2021-2022) and secretary (2022-2023) of the Association Doctorale du campus STIC (ADSTIC).

Notations & Acronyms

Notations

Point Process

$\mathcal{S}$	Image space	Sec. 3.1.1
$\mathcal{M}$	Mark space	Def. 3.1.8
$\mathcal{S}_d$, $\mathcal{M}_d$	Discretized image and mark spaces	Eq. (5.6)
$\mathbf{y}$	Point configuration	Eq. (3.1)
$n(\mathbf{y})$	Number of points in $\mathbf{y}$	Def. 3.1.2
$\mathcal{Y}$, $\mathcal{Y}_n$	Set of all configurations with any number of points / n points	Eq. (3.2)
N^{lf}	Set of all locally finite configurations	Sec. 3.1.1
μ, ν	Density/intensity measure of the Poisson Point Process	Def. 3.1.4
f, h	Normalized / unnormalized density of the Point Process	Eq. (3.5), (3.8)
$\partial_B^A, \mathcal{N}_B^A$	Neighborhood of B in A for a relation $\sim$	Def. 3.1.5
$\lambda(y; \mathbf{y})$	Papangelou intensity	Def. 3.1.7

Sampling

π	Stationary measure of the Markov chain	Eq. (3.32)
K	transition kernel for the Markov chain	Def. 3.3.3
Q, Q_m	perturbation kernel / sub-kernel for the RJMCMC	Sec. 3.3.2
$\alpha(\mathbf{y}, \mathbf{y}')$, $r(\mathbf{y}, \mathbf{y}')$	Acceptance probability / Green ratio for move $\mathbf{y} \to \mathbf{y}'$	Sec. 3.3.2
T_t	Temperature at step t	Sec. 3.3.4

Energy model

$U(\mathbf{y}, X)$	Energy function defining density h	Sec. 3.2.1
V_e	Energy term	Eq. (4.3)
V	Energy of a point: sum of energy terms	Eq. (4.3)
v_e	Interaction potential	Eq. (4.35)
$\mathcal{F}_e$	Aggregation function for interaction potential v_e	Eq. (4.35)
ξ	Set of energy terms	Sec. 3.2.1

Generic

$\mathbf{A}[i, j, k]$, $\mathbf{A}[\rho]$	For a tensor $\mathbf{A}$, the value at coordinates (i, j, k) (interpolated if not integer values). By extension indexing with pixel ρ returns the value at location (ρ_i, ρ_j).

Acronyms

ADS	Airbus Defense and Space	Sec. 6.2.2
BCE	Binary Cross Entropy	Sec 4.2.2.2
CNN	Convolutional Neural Network	Def. 3.4.1
EBM	Energy Based Model	Def. 5.2.1
FCN	Fully Convolutional Network	Def. 3.4.3
GAN	Generative Adversarial Network	Sec. 2.3
GCD	Greatest Common Divisor	Sec. 3.3.1
GPU	Graphical Processing Unit	Sec. 5.1.2
GT	Ground Truth	Sec. 2.1.2
MAP	Maximum A Posteriori	Sec. 1.3.1
MLP	Multi Layer Perceptron	Sec. 4.4.2.2
MSE	Mean Square Error	Sec. 4.2.2.2
n.d.	No Date	Bibliography
NLL	Negative Log-Likelihood	Sec. 5.2.2
NLP	Natural Language Processing	Sec. 2.1.1
PP	Point Process	Def. 3.1.2
RGB	Red, Green, Blue	Sec. 1.2
(RJ)MCMC	(Reversible Jump) Monte Carlo Markov Chain	Sec. 3.3.2
SAR	Synthetic Aperture Radar	Sec 1.2
SOTA	State Of The Art	Sec. 1.5
SVM	Support Vector Machine	Sec. 2.1.1
TP/FP/FN	True Positive/False Positive/False Negative	Sec. 6.2.1.4
VAE	Variational Auto Encoder	Sec. 2.3

Résumé long en français

Introduction

La télédétection, dont les racines remontent jusqu'à la photographie en ballon à la fin du XIXe siècle, a considérablement évolué depuis, pour devenir essentielle pour la recherche, les applications commerciales ou la défense. Au fil des années, les capteurs sont devenus plus sophistiqués, cette tendance a été en partie motivée par l'escalade du renseignement militaire dans le contexte de la Seconde Guerre mondiale, puis de la guerre froide. Cela s'ajoute à un besoin accru de surveiller notre environnement à l'échelle mondiale, à mesure que les effets de l'humanité sur celui-ci deviennent plus préoccupants. Ce faisant, la quantité croissante de données produites par ces capteurs exige des solutions informatiques avancées pour le traitement. Les progrès de la télédétection correspondent à l'essor de la vision par ordinateur : depuis ses débuts dans les années 1950, en passant par le Perceptron en 1958 (Rosenblatt, 1958) et la transformée de Hough en 1972 (Duda & Hart, 1972) jusqu'à la résurgence des réseaux de neurones à la fin des années 1980 et l'essor des réseaux de neurones à convolution (CNN), enfin plus récemment l'adoption du *Deep Learning* avec AlexNet (Krizhevsky, Sutskever, & Hinton, 2012) ou RCNN (Girshick, Donahue, Darrell, & Malik, 2014) par exemple.

À première vue, la *télédétection* peut être comprise comme "la collecte d'informations à distance" (Campbell & Wynne, 2011). Cependant, cette définition trop large engloberait la microscopie jusqu'à l'enregistrement audio. Le domaine de la télédétection se concentre plutôt sur "l'observation des surfaces terrestres et aquatiques de la Terre au moyen de l'énergie électromagnétique réfléchie ou émise" (Campbell & Wynne, 2011). En pratique, les données considérées sont constituées d'images optiques (ou proches-optiques comme l'infrarouge), ainsi que d'images SAR (*Synthetic Aperture Radar*, ou Radar à Synthèse d'Ouverture, RSO, en français) ou Lidar (*Laser Imaging, Detection, and Ranging*) prises depuis des drones, des avions, des ballons ou des satellites.

Les images satellitaires contiennent de nombreux artefacts et occultations inhérentes à leur acquisition (voir (Tupin, Inglada, & Nicolas, 2014)), ce qui les distingue des images *usuelles* ; c'est-à-dire des photographies que nous rencontrons quotidiennement, prises à la hauteur d'œil, généralement avec un sujet proche du centre, où les objets d'intérêt représentent une proportion importante de l'image. Compte tenu de la faible résolution spatiale des images satellitaires (autour de $0,5$ m par pixel), les objets d'intérêt ne mesurent parfois que quelques pixels. Les informations visuelles étant limitées, les opérateurs humains ont souvent recours à leur connaissance a priori de l'objet d'intérêt lors de l'analyse d'une image. De la même façon, nous visons à construire des modèles qui prennent en compte cet a priori.

Le but de cette thèse est d'étudier la combinaison d'approches modernes de réseaux de neurones convolutifs (CNN) avec des modèles géométriques utilisant des a priori ou des contraintes. Bien que les modèles CNN se soient révélés très efficaces pour extraire des informations sur la texture, ils ne parviennent pas à prendre en compte les interactions à plus longue portée entre les objets sans augmenter considérablement la complexité et le nombre de paramètres. D'autre part, les modèles de Processus Ponctuels (PP) proposent de modéliser des configurations d'objets géo-

métriques comme un processus stochastique : premièrement, la construction du Processus Ponctuel elle-même définit le type de géométries extraites. De plus, le modèle stochastique contient des a priori qui peuvent guider le modèle vers des configurations plus probables a priori.

Modèle d'observation de l'image. L'approche par Processus Ponctuel dérive de modèles stochastiques d'analyse d'images. En considérant l'image comme un ensemble de pixels dans $\mathcal{S}_d \subset \mathbb{Z}^2$, nous pouvons la modéliser comme une variable aléatoire X dans l'espace de probabilité $(\Omega, \mathcal{A}, \mathbf{P})$, où les valeurs en niveaux de gris correspondent aux réalisations de la fonction X :

$$X : \Omega \to \mathbb{R}^{\mathcal{S}_d}. \tag{0.0.1}$$

Si l'on considère l'image comme une observation bruitée d'un phénomène sous-jacent Φ, on peut utiliser le modèle bayésien suivant :

$$\mathbf{P}(X = \mathbf{X}) = \sum_{\mathbf{y}} \mathbf{P}(X = \mathbf{X}|\Phi = \mathbf{y})\mathbf{P}(\Phi = \mathbf{y}), \tag{0.0.2}$$

où l'image $\mathbf{X}$ et la configuration $\mathbf{y}$ sont respectivement les réalisations de X et Φ. Ici $\mathbf{P}(X = \mathbf{X})$ correspond à la loi marginale des observations, construite à partir de la loi des observations $\mathbf{P}(X = \mathbf{X}|\Phi = \mathbf{y})$ et de la loi a priori $\mathbf{P}(\Phi = \mathbf{y})$ sur le processus sous-jacent. Avec la formule de Bayes, nous pouvons inverser le conditionnement pour étudier la loi de Φ conditionnellement à l'observation $\mathbf{X}$:

$$\mathbf{P}(\Phi = \mathbf{y}|X = \mathbf{X}) \propto \mathbf{P}(X = \mathbf{X}|\Phi = \mathbf{y})P(\Phi = \mathbf{y}). \tag{0.0.3}$$

Pour minimiser le coût bayésien $L(\Phi^*, \Phi) = \mathbb{1}(\Phi \neq \Phi^*)$, l'estimateur optimal est le Maximum A Posteriori (MAP) :

$$\mathbf{y}^* = \arg\max_{\mathbf{y}} \mathbf{P}(\Phi = \mathbf{y}|X = \mathbf{X}). \tag{0.0.4}$$

Processus Ponctuels. Les Processus Ponctuels permettent de modéliser de tels phénomènes Φ, comme un ensemble d'objets paramétriques représentant des objets géométriques tels que des segments, des cercles, des rectangles, des polygones, etc. Nous revenons plus en détail sur les Processus Ponctuels dans le chapitre 3. Ici, nous considérons les configurations $\mathbf{y}$ comme un ensemble $\{y_1, \ldots, y_{n(\mathbf{y})}\}$ de $n(\mathbf{y})$éléments y qui représentent nos objets d'intérêt. Dans le cadre bayésien, un modèle d'observation peut être construit comme :

$$\mathbf{P}(X_\rho = \mathbf{X}_\rho|Y = \mathbf{y}) = \mathbb{1}(\rho \in \mathbf{y})\mathbf{P}(X_\rho = \mathbf{X}_\rho|\rho \in \mathbf{y}) + \mathbb{1}(\rho \notin \mathbf{y})\mathbf{P}(X_\rho = \mathbf{X}_\rho|\rho \notin \mathbf{y}). \tag{0.0.5}$$

Dans cette approche, on applique un modèle d'objet ($\rho \in \mathbf{y}$) ou d'arrière-plan ($\rho \notin \mathbf{y}$) si un pixel ρ appartient à la silhouette de $\mathbf{y}$ ou pas. Un modèle courant si la silhouette correspond à des pixels clairs sur un fond plus sombre consiste à modéliser le premier plan $\mathbf{P}(X_\rho = \mathbf{X}_\rho|\rho \in \mathbf{y})$ comme une loi gaussienne de grande valeur. L'arrière-plan $\mathbf{P}(X_\rho = \mathbf{X}_\rho|\rho \notin \mathbf{y})$ se voit ensuite assigné une valeur inférieure (Baddeley & Lieshout, 1993; Perrin, Descombes, & Zerubia, 2004).

Cependant, cette approche bayésienne présente certaines limites (Ben Hadj, Chatelain, Descombes, & Zerubia, 2010) :

- Les arrière-plans non homogènes sont plus difficiles, voire impossibles à modéliser. Le modèle gaussien simple pour $\mathbf{P}(X_\rho = \mathbf{X}_\rho|\rho \notin \mathbf{y})$ n'est plus utilisable.

- La vraisemblance des pixels de la classe de premier plan ne dépend pas des propriétés morphologiques de la classe à extraire. Par exemple, le modèle peut essayer d'insérer autant de petits objets que possible dans une grande silhouette (Craciun, Ortner, & Zerubia, 2015).

Pour ces raisons, (Ben Hadj et al., 2010) propose un modèle *détecteur* fondé sur une mesure de contraste entre l'intérieur et l'extérieur de l'objet. Néanmoins, ces mesures de contraste reposent sur le contraste élevé de l'objet par rapport à son arrière-plan. Cette approche atteint ses limites lorsque le fond présente des éléments contrastés qui ne devraient pas être détectés. Nous montrons un exemple pratique de ces limites dans la partie 4.2.1.1.

Réseaux de Neurones Convolutifs. Alors que le modèle d'image ci-dessus était motivé en partie par les limitations de calcul, la disponibilité croissante de la puissance de calcul a facilité l'essor des réseaux de neurones convolutifs (CNN, *Convolutional Neural Network*). Bien qu'introduit pour la classification d'images en 1989 (LeCun et al., 1989), ce n'est que dans les années 2010 que CNN a été largement adopté avec AlexNet (Krizhevsky et al., 2012) pour la classification ou RCNN (Girshick et al., 2014) pour la détection d'objets.

Les réseaux de neurones convolutifs traitent les données d'une topologie de type grille via une séquence d'opérations de convolution et de regroupement (agrégation spatiale). Les modèles convolutifs profonds apprennent les filtres de convolution pour une tâche spécifique à partir de grandes bases de données d'images telles que ImageNet (Krizhevsky et al., 2012) par exemple. Un modèle de segmentation tel qu'Unet (Ronneberger, Fischer, & Brox, 2015) effectue une classification par pixel. Ceux-ci produisent des cartes à valeurs réelles dans la plage $[0, 1]$, qui sont interprétées comme des probabilités de classe. Par exemple, la probabilité que le pixel ρ appartienne à la classe C_1 (parmi les classes N_C) est donnée par :

$$\widehat{P}(\rho \in C_1|\mathbf{X}) = \frac{\exp\left(F(\mathbf{X})[\rho, C_1]\right)}{\sum_{k=1}^{N_C} \exp\left(F(\mathbf{X})[\rho, C_k]\right)}, \tag{0.0.6}$$

avec $F(\mathbf{X})$ est le tenseur de sortie du CNN appartenant à $\mathbb{R}^{\mathcal{S}_d \times [\![1, N_C]\!]}$, et N_C le nombre de classes (2 dans le cas de la classification objet ou arrière-plan). Ce score $\widehat{P}(\rho \in C_1|\mathbf{X})$ mesure la présence ou non d'un objet, avec un modèle convolutif "léger" (faible nombre de paramètres) dont les noyaux sont appris à partir des données.

Contributions

Le point de départ de notre travail — inspiré de (T. Li, Comer, & Zerubia, 2019) — est d'exploiter la partition $\widehat{P}(\rho \in C_1|\mathbf{X})$ et de l'utiliser pour construire l'énergie externe du Processus Ponctuel (attache aux données), remplaçant la mesure de contraste. Dans cette thèse de doctorat, nous proposons l'incorporation de modèles d'interaction dans les méthodes de détection d'objets, tout en tirant parti des capacités des réseaux de neurones convolutifs profonds. Les images satellitaires sont intrinsèquement imparfaites en raison des perturbations atmosphériques, des occultations partielles et de la résolution spatiale limitée. Pour compenser ce manque d'informations visuelles, il devient essentiel d'incorporer des connaissances préalables sur la disposition des objets d'intérêt.

D'une part, les méthodes basées sur des CNN sont excellentes pour extraire des motifs dans les images, mais ont du mal à apprendre des modèles d'interaction objet à objet sans avoir à y introduire des mécanismes d'attention tels que les *Transformers* qui augmentent considérablement

la complexité du modèle : par exemple, certaines approches proposent d'inclure les a priori sous forme de texte descriptif des objets et leurs relations (Lu et al., 2023), tandis que d'autres utilisent des cascades de modules d'attention (Zeng et al., 2023).

D'autre part, les Processus Ponctuels proposent de résoudre conjointement les vraisemblances relatives à l'image (terme d'attache aux données), et la cohérence de la configuration d'objets elle-même (terme a priori). Premièrement, les approches Processus Ponctuel modélisent les configurations sous forme de géométrie vectorielle, contraignant l'espace d'état par construction. De plus, ces modèles permettent de formuler des a priori explicites sur les configurations en tant que fonctions énergétiques. Cependant, dans la plupart de la littérature (Verdié & Lafarge, 2014; Schmidt, Lafarge, Brenner, Rottensteiner, & Heipke, 2017), les termes d'attache aux données reposent sur des mesures de contraste entre les objets d'intérêt et l'arrière-plan. Nous montrons dans cette thèse les limites de ces mesures sur les données satellitaires.

Au lieu d'augmenter la complexité du modèle en y ajoutant par exemple des *Transformers*, nous proposons dans cette thèse de combiner l'extraction de motifs CNN avec l'approche Processus Ponctuel. Le fondement de cette approche est d'utiliser la sortie d'un CNN comme terme de données pour un modèle de Processus Ponctuel.

Construction du modèle d'énergie. Nous construisons le Processus Ponctuel de densité h à travers sa fonction d'énergie U, pour une configuration $\mathbf{y}$ et une image $\mathbf{X}$:

$$h(\mathbf{y}) \propto \exp(-U(\mathbf{y}, X)) \tag{0.0.7}$$

Nous écrivons la fonction d'énergie U comme :

$$U(\mathbf{y}, X) = \sum_{y \in \mathbf{y}} V\left(y, \mathbf{X}, \mathcal{N}^{\mathbf{y}}_{\{y\}}\right), \tag{0.0.8}$$

où $V(y, \mathbf{X}, \mathcal{N}^{\mathbf{y}}_{\{y\}})$ est l'énergie du point y, sachant l'image $\mathbf{X}$ et le voisinage $\mathcal{N}^{\mathbf{y}}_{\{y\}}$ de y dans $\mathbf{y}$. Cette énergie est une combinaison linéaire de termes d'attache aux données et de terme a priori.

Dans le Chapitre 4, nous montrons d'abord les limites des mesures basées sur le contraste utilisées comme attache aux données pour les images satellitaires, puis procédons à la construction d'un terme d'attache aux données à partir de la sortie d'un CNN.

Nous proposons plusieurs manières de construire une carte de potentiel :

- Dans un premier temps, nous proposons d'utiliser une méthode de type contraste (correspondance de motifs utilisant une mesure de corrélation) sur la sortie d'un CNN entraîné pour transformer les centres des objets en blobs dans sa sortie (Mabon, Ortner, & Zerubia, 2021).

- Nous proposons ensuite une nouvelle méthode utilisant un CNN entraîné pour inférer un champ de vecteurs pointant vers le centre de l'objet le plus proche. Pour obtenir une carte de probabilité des centres, il suffit de calculer la divergence de ce champ. Cela permet de séparer facilement des objets proches pour un faible coût de calcul (Mabon, Ortner, & Zerubia, 2022b).

- Enfin, nous montrons comment réinterpréter toute sortie CNN de type *heatmap* (voir partie 2.1.1 ou (Law & Deng, 2018)) comme une énergie pouvant être utilisée pour notre modèle, à la fois pour la position et les marques de chaque objet (Mabon, Ortner, & Zerubia, 2023a).

Avec notre formulation de l'énergie d'attache aux données sous forme d'interpolation d'une carte de potentiel, nous transformons le calcul de la mesure de contraste en une simple extraction et interpolation de valeurs dans un tenseur 2D ou 3D.

Échantillonnage. Avec ces cartes de potentiel pré-calculées, nous procédons dans la partie 5.1.1 à l'adaptation des méthodes d'échantillonnage des Processus Ponctuels pour utiliser ces informations facilement disponibles. En effet, ces tenseurs définissent des densités sur un espace discret facile à échantillonner et qui se rapproche d'une version tronquée du modèle énergétique (qui est défini dans un espace continu). Tout d'abord, nous appliquons cet échantillonnage guidé par les données au noyau de perturbation de transformation locale (Mabon et al., 2021), puis construisons une carte de naissance pour le noyau de naissance et mort (Mabon et al., 2023a). Ces densités spatiales nous permettent également de concentrer l'échantillonnage parallèle du Processus Ponctuel sur les zones où le nombre de points est susceptible d'être plus élevé et nécessite donc plus de propositions. Enfin, nous exploitons les moteurs de différenciation automatique facilement disponibles pour calculer le gradient d'énergie et effectuer une diffusion pour une exploration plus efficace de l'espace d'état pour un nombre de points fixé.

Estimation des paramètres. Les méthodes précédentes utilisent la Programmation Linéaire pour estimer les poids relatifs des énergies du modèle (Q. Yu & Medioni, 2009; Craciun et al., 2015) à partir d'un ensemble de contraintes. Cependant, cela est sujet à des sur-contraintes et ne prend en compte qu'un nombre limité de points de données. Notre première approche d'estimation des paramètres consiste à proposer une méthode utilisant un SVM, qui recherche la meilleure séparation possible entre les échantillons positifs et négatifs. Grâce à sa frontière perméable, cette méthode permet de considérer beaucoup plus de points de données, même ceux bruités qui contrediraient des contraintes strictes (Mabon et al., 2021). Ensuite, nous proposons d'adapter une méthode utilisée pour les modèles basés sur l'énergie (EBM) (Mabon, Ortner, & Zerubia, 2022a) : la divergence contrastive (Hinton, 2002; Du & Mordatch, 2019). Dans un schéma de descente de gradient, nous alternons entre la génération d'échantillons contrastifs/négatifs à partir de l'énergie actuelle et la maximisation de la différence d'énergie entre les échantillons positifs (similaires à la vérité terrain) et les échantillons contrastifs. Cette procédure permet d'estimer non seulement les pondérations énergétiques, mais également les différents paramètres internes des termes énergétiques, qui autrement auraient été définis manuellement (ou avec une procédure distincte pour chacun). Ces deux approches sont détaillées dans la partie 5.2.

Intensité de Papangelou comme score de détection. Pour pouvoir comparer notre modèle avec d'autres modèles de détection d'objets utilisant des CNN, nous introduisons une fonction de notation pour les objets déduits par le modèle de Processus Ponctuel dans la partie 5.3. Ce score est calculé à partir de l'intensité conditionnelle de Papangelou, qui mesure la probabilité d'obtenir un point compte tenu du reste de la configuration. Il s'agit d'une différence essentielle par rapport aux fonctions de score classiques qui ne prennent en compte les points qu'individuellement, quel que soit leur voisinage.

Application. Dans le Chapitre 6, nous appliquons nos méthodes sur des ensembles de données de télédétection ; à la fois sur les benchmarks COWC (Mundhenk, Konjevod, Sakla, & Boakye, 2016) et DOTA (Xia et al., 2018) qui sont accessibles au public, ainsi que sur certaines données

fournies par Airbus Defence and Space (ADS). Même avec la complexité limitée de nos modèles, nous obtenons de bons résultats qualitativement en assurant la régularité de la géométrie déduite. Quantitativement, nos modèles sont meilleurs sur ces données que les autres méthodes CNN auxquelles nous nous comparons. Cette différence de performance est plus prononcé avec des données d'entrée bruitées : nos modèles fournissent de bons résultats qualitatifs et quantitatifs même dans des conditions de bruit défavorables sur l'image d'entrée. Nous démontrons également que la méthode d'estimation des paramètres proposée correspond ou surpasse légèrement celle du réglage manuel des paramètres du modèle, qualitativement et quantitativement, avec l'avantage intrinsèque de limiter les fastidieux essais et erreurs manuels. Enfin, nous montrons que notre modèle peut se généraliser à de nouvelles données en examinant les résultats qualitatifs sur les données ADS après un entrainement du modèle sur DOTA.

Limitations. Cependant, comme notre implémentation n'est que légèrement optimisée (principalement pour permettre la modularité lors de l'expérimentation) et en raison de la nature itérative des méthodes d'échantillonnage par chaines de Markov, le temps d'inférence est significativement plus long que celui des méthodes de détection d'objets purement CNN. Une accélération pourrait être obtenue en approchant la méthode d'échantillonnage : par exemple en utilisant moins d'itérations, avec une inférence approximative comme état initial (par exemple la sortie *CNN-LocalMax.*, voir la partie 6.2.2) et un recuit simulé plus rapide. Cependant, cela s'éloignerait davantage de la configuration d'échantillonnage optimale, mais donnerait des résultats plus rapidement. D'autres solutions existent telles que la discrétisation de l'espace d'état, puis la résolution d'un problème d'optimisation de variables binaires (T. T. Pham, Hamid Rezatofighi, Reid, & Chin, 2016).

Perspectives. Alors que, dans cette thèse, nous nous sommes concentrés sur la détection de petits véhicules, le cadre que nous proposons peut être facilement adapté à toute autre modalité d'objets où les interactions inter-objet sont clés. Par exemple, les réseaux routiers peuvent être modélisés avec des Processus Ponctuels de segments comme dans (Lacoste, Descombes, & Zerubia, 2005), et avoir des a priori importants sur leurs interactions (par exemple un rayon de courbure limité, physiquement et légalement). Les Processus Ponctuels aussi peuvent être appliqués aux données temporelles ; (Craciun et al., 2015) effectue un suivi des données de télédétection. Dans ce cas, les a priori sur la dynamique sont forts et pourraient être capables de compléter le manque d'information visuelle et même d'occlusions complètes (par exemple une voiture passant sous un pont). De plus, comme notre modèle est assez résistant au bruit, nous nous attendons à ce qu'il fonctionne bien sur les données SAR, car le bruit d'entrée (bruit de chatoiement) est plus important avec ce type de données.

Les travaux futurs pourraient se concentrer sur les capacités de composition de tels modèles énergétiques : en bref, il est possible d'estimer les fonctions énergétiques d'attache aux données sur un premier ensemble de données, et le modèle énergétique a priori sur un autre, pour ensuite composer les deux fonctions d'énergie et former un modèle complet (attache aux données et a priori). En pratique, cela permettrait d'apprendre un modèle de données sur des annotations imparfaites, et d'apprendre un modèle a priori sur des données synthétiques *parfaites* qui reproduisent les structures attendues ; pour ces données synthétiques nous n'aurions besoin que de configurations d'objets, pas d'images.

Enfin, la méthode d'estimation des paramètres proposée n'est pas nécessairement liée à la détection d'objets. Comme nous avons noté précédemment la nature générative de notre modèle

de Processus Ponctuel, nous pouvons donc l'appliquer à l'apprentissage de structures de points afin de produire un modèle génératif capable d'imiter ces structures, comme cela est fait dans (Hurtut et al., 2009) par exemple. Ce type d'approche pourrait bénéficier d'un modèle a priori non spécifique à l'application qui serait appris sur les données, comme nous le proposons en annexe G : utiliser un approximateur universel pour l'a priori par objet et apprendre les opérateurs d'agrégation sur les interactions en utilisant les mécanismes d'attention.

PART

Introduction

CHAPTER 1

Introduction

Remote sensing, a technology with roots dating back to photography from balloons in the late 19th century (Figure 1.1), has evolved significantly since, to become key for research, commercial use or defense. Over the years, sensors got more sophisticated, in part sparked by the escalation of military intelligence in the context of World War II and subsequently the Cold War. This coupled with an increased need to monitor our environment globally as humanity's effects on it became more concerning. In turns, the growing amount of data produced by those sensors demanded advanced computing solutions for processing, analysis and interpretation. The advancements in remote sensing matches the rise of computer vision: from the early foundation of computer vision in the 1950s, through the Perceptron in 1958 (Rosenblatt, 1958) and the Hough transform in 1972 (Duda & Hart, 1972) to the resurgence of neural networks in the late 1980s and the rise of Convolution Neural Networks (CNN), and most recently the adoption of Deep Learning with AlexNet (Krizhevsky et al., 2012) or RCNN (Girshick et al., 2014) for instance.

Figure 1.1: *Nadar Élevant la Photographie à la Hauteur de l'Art* (Honoré Daumier, 1862).

At first glance, *remote sensing* can be understood as "the gathering of information at a distance" (Campbell & Wynne, 2011). However, this too broad definition would encompass from microscopy to audio recording. The field of remote sensing, rather focuses on the "observation of the Earth's land and water surfaces by means of reflected or emitted electromagnetic energy" (Campbell & Wynne, 2011). In practice, the considered data consists of optical (or near-optical such as infrared) imagery, as well as SAR (Synthetic Aperture Radar) or Lidar (Laser Imaging, Detection, and Ranging) taken from drones, planes, balloons or satellites; in short "from an overhead perspective" (Campbell & Wynne, 2011).

In this thesis our main goal is to detect small objects, such as vehicles, in satellite images. With resolutions around 0.5m per pixel and the perturbations inherent to the image acquisition process, we try to complement the lack of information in the image with priors on the extracted geometries. We aim at complementing modern CNN approaches with interaction models from Point Processes to regularize the extracted configurations of objects.

This introductory chapter, first presents some practical applications of remote sensing and applications of object detection. We then present the key elements of satellite image acquisition

with its challenges and constraints. Finally, we broadly review approaches for object detection to finally conclude on the thesis structure.

1.1 Computer vision in remote sensing

Thanks to constant technological advances and the rise of open data, the uses for computer vision in remote sensing are growing. As a motivation, we present a few examples of practical applications.

Land use. Satellite images provide a live view of a territory and helps keeping cartographic resources up to date (IGN, 2020). This allows for instance to monitor urban sprawl at a large scale (*Une carte d'occupation du sol pour faciliter le suivi de l'artificialisation - Institut - IGN*, 2023), which is of great importance as it is a key cause for loss of biodiversity (McKinney, 2002), pollution and heath issues (Jones & Kammen, 2014). Combining object detection in satellite images with up-to-date land registers can have uses for public finances; recently, the French tax office has been experimenting with detection of undeclared swimming pools from satellite imagery ("La détection par intelligence artificielle de piscines non déclarées va être généralisée en France", 2022).

Disaster monitoring. A live view of large swathes of land can prove crucial in monitoring disasters, both for responding to humanitarian crises or in the long run. Satellites can help monitor the ongoing climate crisis, from measuring the rising sea surface temperature (*European Sea Surface Temperature*, 2023) to producing a live-view of forest fires across the world ("Cartes des incendies dans le monde : suivez en temps réel les feux de forêt observés sur Terre", 2023), (see this monitoring tool from NASA with only 3 hours delay (Earth Science Data Systems, 2015)). Optical satellite imagery is also in use for disaster response: for instance, in September 2023, in response to the earthquake in Morocco, the available satellites of 8 space agencies where programmed to produce pictures of the affected area (Bronner, 2023). This data allows identifying and measuring the extent of the damage, and helps to coordinate the relief efforts.

Military intelligence. As mentioned previously, remote sensing technologies are deeply rooted in military intelligence. From observation balloons in 1794 during French Revolutionary wars, to World War I with the first plane photographic reconnaissance missions, and the development of spy satellite during the cold-war: monitoring — preferably from an overhead perspective — the presence, amount and nature of military assets is key to any conflict or defense strategy. In recent years, the wide-spread availability of satellite image resources has lead to the rise of Open Source Intelligence (OSINT) (*OSINT : aux sources d'un nouveau journalisme ?*, 2023), which became a significant source of public information during the recent conflict in Ukraine.

Domestic intelligence. However, it is important to note the global increase in use of automated acquisition and detection systems aimed towards domestic surveillance; from the NSA global communication eavesdropping programs unveiled in 2013 (Gellman & Poitras, 2013), to the ongoing growth of the Chinese surveillance state (facial recognition, phone trackers, voice recognition etc.) (Xiao, Mozur, Qian, & Cardia, 2022), or France's recent legislation on the usage of police drone during protests, or the experimentation on automatic facial recognition opening the way for the

trivialization of such practices (Verdon & Nabat, 2023). Although non-military satellite technology is not capable enough yet for these practices, the use of drone systems coupled with automated recognition algorithms can raise ethical concerns.

1.2 Satellite image acquisition

To better understand the specificities and constraints of detection in satellite images, we present in this section a few key elements of the image acquisition process. Most of the information in this section have been sourced from (Tupin et al., 2014), to which we refer the reader for a more thorough explanation.

Orbits and cycles. Satellites for civilian imagery, usually orbit from 450 to 910 km around the Earth. The orbital plane is often inclined with respect to the equatorial plane in order to cover as much ground as possible. For instance a *polar orbit* passes above the North and South poles, describing an orbit orthogonal to the equatorial plane. Lower orbits require less fuel to launch and make for higher resolution images, however as satellites get lower in orbit the effects of atmospheric drag get higher. Even if satellites are loaded with some fuel for trajectory corrections, low orbits demand more corrections because of the drag. Thus, the lifespan of low orbit satellites is often limited.

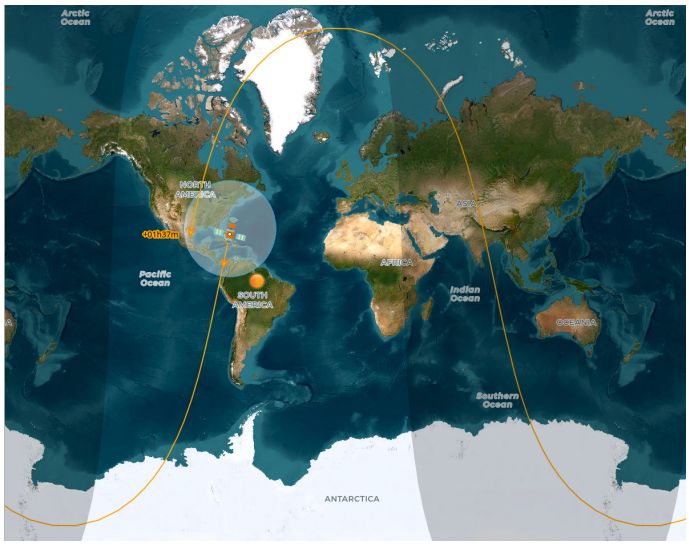


Figure 1.2: Ground track of the *Pleiades NEO-4* satellite orbited in August 2021. *From (*OrbTrack.Org*, 2023).*

Due to the combination of the satellite's orbit and the Earth's rotation (eastward with North up), a position on the orbit does correspond to a single position on the ground; at every orbit the position of the satellite gets shifted to the west (see Figure 1.2). The time a satellite takes to come back to the same orbit position and location relative to the ground is called the *cycle* of the orbit (e.g. 1 day for Formosat-2, 26 days for SPOT or Pleiades NEO). This cycle time will determine the minimal time difference between two acquisitions of the same area; which plays a key role in

change detection. This can be mitigated by having *constellations* of satellites, better covering the globe at all times.

Passive sensors: Optical. There exists two types of sensors: *passive* and *active*. The passive sensors measure the back scattering of sunlight (or artificial lights) on the surface. For thermal sensors it can measure the ground radiation. The first sensors where film stock. In the 1960s US satellite platforms such as GAMBIT-1 or CORONA would drop a reentry canister back into the atmosphere to be recovered midair by plane or on the ground (*60th Anniversary of the First GAMBIT-1 Photoreconnaissance Satellite Flight*, 2023). Modern satellites use a linear array of Charged Coupled Device (CDD) that scan an area as the satellites moves over it (as a document scanner does), this is referred to as *pushbroom* (see Figure 1.3). 2D CCD arrays (that we find in everyday digital cameras) are marginal in use up to now, although the upcoming CO3D constellation will embark two 2D CCD sensors, one for RGB bands, the other for NIR (*CO3D Constellation*, 2023).

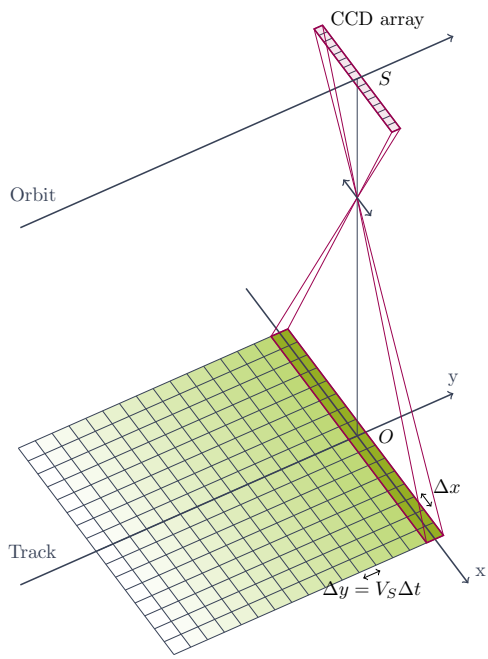


Figure 1.3: Optical pushbroom: the CCD linear detector acquires at a period Δt. The interval between two rows of pixels Δy is given by the time interval Δt multiplied the satellite ground speed V_S. *Adapted from (Tupin et al., 2014).*

The *footprint* of a satellite and sensor describes the intersection of the perspective cone and ground surface, in which two objects cannot be resolved. The footprint is dependent on the optical system and sensor, as well as incidence (angle between ground normal and sensor), altitude and ground inclination. For a horizontal ground, the lowest footprint is at low altitude and incidence; i.e. the optical satellite best acquires images at *Nadir* (satellite pointing towards the center of the Earth). *Off-Nadir* images have less spatial resolution, and are more prone to occlusion due to mountains, trees or tall buildings. Note that for a satellite at Nadir, as we move further from the optical axis (vector $\overrightarrow{SO}$ in Figure 1.3), the light paths get progressively off Nadir.

Active sensors: Synthetic Aperture Radar (SAR). Synthetic Aperture Radar (SAR) satellites are active sensors; they emit electromagnetic wave with an antenna, that gets reflected on the surface and measured back by the antenna. The system measures the signal return time, thus the SAR footprint depends on the temporal resolution and the incidence. Here lower incidence increases footprint; i.e. SAR satellites are preferably pointing at the ground with an angle. The movement of the satellite relative to the ground introduces some more subtleties to this system; we will not go into further details about of SAR acquisition in this thesis, we rather refer the reader to (Tupin et al., 2014) or (Campbell & Wynne, 2011) for more detailed information.

Spectral resolution. Optical sensor measures the quantity of light within a given range of wavelengths. For instance *panchromatic* images are produced by capturing the visible spectrum from 400 to 700 nm, producing a grayscale image corresponding to the integrated intensity over the bandwidth, exposure time and sensor area. The bandwidth can be split into bands, to produce *multispectral* images. The most common we encounter every day is splitting into Red, Green and Blue bands (RGB) to be able to reproduce color images for the human eye. Satellites can have from ten to hundreds of bands. The fine spectral resolution allows discerning different materials and surfaces given their reflectance properties. Multispectral sensors often include bands beyond the visible spectrum such as near infrared (NIR), which is strongly reflected by vegetation. For instance Pleiades NEO satellites (PNEO-3 and PNEO-4) have 7 spectral bands: panchromatic ($450 - 800$ nm), RGB ($450 - 520$, $530 - 590$, and $620 - 690$ nm), NIR ($770 - 880$ nm), deep blue ($400 - 450$ nm, used for instance for oceanography) and red edge ($700 - 750$ nm, used for vegetation) (*Pléiades Neo*, n.d.). As bands get finer, the flux of light gets lower, thus the signal-to-noise ratio worsens. As a compromise narrowband systems often increase sensor surface, at the cost of increased footprint (i.e. lower spatial resolution). For instance the panchromatic band in Pleiades NEO has a resolution of 30 cm while the other narrower bands have a resolution of 1.2 m.

Atmospheric Perturbations. "It is necessary to talk about the atmosphere, since it is situated between the satellite and the surface" (Tupin et al., 2014). The most glaring hindrance to optical imaging is clouds; those hide the surface and project shadows. Their non-solid edges make them difficult to isolate and correct for. However, clear skies also interact with the light the satellite is capturing. The atmosphere both absorbs and scatters the light depending on its chemical composition and density. Both do not have uniform effect across the light spectrum. We shall also mention, surface effects, such as specular reflections on mirror like surfaces, which can cause lens flare artifacts.

Image transmission. Nowadays, satellites beam back the acquisitions by radio; both the bit rate and transmission window are limited as the satellite is not always in range of the ground antenna. The size of the transmitted signal depends on the number of bands, pixels and *radiometric* resolution (the number of quantized values for the intensity of one pixel in one band). Usually, the ground to satellite link has a low transmission rate as it only transmits commands to the satellite. Meanwhile, satellite to ground links have higher throughput to accommodate for the large amounts of data to send back. Some Earth observation satellites (such as Sentinel) can beam back data trough higher orbit relay satellites, such as the *European Data Relay Satellites*, EDRS-A and C (*European Data Relay Satellite System (EDRS) Overview*, n.d.). The transmission rate is still limited due to the distances (loss of signal energy) involved with geostationary satellites.

Image processing. Once on ground, the image can be processed. Modeling the atmospheric interactions allows correcting for part of the above-mentioned absorptions. The images are bespangled to conform to an *orthoimage*; i.e. transformed image that corresponds to a cartographic geometry. To compensate for the limited resolution of fine spectral bands, one can combine the high spatial resolution of panchromatic images with the fine spectral resolution of multispectral. This process is known as *pansharpening*.

From the multispectral output, *spectral indices* are built. Those combine the information of multiple bands into a single index that corresponds to a specific element of interest. For instance NDVI combines red (R_R) and near infrared (R_{NIR}) bands. It is a great tool to quickly identify vegetated areas without any complicated image processing:

$$NDVI = \frac{(R_R - R_{NIR})}{(R_R + R_{NIR})} \tag{1.1}$$

1.3 Object detection

As shown previously, satellite images contain many artifacts and occlusions compared to *usual* images; i.e. photographs we encounter every day, taken at human eye-level, usually with a close centered subject, where objects of interest represent a significant proportion of the image. Given the low spatial resolution of satellite images, the objects of interest may be only a few pixels large. As the visual information is limited, human operators often resort to their prior knowledge of the object of interest when analyzing an image. Similarly, we aim to build models that factor in this prior.

The goal of this thesis is to study the combination of modern Convolutional Neural Network (CNN) approaches with geometrical models using priors or constraints. While CNN models have shown to be really efficient at extracting texture information, they fail to encompass longer range interactions between objects without drastically increasing complexity and parameters count. On the other hand, Point Process (PP) models propose to model configurations of geometrical objects as a stochastic process. First, the Point Process construction itself defines the type of geometries that are extracted. Also, the stochastic model contains priors that can guide the model towards configurations that are more likely *a priori*.

1.3.1 Image probability model

The Point Process approach derives from stochastic models of image analysis. Considering the image as a set of pixels in $\mathcal{S}_d \subset \mathbb{Z}^2$, we can model it as a random variable X in probability space $(\Omega, \mathcal{A}, \mathbf{P})$, where grayscale values correspond to the realizations of function X:

$$X : \Omega \to \mathbb{R}^{\mathcal{S}_d}. \tag{1.2}$$

If we consider the image as a noisy observation of underlying phenomenon Φ, a Bayesian model for the image is then:

$$\mathbf{P}(X = \mathbf{X}) = \sum_{\mathbf{y}} \mathbf{P}(X = \mathbf{X} | \Phi = \mathbf{y}) \mathbf{P}(\Phi = \mathbf{y}), \tag{1.3}$$

where image $\mathbf{X}$ and configuration $\mathbf{y}$ are respective realizations of X and Φ. Here $\mathbf{P}(X = \mathbf{X})$ corresponds to the marginal law of observations, built from the law of observations $\mathbf{P}(X = \mathbf{X} | \Phi =$

$\mathbf{y}$) and prior law $\mathbf{P}(\Phi = \mathbf{y})$ on the underlying process. With Bayes formula we can revert the conditioning to study the law of Φ conditional to the observation $\mathbf{X}$:

$$\mathbf{P}(\Phi = \mathbf{y}|X = \mathbf{X}) \propto \mathbf{P}(X = \mathbf{X}|\Phi = \mathbf{y})P(\Phi = \mathbf{y}). \tag{1.4}$$

The Bayes model provides a class of natural estimators: Bayes estimators. An estimator of Φ is a function $\Phi^*(\mathbf{X})$ of observation $\mathbf{X}$, which we want to best approach the underlying values of Φ that generated image $\mathbf{X}$. This relies on Bayes cost:

$$R_B(\Phi^*) = \mathbb{E}_\Phi \left[\mathbb{E}_{X|\Phi}\left[L(\Phi^*, \Phi)\right]\right]. \tag{1.5}$$

For cost $L(\Phi^*, \Phi) = \mathbb{1}(\Phi \neq \Phi^*)$, the optimal estimator is the Maximum A Posteriori (MAP) estimator:

$$\mathbf{y}^* = \arg\max_{\mathbf{y}} \mathbf{P}(\Phi = \mathbf{y}|X = \mathbf{X}). \tag{1.6}$$

If we consider all pixels to be independent of each other we get:

$$\mathbf{y}^* = \arg\max_{\mathbf{y}} \prod_{\rho \in \mathcal{S}_d} \mathbf{P}(\Phi = \mathbf{y}|X_\rho = \mathbf{X}_\rho). \tag{1.7}$$

1.3.2 Point Process

Point Processes allow to model such phenomenons Φ, as set of parametric objects representing geometrical objects such as segments, circles, rectangles, polygons etc... We will go further into details of Point Processes later on. Here we consider configurations $\mathbf{y}$ as a set $\{y_1, \ldots, y_{n(\mathbf{y})}\}$ of $n(\mathbf{y})$ elements y that represent our objects of interest.

Within the Bayesian framework, an observation model can be built as:

$$\mathbf{P}(X_\rho = \mathbf{X}_\rho|Y = \mathbf{y}) = \mathbb{1}(\rho \in \mathbf{y})\mathbf{P}(X_\rho = \mathbf{X}_\rho|\rho \in \mathbf{y}) + \mathbb{1}(\rho \notin \mathbf{y})\mathbf{P}(X_\rho = \mathbf{X}_\rho|\rho \notin \mathbf{y}). \tag{1.8}$$

In this formula, we apply a foreground ($\rho \in \mathbf{y}$) or background ($\rho \notin \mathbf{y}$) model whether a pixel ρ belongs to the silhouette of $\mathbf{y}$ or not. A common model if the silhouette corresponds to bright pixels on a darker background is to model the foreground $\mathbf{P}(X_\rho = \mathbf{X}_\rho|\rho \in \mathbf{y})$ as a high value Gaussian law. The background $\mathbf{P}(X_\rho = \mathbf{X}_\rho|\rho \notin \mathbf{y})$ is then mapped to a lower value (Baddeley & Lieshout, 1993; Perrin et al., 2004).

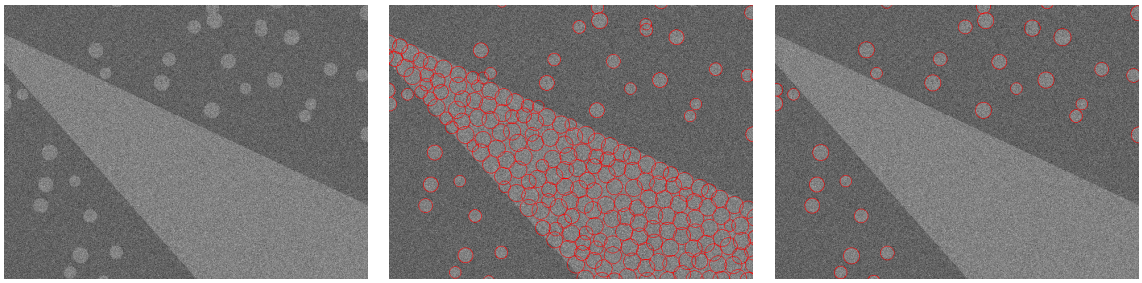
However, this Bayesian approach has some limitations (Ben Hadj et al., 2010):

- Non-homogeneous backgrounds are harder or impossible to model. The simple Gaussian model for $\mathbf{P}(X_\rho = \mathbf{X}_\rho|\rho \notin \mathbf{y})$ is no longer usable.
- The likelihood of pixels of the foreground class does not rely on the morphological properties of the class to extract. For instance the model may try to fit as many small objects it can in a large silhouette (Craciun et al., 2015) as illustrated in Figure 1.4.

For those reasons, (Ben Hadj et al., 2010) propose a *detector* model, formulating the *external energy* U_d as:

$$U_d(\mathbf{y}, \mathbf{X}) = w_d \sum_{y \in \mathbf{y}} V_d(y), \text{ with} \tag{1.9}$$

$$\mathbf{P}(\Phi = \mathbf{y}|X = \mathbf{X}) \propto \exp\left(-(U_d(\mathbf{y}, \mathbf{X}) + U_p(\mathbf{y}))\right), \tag{1.10}$$



(a) (b) (c)

Figure 1.4: Limitation of the Bayesian approach: (a) synthetic image; (b) Bayesian model; (c) *detector* model using contrast measures. *From (Ben Hadj et al., 2010).*

where w_d is a scalar weight, and U_p is the prior model formulated as an energy. Meanwhile, $V_d(y)$ measures the statistical difference between the inside and outside of object y, with a low value for high contrast (likely an object) and high values for low contrast (likely not an object). Still these contrast measures rely on the high and consistent contrast of the object against its background. This approach reaches its limits when the background start to exhibit contrasted elements that should not be detected for instance. We show a practical example of these limits in Section 4.2.1.1.

Notably, (T. Li et al., 2019) propose to perform such contrast based approach on the output of a CNN trained for segmentation; that way the authors fall back into a simple foreground and background where the contrast measure is efficient.

1.3.3 Convolutional Neural Networks

While the above image model with independent pixel was motivated in part by the computational limitations, the increasing availability of computational power eased the rise of Convolutional Neural Networks (CNN). Although introduced for image classification in 1989 (LeCun et al., 1989), it was only in the 2010s that CNN gained widespread adoption with AlexNet (Krizhevsky et al., 2012) for classification or RCNN (Girshick et al., 2014) for object detection.

Convolutional Neural Networks process data from a grid-like topology through a sequence of convolution and pooling (spatial aggregation) operations. We will go further in depth about CNN later in the thesis. Deep convolutional models learn the convolution filters for a specific task from large image databases such as ImageNet (Krizhevsky et al., 2012) for instance. Initially those were built towards classification of images (i.e. assigning a category to the whole image, usually describing its main content). Then (Girshick et al., 2014) propose RCNN that classifies a series of bounding boxes to build the first *two stages* object detector. Segmentation model such as Unet (Ronneberger et al., 2015) perform pixel-wise classification. Those produce real-valued maps in the range $[0, 1]$, that are interpreted as class probabilities. For instance, the probability of pixel ρ belonging to class C_1 (amongst N_C classes) is given as:

$$\widehat{P}(\rho \in C_1|\mathbf{X}) = \frac{\exp\left(F(\mathbf{X})[\rho, C_1]\right)}{\sum_{k=1}^{N_C} \exp\left(F(\mathbf{X})[\rho, C_k]\right)}, \tag{1.11}$$

with $F(\mathbf{X})$ is the CNN output tensor in $\mathbb{R}^{\mathcal{S}_d \times [\![1, N_C]\!]}$, and N_C the number of classes. This approach is leveraged in *heatmap* based object detection such as (X. Zhou, Wang, & Krähenbühl, 2019).

The *heatmap* corresponds to a *centerness* probability map, measuring the probability of each pixel to be an object center. One has to be careful about the interpretation of these as probabilities: as (C. Guo, Pleiss, Sun, & Weinberger, 2017) show, these measures are often poorly calibrated and should rather be considered as *scores*.

Nonetheless, this score measures the presence or not of an object, with a lightweight convolutional model which kernels are learned from data.

1.4 Proposed approach

The starting point of our work — inspired from (T. Li et al., 2019) — is to leverage the score $\widehat{P}(\rho \in C_1|\mathbf{X})$ and use it to build the external energy of the Point Process V_d, replacing the contrast measure.

In this PhD thesis, we propose the incorporation of interaction models into object detection methods, while taking advantage of the capabilities of deep convolutional neural networks. Satellite images are inherently imperfect due to atmospheric disturbances, partial occlusions and limited spatial resolution. To compensate for this lack of visual information, it becomes essential to incorporate prior knowledge about the layout of objects of interest.

On the one hand, methods based on CNN are excellent for extracting patterns in images, but struggle to learn object-to-object interaction models without having to introduce attention mechanisms such as *Transformers* that considerably increase the complexity of the model. For example, some approaches propose to include a priori information in the form of descriptive text about objects and their relationships (Lu et al., 2023), while others use cascades of attention modules (Zeng et al., 2023).

On the other hand, Point Processes propose to jointly solve the likelihood relative to the image (data term), and consistency of the object configuration itself (prior term). Firstly, Point Process approaches model configurations as vector geometry, constraining the state space by construction. In addition, these models allow explicit priors relative to configurations to be specified as energy functions. However, in most of the literature (Verdié & Lafarge, 2014; Schmidt et al., 2017), data terms rely on contrast measures between objects of interest and background. We illustrate the limitations of these measures on satellite data later in this thesis.

Instead of increasing the complexity of the model by adding, for example, *Transformers*, we propose in this thesis to combine CNN pattern extraction with the Point Process approach. The starting point of this approach is to use the output of a CNN as the data term for a Point Process model. From the latter we derive more efficient sampling methods for the Point Process that do not rely on application specific heuristics. Finally, we propose to bridge the gap in terms of parameters estimation using modern learning techniques inspired from Energy Based Models.

1.5 Thesis structure

The thesis is organized as follows. We choose to present our contributions structured into the three main elements of our models (energy model in Chapter 4, sampling and parameters estimation in Chapter 5) rather than going through the incremental improvements of each model we published (which would induce many repetitions) :

- **Chapter 2** provides a review of the State Of The Art (SOTA) approaches for object detection both with CNN and PP based methods, with a focus on remote sensing applications.

- **Chapter 3** goes through the fundamental theory on Point Process and the basics of Convolutional Neural Networks necessary to build our models.
- **Chapter 4** introduces several ways to build our Point Process model, in which we contribute with several novel approaches:
  - Building data terms for PP from CNN.
  - Interpreting classical CNN output as energy to incorporate in the PP model.
- **Chapter 5** develops the methods to sample and estimate the parameters of the model. Our main contributions are:
  - Thanks to CNN output potential maps we adapt sampling methods for more efficiency (parallel sampling with cell picking, birth with density from truncated model).
  - We propose two parameter estimation methods: the first based on Support Vector Machines ; the second based on gradient descent and contrastive divergence inspired from the literature related to Energy Based Models.
  - We propose using the Papangelou intensity as a per object score value in order to compute precision recall curves and compare our method to the classical ones.
- **Chapter 6** shows application of our model and methods on synthetic and real data on benchmarks and data provided by Airbus Defense and Space (ADS).
- **Chapter 7** summarizes contributions, limitations and perspectives for future works.

CHAPTER 2

State of the art

Dans ce chapitre, nous passons en revue la littérature sur la détection d'objets et ses méthodes associées. Notre objectif étant de construire un modèle combinant Processus Ponctuels (PP) et Réseaux de Neurones Convolutifs (CNN), nous nous concentrons d'abord sur les méthodes utilisant les CNN pour la détection d'objets, et plus spécifiquement ceux appliquées aux données de télédétection. Nous nous intéressons ensuite aux approches fondées sur les PP, appliquées à la détection d'objets. Enfin, nous abordons le domaine des modèles fondés sur l'énergie, car leur approche sera utile ultérieurement lors d'apprentissage de notre modèle.

In this chapter we review the literature around object detection and its related methods. As our aim is to build a model combining Point Processes (PP) and Convolutional Neural Networks (CNN), we first focus on the methods using CNN models for object detection, and more specifically applied to remote sensing data. We then look into the approaches based on PP, applied to object detection. Finally, we delve into the realm of Energy Based Models as their approach will be of use when training our model later on.

2.1 Convolutional Neural Networks for object detection 28
- 2.1.1 The rise of CNN models for object detection 28
- 2.1.2 CNN for detection in remote sensing 30
- 2.1.3 Remote sensing datasets 34

2.2 Point Processes for object detection 34
- 2.2.1 Point Process models 35
- 2.2.2 Sampling the Point Process 35
- 2.2.3 Point Process parameter estimation 36

2.3 Energy Based Models 36
- 2.3.1 Energy Based Models as generative models 36
- 2.3.2 Learning EBMs 37
- 2.3.3 EBM applications to computer vision 37

Looking back at the references used for this thesis, we identify three main topics, which we will explore in this chapter. Figure 2.1 shows relations between publications and topics, as well as these thematic clusters. As our aim is to build a model combining Point Processes (PP) and Convolutional Neural Networks (CNN), we first focus on the methods using a CNN for object detection, and more specifically applied to remote sensing data. We then look into the approaches based on Point Processes, applied to object detection. Finally, we delve into the realm of Energy Based Models (EBM) as their approach will be of use when training our model later on.

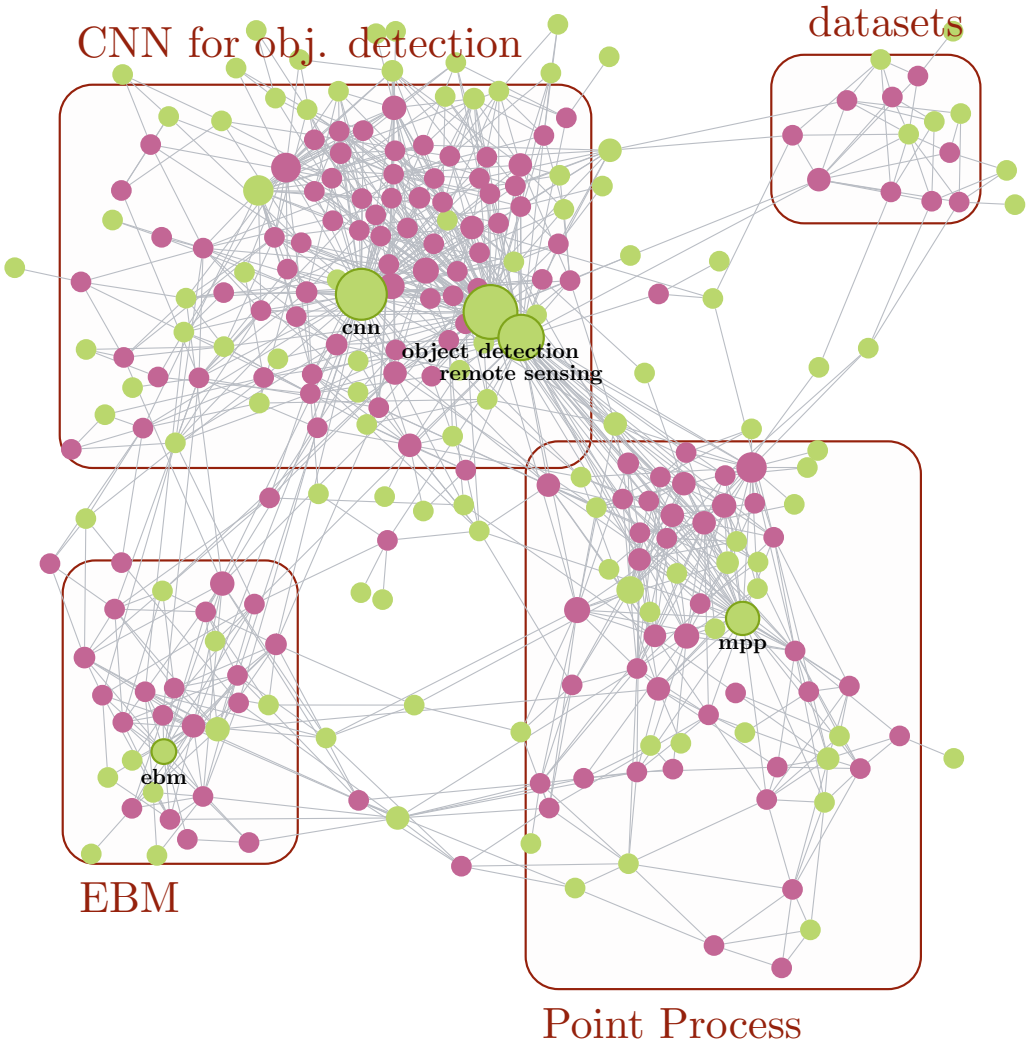


Figure 2.1: A graph of the bibliographic references, **papers** are linked together through (most of) their references and to the **themes** (or keywords) of these papers. We highlight the main clusters in this graph.

2.1 Convolutional Neural Networks for object detection

In this section we review the object detection first through the emergence of Convolutional Neural Networks (CNN) and then their use for remote sensing. We distinguish *usual* images — photographs usually taken at human eye-level with large mostly-centered objects of interest — from the images seen in remote sensing (or in microscopy for instance), as those have different characteristics that will be discussed further in this chapter.

2.1.1 The rise of CNN models for object detection

This first part presenting a short history of object detection is mainly sourced from (Zou, Chen, Shi, Guo, & Ye, 2023), in which the reader will find an even more thorough survey of object detection over the last 20 years. First it is important to mention that object detection is adjacent to many other computer vision tasks such as segmentation, classification, change detection etc...

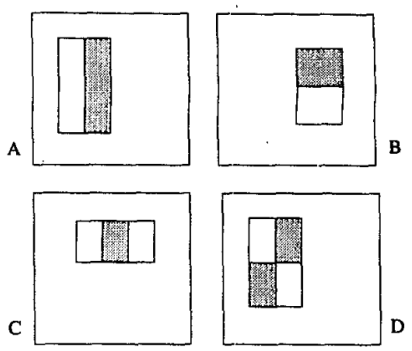


Figure 2.2: Example rectangle features shown relative to the enclosing detection window. The sum of the pixels which lie within the white rectangles are subtracted from the sum of pixels in the gray rectangles. *From (Viola & Jones, 2001).*

Initially, classical detection methods were based on handcrafted features such as for (Viola & Jones, 2001). This real-time face detection algorithm uses a sliding window with some handcrafted rectangular features (see Figure 2.2) for which a clever use of integral images allows to simplify the algorithm to a few lookup operations.

As handcrafted features showed their limit, object detection went through a shift with the arrival of Convolutional Neural Networks. Thanks to previous key works on backpropagation (LeCun et al., 1989), CNNs took off in computer vision in 2012 with (Krizhevsky et al., 2012) introducing a large scale training data (ImageNet) and proposing a CNN model (AlexNet) trained to perform image classification in *usual* images.

Two stage detectors. In 2014 Girshick et al. propose *Region with CNN features* (RCNN) in (Girshick et al., 2014) to bridge the gap from classification to object detection. RCNN works by proposing a set of objects as boxes. Each box is then translated into a vector of features by a CNN model trained on ImageNet (Krizhevsky et al., 2012), a Support Vector Machine (SVM) then predicts the presence and class of the potential object in the box (see Figure 2.3). The large number

of proposals that overlap is a limitation of that method (over 2000). In later works (Girshick, 2015) propose FastRCNN, simultaneously training the detector and a bounding box regressor of the model, making for a great improvement in speed and performance. Even though (Ren, He, Girshick, & Sun, 2015) introduce the region proposal network, computational redundancy remains due to the high number of proposals. Such approach of box proposal then classification is referred to as *two stage* approach.

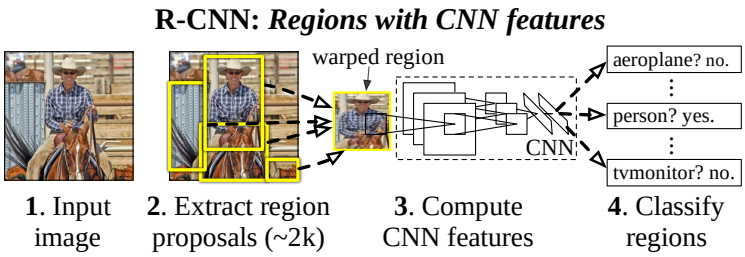


Figure 2.3: Region with CNN features method overview. *From (Girshick et al., 2014).*

From these works, improvements are proposed to alleviate the limitations of the initial two stage models: for instance, (K. He, Zhang, Ren, & Sun, 2015) propose Spatial Pyramid Pooling (SPP) to bypass the fixed input size restriction of previous CNN models with a new pooling method. Later on Feature Pyramid network (FPN) proposed in (Lin, Dollar, et al., 2017), uses features of deeper layers — previous approaches would use only the last layer — introducing top down and lateral connections. This architecture integrated in FastRCNN gives improved results.

Single stage detectors. The first *single stage* object detector is introduced in (Redmon, Divvala, Girshick, & Farhadi, 2016) as YOLO (You Only Look Once). It shows greater performances in terms of object detection metrics and speed than the two-stage approaches at the time. While YOLO initially struggles with small objects, the following version, YOLOv4 (Bochkovskiy, Wang, & Liao, 2020), YOLOv7 (Wang, Bochkovskiy, & Liao, 2023) and so on, incrementally alleviate these issues.

In their work (Lin, Goyal, Girshick, He, & Dollar, 2017) propose that the reason one stage detectors kept being outperformed by modern two stage detectors was the imbalance in foreground and background classes. They propose a new *focal loss* to put more focus on hard, misclassified elements when training the single stage CNN model.

For most previous methods, *anchor boxes* are used as reference box proposals for classification and regression. As the objects are varied in size and shape, the user has to set up these reference boxes tailored to the application. CornerNet (Law & Deng, 2018) propose to instead view the bounding box estimation as a keypoint detection problem: for each object they infer the two points that would define a bounding box, the points are matched together by some extra embedding information. The estimation of the points' location is done though some *heatmap*: raster maps corresponding to a pseudo probability map of points presence. Those are used not only for object localization: for instance (Y. Guo, Wu, Du, & Zhang, 2022) use heatmaps to count vehicles in satellite images. CenterNet simplifies this approach further (X. Zhou et al., 2019), considering objects to be a single point (their center) extracted through a heatmap. From the center location is regressed the other attributes such as size, orientation, location, and pose (see Figure 2.4).

keypoint heatmap [C] local offset [2] object size [2]

Figure 2.4: Heatmap for center detection. Here the heatmap resolution is coarser than the image resolution, the local offset allows compensating for the discretization error. *From (X. Zhou et al., 2019).*

Attention and transformers. Attention mechanisms were introduced in Natural Language Processing (NLP) (Bahdanau, Cho, & Bengio, 2015) to help learn relations between distant words in a sentence, and quickly gained popularity in their field. Works in (Vaswani et al., 2017) introduced *transformers*, a deep neural network block relying on attention mechanisms, that aims to use global dependencies between input elements, while avoiding recurrence based models[1]. It did not take long for the object detection community to adapt this method to object detection. In 2020 (Carion et al., 2020) propose DEtection TRansformer (DETR) — which as the name suggests uses transformers for object detection — viewing detection as a set prediction problem and using transformers to learn relations between the elements of these sets; i.e. relations between objects. We discuss of the limitations of attention mechanisms in next Section 2.1.2.

2.1.2 CNN for detection in remote sensing

We now focus on the application of the object detection methods above to remote sensing data. We articulate this section through the multiple challenges that remote sensing data exhibits and the proposed methods to alleviate those.

Small objects. In remote sensing, and more specifically with satellite imagery, objects of interest can be of very small size (only a few pixels). A first approach is to use super-resolution methods to compensate for the object size limitations of the detection model as in (J. Zhang, Lei, Xie, Fang, et al., 2023). We see it used also for microscopy images in (Mayo, Anantrasirichai, Chalidabhongse, Palasuwan, & Achim, 2022) using Pix2Pix (Isola, Zhu, Zhou, & Efros, 2017) for rescaling images. One limitation of super-resolution approaches is the very large memory footprint of resulting images (images being large in the first place, even at low resolution). Moreover, deep-learning super-resolution models have a risk of hallucinating patterns; i.e. introducing non-existing and non-relevant information into the scaled-up image. The Feature Pyramid Network (Lin, Dollar, et al., 2017) introduced previously, is great at improving detection at multiple scale, as the lateral connections in the architecture allow to maintain a good localization while detecting

[1]Previous approaches in NLP would build models that would run through each work of the sentence one by one, while updating a hidden state. This sequential approach limits parallelization and is limited by the memory that the hidden state represents (Vaswani et al., 2017).

object at higher scales in the deeper part of the model. This type of architecture is similar to the Unet introduced in (Ronneberger et al., 2015).

It is to note that small objects induce some issues on the metrics used to evaluate or train the models. Indeed, the Intersection Over Union (IOU) becomes instable for small objects; (Jeune & Mokraoui, 2023) propose a more perceptually accurate measure for small objects.

Large images. Satellite images span huge areas of land, resulting in oftentimes very large images. It requires CNN approaches to be able to handle varied image size, and be able to process patched images; as memory is limited, one needs to be able to split the image, process each patch independently, and stitch back the results without inconsistencies in the output. Fully Convolutional Networks (FCN) formalized in (Long, Shelhamer, & Darrell, 2015) for segmentation, then in (Tian, Shen, Chen, & He, 2019) for object detection (see (Sun et al., 2021) for an application in remote sensing), use CNN with only convolution and pooling operations, allowing for translation invariance. FCN are introduced in biomedical image segmentation with the Unet (Ronneberger et al., 2015).

Limited spatial resolution. In remote sensing, the sensor is intrinsically quite distant from the observed objects, implying a limited spatial resolution (usually measured in meters per pixel). This limited resolution in turn induces a limited amount of visual information to perform object detection from. Combines with sensor noise or atmospheric perturbation, remote sensing detection methods need to innovate in compensating the limited signal. Approaches such as (LaLonde, Zhang, & Shah, 2018) (Corsel, van Lier, Kampmeijer, Boehrer, & Bakker, 2023) use the temporal information from image time series to improve detection of small objects. However, static small objects remain difficult to detect with that method. Others propose using the multiple modalities of data the satellite sensor can produce; using the multiple spectral bands instead of optical grayscale or RGB. For instance (Belmouhcine, Burnel, Courtrai, Pham, & Lefèvre, 2023) propose to fuse sensors with attention mechanisms, and (J. Zhang, Lei, Xie, Fang, et al., 2023) combined this with super resolution. The method proposed in (Lu et al., 2023) uses text-modal descriptors to introduce prior knowledge on the objects into the model and their relation. For instance *"An airport consists of [...] runways for planes to take off and land"* introduces a relation between the airport, runway and plane objects and their co-occurence. Similarly, the model proposed in (K. Zheng, Dong, Xu, Tan, & Huang, 2023) learns the co-occurence of objects in images in the training data. Another approach is to model the interaction between objects. Markov Random Field approaches (Moser, Serpico, & Benediktsson, 2013) model pixel spatial context, which is combined with CNN for image segmentation in (Pastorino, Moser, Serpico, & Zerubia, 2022). Works in (Z. Zheng, Zhong, Wang, Ma, & Zhang, 2023) propose to relate foreground and background elements while (Cao, Bai, Pang, Liu, & Zhang, 2023) use transformer to learn object-level relations. Finally, (Zeng et al., 2023) use prior on objects alignment (thus considering object level spatial interaction) within a cascade of attention modules.

Attention models applied to images remain limited in use due to their limitations in terms of memory footprint: often time the solution would be to sub-sample (loosing spatial precision) or consider attention locally (loosing longer range interactions) (Cherel, Almansa, Gousseau, & Newson, 2022). Alternatively (Cherel et al., 2022) propose an efficient attention layer based on a stochastic algorithm for patch matching which is used for determining approximate nearest neighbors.

The limited signal-to-noise ratio raises concerns about the robustness of the developed methods. Works in (Mei et al., 2023) and (H. He, Ding, & Xia, 2023) study the robustness of remote sensing computer vision methods. (Mei et al., 2023) show the effect of natural perturbations (Gaussian blur, fog, etc...) as well as adversarial attacks including background obfuscation patterns on which to place objects to make those virtually invisible to the detector.

Oriented objects. The detection of oriented objects raises some challenges compared to the detection of horizontal (non-oriented) bounding boxes. Importantly the angle parameter defining the oriented rectangle introduces a discontinuity (or cyclic aspect) to the parameter space which poses problem when computing the loss over the inferred configurations of rotated objects. To bypass this limitation (Xu et al., 2023) propose reformulating angles as a circular Gaussian density, while (Yao et al., 2022) or (D. Yu et al., 2023) propose redefining the oriented bounding box representation into a continuous one. Finally, (Llerena, Zeni, Kristen, & Jung, 2021) and (Z. Li et al., 2023) propose a *fuzzy* formulation of bounding boxes using Gaussian distributions and a probabilistic IOU. The above-mentioned IOU issues caused by small objects is even more present with oriented objects, where a small angle error can drastically change the IOU as (Xu et al., 2023) note.

Annotated data is costly. As for any deep learning approach, annotated data is key to training models. As data annotation is a time-consuming and arduous task, some methods try to learn efficient model with limited amounts of data. Weakly supervised methods propose to learn models on sparsely annotated data to extract more fine-grained information at inference. For instance (M.-T. Pham, Gangloff, & Lefèvre, 2023) train a Variational Auto-Encoder (VAE) to perform image reconstruction, and use it to perform detection of marine animals. Similarly, (J. Bai et al., 2023) train a model to classify images (requiring simpler annotations than object location Ground Truth (GT)). The authors then use Class Activation Maps (CAM, introduced in (B. Zhou, Khosla, Lapedriza, Oliva, & Torralba, 2016)) to extract the object(s) location in the image looking at which part of the image activates the classification output. While weakly supervised models are trained on sparser data, few shot models are built to learn with limited data samples, for instance being able to categorize never seen classes (as humans can do) (Antonelli et al., 2022). Previously mentioned works in (Lu et al., 2023) combine few shot model with the text-modal priors.

Lightweight models for onboard processing. A challenge for some applications in remote sensing is the ability to apply the algorithm in flight, aboard the satellite for instance. Onboard computing faces multiples challenges: the monetary and energetic cost of launching mass (thus processing power) in orbit, the necessity of systems to be stable and reliable (no on-site maintenance possible), limitations with power and cooling, etc... Thus, lightweight models for remote sensing computer vision are key to some applications. For that matter (Hu et al., 2023) propose to train a model based on YOLO with limited number of parameters. Other methods such as neural network reduction (Vandame, Karam, Argentier, & Chanussot, 2023) utilize distillation (Hinton, Vinyals, & Dean, 2014) to transfer knowledge from a larger model to a sparser one. Lastly, quantization (limiting the number of bits the model parameters are stored on) is proposed in (J. Zhang, Lei, Xie, Li, et al., 2023) using distillation.

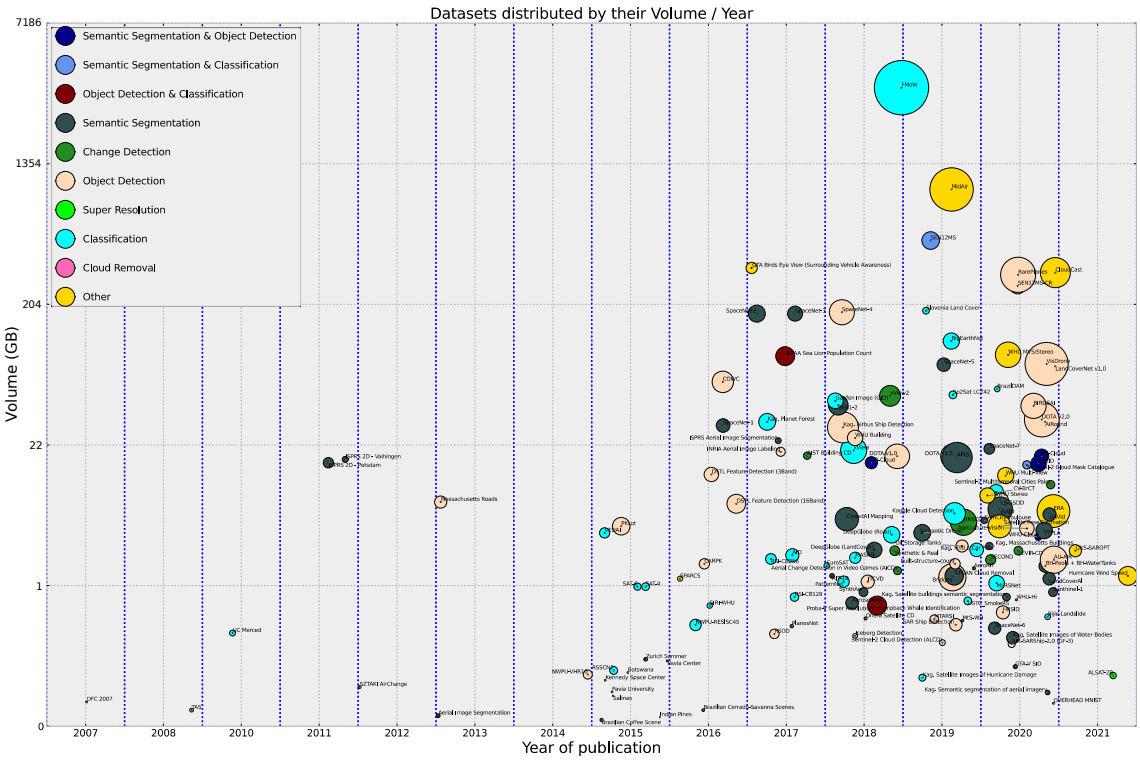


Figure 2.5: Evolution of remote sensing datasets dedicated to machine learning tasks. The vertical axis represents data volume, while the circle size maps to the number of spatial pixels covered. *From (Schmitt et al., 2021).*

2.1.3 Remote sensing datasets

As data is the key to deep learning approaches for remote sensing, (Schmitt et al., 2021) provide a review of available datasets in remote sensing (see Figure 2.5). Amongst the object detection datasets we identify a few where our approach could bring improvements on:

- COWC (Mundhenk et al., 2016): satellite optical images with vehicle position annotations. *As this dataset only provides positions and no geometries, we only use this dataset for the first non-marked point process models.*
- DOTA v2.0 (Xia et al., 2018): aerial and satellite optical images with oriented rectangles annotations for multiples classes including ground vehicles, planes, boats, peers etc... *It provides a large diversity of contexts and objects density, we use it extensively for training and estimation of our methods.*
- spacenet 4 (Weir et al., 2019): optical satellite images with annotated building footprints. *While not investigated in this thesis, applying point process models to building extraction is quite relevant as the interaction priors are strong (Ortner, Descombes, & Zerubia, 2008)*
- BIRDSAI (Bondi et al., 2020): aerial thermal images from drones, with person tracking annotations. *Point processes are also useful to take into account priors on dynamics (Craciun et al., 2015). We thus mention this dataset for future works.*
- AU-AIR (Bozcan & Kayacan, 2020): drone optical images with horizontal bounding boxes for multiple classes of objects. *While this dataset would be interesting to model priors over dynamics, the viewing angle and horizontal bounding boxes make the elaboration of interaction priors less pertinent.*
- RarePlanes (Shermeyer et al., 2021): synthetic optical images (reproducing satellite data) with fine plane location and type annotations. *The non-oriented bounding boxes this dataset provides, and the limited interaction between neighboring planes make this dataset of limited interest to our approach.*
- HRSID (Wei et al., 2020): synthetic Aperture Radar (SAR) satellite images, with annotated oriented ships. *This dataset provides a good challenge to test a model robustness to noise. However, the objects of interest are often isolated, thus the modeling of interaction priors is of limited interest. Still the dataset exhibits some high density samples that are worth looking into.*
- VisDrone (Zhu et al., 2022): optical drone images with annotated tracks of bounding boxes. *Same as for AU-AIR, the viewing angle and horizontal bounding boxes make the elaboration of interaction priors less pertinent.*

2.2 Point Processes for object detection

Point Processes model the distribution of point in space (Cox & Isham, 1980; Daley & Vere-Jones, 1988; Lieshout, 2000). These are used for a variety of tasks such as modeling the species distribution for ecological studies (Renner et al., 2015), random tessellation for image modeling

(Bordenave, Gousseau, & Roueff, 2006), building fractal-like Poisson cascades for super resolution (Chainais, Kœnig, Delouille, & Hochedez, 2011), or Determinantal Point Processes (DPP), a repulsive Point Processes used for efficient sampling of a volume (Macchi, 1975) used for column subset selection (Belhadji, Bardenet, & Chainais, 2020) or Kernel quadrature (Belhadji, Bardenet, & Chainais, 2019). Here we focus on Point Process for image analysis/object detection (Descombes & Zerubia, 2002; Descombes, 2013).

2.2.1 Point Process models

Multiple flavors of points. While simple Point Processes model the spatial distribution of points, Marked Point Processes add a random vector to each point modeling some attributes of the objects; size, angle, color, type of object etc . . . This allows modeling a variety of objects:

- Circles to model flamingos in (Descamps, Descombes, Bechet, & Zerubia, 2008).
- Segments for road detection (Lacoste et al., 2005), to model blood vessels (T. Li, Comer, & Zerubia, 2020), or networks in general (Schmidt et al., 2017).
- Rectangles in time, adding a persistent label to each object in space and time allows for tracking objects (Crăciun, 2015).
- Contours as polygons for object detection in microscopy images (Kulikova, Jermyn, Descombes, Zhizhina, & Zerubia, 2011) or as a dictionary of precomputed shapes (Descombes, 2017).
- Mix of processes; for instance (Ortner et al., 2008) use a mix of rectangles and segments to model buildings and the networks separating those.

Energy terms for the Point Process. Point Process models for object detection are defined from a density that on one side models prior on objects such as their shapes or interactions, and on the other the fitness of the point configuration against the image. Most of the approaches presented above use contrast measures. These measures are based on statistical tests between the inner and outer pixels of an object such as T-test (Student, 1908) or the Bhattacharyya distance (Goudail, Réfrégier, & Delyon, 2004). These measures perform well on high foreground-to-background contrast images with limited clutter in the background. Notably, (T. Li et al., 2019) build a Point Process with contrast measures from the output of a CNN model trained for segmentation. It allows transforming the original image into a highly contrasted map to apply the contrast measure on.

2.2.2 Sampling the Point Process

Sampling a Point Process poses a challenge due to the non-fixed dimension of the space to sample in; as the number of points (i.e. number of objects) is a random variable too. Using the Reversible Jump Monte Carlo Markov Chain developed by (Green, 1995) and (Geyer & Møller, 1994) allow improving from the classical Metropolis Hasting algorithm (Metropolis, Rosenbluth, Rosenbluth, Teller, & Teller, 1953) to jump properly in between dimensions. It involves adding a Birth and Death perturbation kernel that adds or removes one point at a time. To speed up sampling (Descombes, Minlos, & Zhizhina, 2009) introduce multiple births and deaths to propose more than a single point per step, later on improved with graph cuts in (Eldin, Descombes, Charpiat,

& Zerubia, 2012). For more efficient fixed-dimension exploration (Lafarge, Gimel'farb, & Descombes, 2010) and (Tu & Zhu, 2002) (doing image segmentation with data driven MCMC) use Jump Diffusion from (Grenander & Miller, 1994). It uses the energy gradient to modify the current state instead of proposing perturbations independent of the energy (with a simple Gaussian perturbation for instance). This diffusion process stems from Langevin dynamics (Welling & Teh, 2011). As Point Process exhibit spatial Markovianity, (Verdié & Lafarge, 2014) propose sampling the Point Process in parallel throughout the image, allowing to process more than one perturbation per step and making use of multiple core computing. Finally, (T. T. Pham et al., 2016) propose to approximate the state space into a finite set, the sampling of the most likely configuration of the Point Process becomes a binary variable optimization problem

2.2.3 Point Process parameter estimation

As any model, Point Process models come with their set of parameters to estimate. One of the most important is the relative importance/weight of the several priors and data terms of the energy model that define the Point Process density. In their work (Craciun et al., 2015) and (Q. Yu & Medioni, 2009) generate a set of linear constraints from generated *bad* configurations, and solve for the parameters with linear programming. For their model (Chatelain, Descombes, & Zerubia, 2009) use an approximation of the Expectation Maximisation algorithm (Dempster, Laird, & Rubin, 1977) to learn the Point Process parameters; using pseudo-likelihood estimation and stochastic EM (SEM) (Celeux, Chauveau, & Diebolt, 1995). Lastly, (Hurtut et al., 2009) propose to synthesize textures (here arrangements of elements in space), from an example patch. To do that, they extract a set of perceptually meaningful statistics, and build a Point Process such that its statistics matches the observed ones by maximizing the pseudo log likelihood.

2.3 Energy Based Models

"Energy-Based Models (EBMs) capture dependencies by associating a scalar energy (a measure of compatibility) to each configuration of the variables" (LeCun, Chopra, Hadsell, Ranzato, & Huang, 2006). These models define a distribution over a state space from a Gibbs distribution (also known as Boltzmann distribution) as proportional to the exponential of an energy $h \propto \exp(-U)$. As most of the Point Processes are defined through such a distribution, and aim to capture the dependency between a configuration of points and the image, it makes these Energy Based Models as defined above; we shall then consider methods developed in the EBMs literature for our Point Process model.

2.3.1 Energy Based Models as generative models

Recent interest in Energy Based Models sparks from their generative capabilities; as those define a density, one can sample new elements $\mathbf{x} \sim \exp(-U(\mathbf{x}))$. Initially generative models (e.g. image generation) were based on Generative Adversarial Networks (GAN) (Goodfellow et al., 2014) built from Variational Auto Encoders (VAE) (Kingma & Welling, 2014). However, in recent year, popular image generation models such as (Ramesh, Dhariwal, Nichol, Chu, & Chen, 2022) leverage diffusion models (Ho, Jain, & Abbeel, 2020) that progressively denoise an image into a new sample (see Figure 2.6). This denoising mimics Langevin dynamics, but instead of computing the gradient of a learned energy function, (Ho et al., 2020) estimate the gradient value directly.

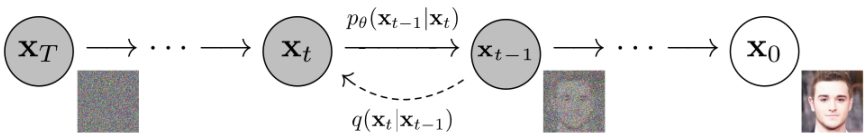


Figure 2.6: Diffusion model for image generation. The model learns the transition kernel p_θ via the gradient of an energy within Langevin dynamics. *From (Ho et al., 2020).*

2.3.2 Learning EBMs

In their work (LeCun et al., 2006) propose to learn EBMs by maximizing their likelihood within a gradient descent scheme. However, this formulation produces some intractable integrals over the state space. While those can be approximated with Monte Carlo sampling, (Hinton, 2002) proposed the *Contrastive Divergence* method, in short taking a single sample of the Monte Carlo sampling, initialized at the ground truth to which only a few steps of the Markov Chain is applied. Later on (Tieleman, 2008) propose *persistent divergence*, initializing this contrastive sample with the sample obtained at the previous iteration of the gradient descent. We find this method used for image generation in (Du, Li, Tenenbaum, & Mordatch, 2021) and (Song & Ermon, 2019). We will go further in details about these methods in Section 5.2.2.

2.3.3 EBM applications to computer vision

Generative models can seem quite remote from our object detection tasks. However, we see some interesting works proposing to use the composable nature of EBMs[2] such as (R. Zhang et al., 2022) performing unsupervised object discovery and controllable scene manipulation. EBMs are intrinsically bound to classification task, as (Grathwohl et al., 2019) note: any classifier with a Softmax output, can be reinterpreted as an energy model. For instance (Castillo-Navarro, Le Saux, Boulch, & Lefèvre, 2021) propose a novel model for semi-supervised classification and generation of images in Earth observations.

[2]One can easily combine the densities of two EBMs by summing their energies.

PART

Model foundations

ฟันเฟืองขาม
เพลา
ฟันเฟืองตรง
ฟันเฟืองเอียง
ฟันเฟืองเอียงคู่

CHAPTER 3

Foundations for Point Processes and Convolutional Neural Networks

Ce chapitre introduit les notions fondamentales sur les Processus Ponctuels et Réseaux de Neurones à Convolution. Celles-ci sont nécessaires à la construction de nos modèles de détection dans les chapitres suivants. Pour plus de détails sur les Processus Ponctuels, le lecteur peut se référer à (Stoyan, Kendall, & Mecke, 1995; Lieshout, 2000), dont est issu en grande partie ce chapitre. Par la suite, on décrit comment modéliser la détection d'objets avec des Processus Ponctuels et les méthodes nécessaires pour échantillonner ces derniers. Enfin, sont présentés les outils relatifs aux réseaux de neurones à convolution qui seront utilisés dans nos contributions.

In this chapter we introduce the fundamental notions for Point Processes and Convolutional Neural Networks. These are needed to build our models in the later chapters. For more details on Point Processes please refer to (Stoyan et al., 1995; Lieshout, 2000) from which most of this chapter is sourced. We describe how to model object detection with Point Processes, and the necessary methods for sampling the latter. Finally, we define the necessary tools of Convolutional Neural Network needed for our contributions.

3.1 Point Process fundamentals **43**
3.1.1 Poisson Point Processes 43
3.1.2 Markov Point Processes 44
3.1.3 Markov marked Point Processes 46
3.1.4 Stability conditions 47
3.2 Point Process for object detection **48**
3.2.1 Energy model 49
3.2.1.1 Markovianity relation 49
3.2.1.2 Local stability 50
3.3 Sampling the Point Process **50**
3.3.1 Markov chains 50
3.3.2 Reversible Jump Monte Carlo Markov chain 52
3.3.2.1 Detailed Balance 53
3.3.2.2 Balancing in terms of random numbers 54
3.3.2.3 Perturbation kernels 54
3.3.2.4 Parallel sampling 57
3.3.3 Jump diffusion 57
3.3.4 Simulated annealing 58
3.3.5 Stopping conditions 59
3.4 CNN fundamentals **60**
3.5 Conclusion **62**

3.1 Point Process fundamentals

3.1.1 Poisson Point Processes

We first introduce the fundamentals of Point Process. For more details please refer to (Stoyan et al., 1995; Lieshout, 2000).

We consider configurations of points in the space $\mathcal{S} \subseteq \mathbb{R}^d$ (e.g. a square in $\mathbb{R}^2$). In the rest of this chapter, we associate a metric d to the space $\mathcal{S}$; e.g. the Euclidean distance in $\mathbb{R}^d$.

Definition 3.1.1. We call **configuration**, denoted $\mathbf{y}$, a finite unordered set of points y^k, $k = 1, \ldots, n(\mathbf{y})$ in $\mathcal{S}$, with $n(\mathbf{y})$ denoting the number of points in $\mathbf{y}$:

$$\mathbf{y} = \{y^1, \ldots, y^{n(\mathbf{y})}\}. \tag{3.1}$$

The set of all possible configurations of points in $\mathcal{S}$ with any number of points is denoted:

$$\mathcal{Y} = \bigcup_{n=0}^{\infty} \mathcal{Y}_n, \tag{3.2}$$

where $\mathcal{Y}_0 = \emptyset$ and $\mathcal{Y}_n = \{\{y^1, \ldots, y^n\},\ y^i \in \mathcal{S},\ \forall i\}$.

A configuration $\mathbf{y}$ is said to be *locally finite* if for any bounded Borel set $A \subseteq \mathcal{S}$, the number of points of $\mathbf{y}$ in A is finite. The set of locally finite configurations is denoted N^{lf}.

Definition 3.1.2. A **Point Process** (PP) on $\mathcal{S}$ is a mapping Φ from a probability space $(\Omega, \mathcal{A}, \mathcal{P})$ into N^{lf} such that for any bounded Borel set $A \subseteq \mathcal{S}$. The number of points that fall in A, denoted $N_\Phi(A)$, is a finite random variable.

Practically, a Point Process is a random variable the realization of which is a configuration of points. If the space $\mathcal{S}$ is bounded or $N_\Phi(A)$ is almost surely finite, we call the Point Process *finite*.

For most Point Processes, a point pattern will not contain multiple points at exactly the same location, either because it is impossible, or the multiplicity might be encoded within a mark (see Definition 3.1.8); i.e. $N_\Phi(\{y\}) \in \{0, 1\}$ for all $y \in \mathcal{S}$. The set of locally finite configurations composed of distinct points is denoted N_s^{lf}.

Definition 3.1.3. A Point Process is **simple** if it takes its values in N_s^{lf} almost surely.

Definition 3.1.4. Given a measure ν on space $\mathcal{S}$, such that $\nu(\mathcal{S}) > 0$ and $\nu(A) < \infty$ for all bounded Borel set A. A Point Process Φ on $\mathcal{S}$ is a **Poisson Point Process** of intensity measure ν if and only if:

1. $N_\Phi(A)$ is a Poisson random variable of mean $\nu(A)$
2. For k disjoint Borel sets $A_1, \ldots, A_k$, random variables $N_\Phi(A_1), \ldots, N_\Phi(A_k)$ are independent.

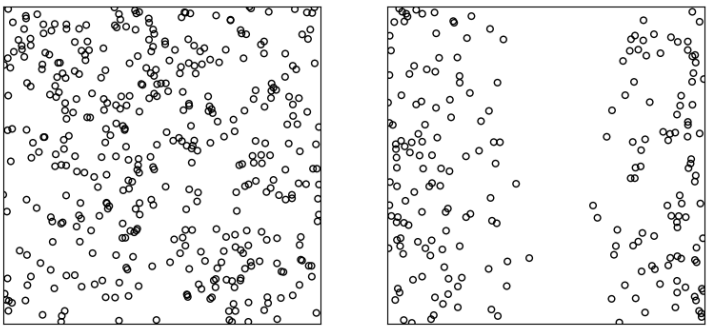
The Poisson Point Process is called *homogeneous* (see Figure 3.1 (a)) when ν is of the form $\lambda|A|$, with $|A|$ the Lebesgue measure of $A \subseteq \mathcal{S}$ and λ a strictly positive scalar. The intensity measure ν can be used to control the point density over the space $\mathcal{S}$ as shown in Figure 3.1 (b).

The law of a Poisson Point Process on intensity measure ν in a window $\mathcal{S} \subset \mathbb{R}^d$ is defined as

$$\mu(A) = \sum_{n=0}^{\infty} \underbrace{\frac{\nu(\mathcal{S})^n \exp(-\nu(\mathcal{S}))}{n!}}_{p_n} \underbrace{\int_{y^1 \in \mathcal{S}} \cdots \int_{y^n \in \mathcal{S}}}_{n \text{ times}} \mathbb{1}_A(\{y^1, \ldots, y^n\}) \frac{1}{\nu(F)^n} dy^1 \ldots dy^n \tag{3.3}$$

$$\mu(A) = \sum_{n=0}^{\infty} \frac{\exp(-\nu(F))}{n!} \int_{\mathcal{S}^n} \mathbb{1}_A(\mathbf{y}) \nu(dy^1) \ldots \nu(dy^n), \tag{3.4}$$

where A is part of the sigma-algebra $\mathcal{B}$ from $\mathcal{Y}$.



(a) Homogeneous **(b) Non-homogeneous**

Figure 3.1: Poisson Point Processes on $\mathcal{S} = [0, 1]^2$: left is homogeneous with $\lambda = 400$; right has intensity measure $\nu(i, j) = \lambda(\cos(2\pi i) + 1)/2$.

3.1.2 Markov Point Processes

While this modeling of points in space accounts for a wider variety of distributions, there is still no consideration for point interactions. We can build a wider range of Point Process models by means of their probability density with respect to the Poisson process. Here we consider a metric space $(\mathcal{S}, d)$ and the Poisson Point Process on $\mathcal{S}$ with density μ, and intensity measure ν. The probability density f of a Point Process w.r.t. the law μ of a Poisson Point Process is a mapping from the configuration space $\mathcal{Y}$ to $[0, \infty[$ such that:

$$f : \mathcal{Y} \mapsto [0, \infty[, \int_{\mathcal{Y}} f(\mathbf{y}) \mu(d\mathbf{y}) = 1. \tag{3.5}$$

With density f w.r.t. the measure of a Poisson process, we can now model interactions. While the density function can take into account interactions between any points in the configuration, real life spatial processes often only interact locally. This characteristic simplifies the modeling of the Point Process, and yields some properties that are useful when simulating it. The *locality* of the Point Process is formalized through *Markov Point Processes*:

Definition 3.1.5. We define the **neighborhood** $\partial_A^{\mathcal{S}}$ of a set $A \subseteq \mathcal{S}$ in $\mathcal{S}$ for reflexive and symmetric relation $\sim$ as:

$$\partial_A^{\mathcal{S}} = \{x \in \mathcal{S} : x \sim a \text{ for some } a \in A\}. \tag{3.6}$$

By extension the set of neighbors of a single point y is $\partial^{\mathcal{S}}_{\{y\}}$, and the set of points of configuration $\mathbf{y}$ neighboring y is $\partial^{\mathbf{y}}_{\{y\}}$.

From the definition of neighborhood, we can define a Markov Point Process:

Definition 3.1.6. Let Φ be a Point Process of density f. Point Process Φ is a **Markov Point Process** under the symmetric and reflexive relation $\sim$ if and only if, for every configuration $\mathbf{y} \in \mathcal{Y}$ such that $f(\mathbf{y}) > 0$:

1. $\forall \mathbf{x} \subset \mathbf{y},\ f(\mathbf{x}) > 0$
2. For every point $u \in \mathcal{S}$, $f(\{u\} \cup \mathbf{y})/f(\mathbf{y})$ only depends on u and its neighborhood in $\mathbf{y}$, $\partial^{\mathbf{y}}_{\{u\}}$.

The Hammersley-Clifford theorem allows the density of the Markov process to be decomposed as the product of local functions defined on cliques; a clique being a subset of $\mathbf{y}$ where all points are neighbors of each other.

Theorem 3.1.1. *A Point Process density $f : \mathcal{Y} \mapsto [0, \infty[$ is Markov w.r.t. the relation $\sim$ if and only if there is a measurable function $\Psi : \mathcal{Y} \mapsto [0, \infty[$ such that:*

$$f(\mathbf{y}) = \prod_{\mathbf{y} \in \mathcal{C}_{\mathbf{y}}} \Psi(\mathbf{y}), \tag{3.7}$$

where $\mathcal{C}_{\mathbf{y}}$ is the set of cliques $\mathcal{C}_{\mathbf{y}} = \{\mathbf{y} \subseteq \mathbf{y} : \forall \{y, y'\} \subseteq \mathbf{y},\ y \sim y'\}$

In practice, we often write f as the normalized version of an unnormalized function h:

$$f(\mathbf{y}) = \frac{h(\mathbf{y})}{Z}, \tag{3.8}$$

where $Z = \int_{\mathcal{Y}} f(\mathbf{y}) d\mathbf{y}$. Markov Point Processes are also known as *Gibbs Point Process*, since (3.7) can be written in an energy form, where $V(\mathbf{y})$ represents the potential of a clique :

$$f(\mathbf{y}) \propto h(\mathbf{y}) = \exp\left(-\sum_{\mathbf{x} \in C_{\mathbf{y}}} V(\mathbf{x})\right), \tag{3.9}$$

with $\propto$ standing for "proportional to" (skipping the normalizing factor $\int_{\mathcal{Y}} h(\mathbf{y}) d\mathbf{y}$).

The Markov property of a Point Process comes in useful when the above-mentioned normalizing factor becomes intractable. Indeed, the second property in definition 3.1.6 allows the normalizing constant to be canceled out; as long as the function h in integrable over $\mathcal{Y}$, we can skip computing its integral. The density ratio $f(\{u\} \cup \mathbf{y})/f(\mathbf{y})$ is in fact the Papangelou intensity, which we now define:

Definition 3.1.7. The **Papangelou intensity** $\lambda(\cdot;\cdot)$ associated to a simple Point Process Φ, can be interpreted as:

$$\lambda(y; \mathbf{y}) dy = p\left(N_{\Phi}(dy) = 1 | \Phi \cap (dy)^c = \mathbf{y} \cap (dy)^c\right), \tag{3.10}$$

i.e. the infinitesimal probability to find a point in region dy around $y \in \mathcal{S}$, given the configuration $\mathbf{y}$ outside dy ($(dy)^c$ being the complement of dy in $\mathcal{S}$).

We then have the following:

Theorem 3.1.2. *For a finite Point Process Φ specified by a density f w.r.t. a Poisson Point Process with intensity measure ν, the Papangelou conditional intensity is given as:*

$$\lambda(u; \mathbf{y}) = \frac{f(\{u\} \cup \mathbf{y})}{f(\mathbf{y})}, \tag{3.11}$$

for $u \notin \mathbf{y}$.

Example: pairwise interaction process. A typical Markov Point Process is the pairwise interaction process, for which the density function is

$$h(\mathbf{y}) = \frac{1}{z} \prod_{y \in \mathbf{y}} \beta(y) \prod_{u,v \in \mathbf{y},\, u \sim v} \gamma(u, v), \tag{3.12}$$

where z is the normalizing constant, $\beta : \mathcal{S} \mapsto [0, \infty[$ the intensity function and $\gamma : \mathcal{S} \times \mathcal{S} \mapsto [0, \infty[$ the pair interaction function.

The Strauss Point Process is one instance of pairwise interaction process. Its density is given as:

$$h(\mathbf{y}) \propto \beta^{n(\mathbf{y})} \gamma^{s(\mathbf{y})}, \tag{3.13}$$

where $\beta > 0$ is an intensity parameter, and $s(\mathbf{y})$ the number of pairs of points in $\mathbf{y}$ that are at a distance r or less apart.

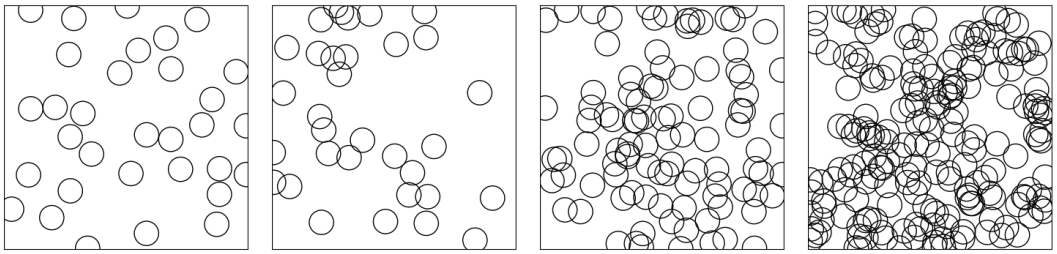
Depending on the value of parameter γ, the process exhibits different behaviors illustrated in Figure 3.2 :

- $\gamma = 0$: points cannot be closer than distance r. This is called a *hardcore* process.
- $0 < \gamma < 1$: the points are repulsive to each other.
- $\gamma = 1$: this is the Poisson process of intensity $\beta\lambda$.
- $\gamma > 1$: the density is not integrable, the Point Process is not defined. Introducing a limit on the number of points $n_{\max}$ such that $h(\mathbf{y}) \propto \mathbb{1}\{n(\mathbf{y}) \leq n_{\max}\}\beta^{n(\mathbf{y})}\gamma^{s(\mathbf{y})}$, allows defining this process for which the points are attractive.

3.1.3 Markov marked Point Processes

Marked Point Processes are Point Processes for which a mark is attached to each point. As such, they are useful for applications where each point or event has some properties attached to it. For instance marks can encode the radius of a point, or combining radius with height: a cylinder. The mark can also carry semantic information: the type of object (e.g. species of trees), or multiplicity (if one point can represent several events). All these descriptors can be combined to enrich the Point Process.

Definition 3.1.8. Let $(\mathcal{S}, d)$ and $(\mathcal{M}, d')$ complete, separable metric spaces, ν a finite Borel measure on $\mathcal{S}$, η a probability distribution on the sigma-algebra of $\mathcal{M}$, and $f_{\nu \times \eta}$ the density of a Poisson process on $\mathcal{S} \times \mathcal{M}$.



(a) $\gamma = 0$ **(b)** $\gamma = 0.1$ **(c)** $\gamma = 1$ **(d)** $\gamma = 1.1$ ($n_{\max} = 200$)

Figure 3.2: Realizations of the Strauss Point Process for several values of γ with $\beta = 1$. The underlying homogeneous Poisson process intensity over $\mathcal{S} = [0, 1]^2$ is $\lambda = 100$. The radius of the circles represent half the interaction distance r.

Let Φ be a marked Point Process with positions in $\mathcal{S}$ and marks in $\mathcal{M}$ specified by a density h w.r.t. $f_{\nu\times\eta}$. Then Φ is a *Markov marked Point Process* w.r.t. the symmetric reflexive relation $\sim$ on $\mathcal{S} \times \mathcal{M}$ if for all $\mathbf{y}$ such that $f(\mathbf{y}) > 0$,

1. $f(\mathbf{x}) > 0$ for all $\mathbf{x} \subseteq \mathbf{y}$;
2. for all $(u, l) \in \mathcal{S} \times \mathcal{M}$, $f(\mathbf{y} \cup \{(u, l)\})/f(\mathbf{y})$ depends only on (u, l) and its neighborhood $\partial^{\mathbf{y}}_{\{(u,l)\}}$.

The Hammersley-Clifford theorem remains valid for Markov marked Point Processes.

Example: Marked pairwise interaction process. We consider a *marked pairwise interaction process* on $\mathcal{S} \times \mathcal{M}$, with $\mathcal{S}$ a compact window in $\mathbb{R}^2$ and $\mathcal{M} = [0, r_{\max}]$ a mark representing a radius, bounded by $r_{\max}$:

$$h(\mathbf{y}) = \prod_{(y,k)\in\mathbf{y}} \beta \prod_{(u,k),(v,l)\in\mathbf{y}} \gamma(||u - v|| - k - l), \tag{3.14}$$

for some intensity $\beta_k > 0$ and measurable interaction function $\gamma : [0, \infty[\mapsto [0, \infty[$. If $\gamma(r) = 1$ for $r > r_0$ for some permeability distance r_0 else $\gamma(r) = 0$, the process is Markov with respect to the relation

$$(u, k) \sim (v, l) \Leftrightarrow ||u - v|| - k - l \leq r_0. \tag{3.15}$$

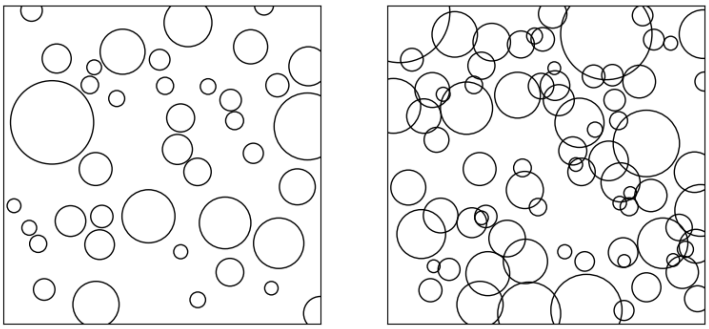
We show the effect of the permeability parameters r_0 on some realizations in Figure 3.3.

3.1.4 Stability conditions

As we have seen previously we can build the density of the Point Process f from an unnormalized density h. This density h needs to be normalizable w.r.t. the reference Point Process. While this can prove tedious, the Ruelle condition (Ruelle, 1970) allows us to simply bound the density to make sure it is normalizable.

Condition 3.1.1. A Point Process specified by an unnormalized density h w.r.t. the measure μ of the Poisson process is *Ruelle-stable* if there exist $M \leq 1$ such that

$$h(\mathbf{y}) \leq M^{n(\mathbf{y})}, \forall \mathbf{y} \in \mathcal{Y}. \tag{3.16}$$



(a) $r_0 = 0$ **(b)** $r_0 = -0.05$

Figure 3.3: Realizations of the marked pairwise interaction process from (3.15) on $\mathcal{S} \times \mathcal{M} = [0,1]^2 \times [0.02, 0.2]$, with $\beta = 1$. The mark corresponds to the radius of each circle.

This condition is enough to ensure that h can be normalized:

$$\int_{\mathbf{y} \in \mathcal{Y}} h(\mathbf{y}) d\mu(\mathbf{y}) \leq \sum_{n=0}^{\infty} \frac{M^n \nu(\mathcal{S})}{n!} = \exp(M\nu(\mathcal{S})). \tag{3.17}$$

Secondly, the Monte Carlo Markov Chain sampling (see Section 3.3.2) requires the following condition:

Condition 3.1.2. A Point Process derived from an unnormalized density h w.r.t. the measure μ of the Poisson process is *locally stable* if there exist $M \in \mathbb{R}$ such that

$$h(\mathbf{y} \cup \{y\}) \leq M h(\mathbf{y}), \forall \mathbf{y} \in \mathcal{Y}, \forall u \in \mathcal{S}. \tag{3.18}$$

Note that Condition 3.1.2 implies Condition 3.1.1.

3.2 Point Process for object detection

Up to now, we settled the fundamentals of Point Processes, however our goal remains to perform object detection in an image. How can we use the Point Process to model our objects both with their geometrical properties, and their relation to the image ?

First, we define configurations such that they represent correctly the objects of interest, where the point position informs on the location of the object (in space, but also in time if necessary), and marks add geometrical or semantic information. Tree crowns can be represented as Point Processes of circles (Perrin et al., 2004), road networks as segments (Lacoste et al., 2005) or buildings as rectangles (Ortner et al., 2008).

Priors on the interactions (and on the geometry of object themselves) are added; penalizing overlaps, forbidding some configurations and favoring others. We denote $h_{\text{priors}}(\mathbf{y})$ the priors' density, also referred as *internal energy* when using potentials.

Then we take into account the image data. The Bayesian approach is to compute the *likelihood* of the observation (or image) $\mathbf{X}$ given the configuration $\mathbf{y}$ (Rue & Hurn, 1999; Perrin et al., 2004):

$$h_{\text{data}}(\mathbf{y}) = h_{\text{data}}(\mathbf{X}|\mathbf{y}) = \prod_{p \mathcal{P}_{\mathbf{X}}} g(\mathbf{X}[\rho] | s_\rho(\mathbf{y})), \tag{3.19}$$

where $g(\mathbf{X}_\rho|\mathbf{y}_p x)$ if the likelihood of pixel value $\mathbf{X}[\rho]$ given the silhouette of the Point Process $s_\rho(\mathbf{y})$ at ρ. However, such approach requires both a foreground and background model to compute g. Thus, in cases where the background proves to be more complex (presence of other objects, clutter etc...), the approach is to use data terms for each object of configuration $\mathbf{y}$ as an *external field*. For instance one can evaluate the homogeneity of the image within the object boundary and its contrast with nearby background (Descombes, 2017; Schmidt et al., 2017; Lacoste et al., 2005; Ortner, 2004):

$$h_{\text{data}}(\mathbf{y}) = \exp\left(-\sum_{y\in\mathbf{y}} V_{\text{data}}(y,\mathbf{X})\right). \tag{3.20}$$

Each potential gets lower as the object better fits the image. With this non-Bayesian approach we risk overlapping objects by accumulating low potentials, thus high penalization on overlaps is needed.

Finally, we get:

$$f(\mathbf{y}) \propto h_{\text{data}}(\mathbf{y}) h_{\text{priors}}(\mathbf{y}). \tag{3.21}$$

In the Bayesian case (image model using the likelihood (3.19)) we get the posterior density $f(\mathbf{y}|\mathbf{X})$ instead of $f(\mathbf{y})$.

At inference, one needs to sample the configuration $\mathbf{y}^*$ that maximizes the density $h(\mathbf{y})$ (or minimizes the energy U for a Gibbs model):

$$\mathbf{y}^* = \arg\max_{\mathbf{y}\in\mathcal{Y}} h(\mathbf{y}). \tag{3.22}$$

3.2.1 Energy model

In the following, we define our density h through an energy $U(\mathbf{y},\mathbf{X})$ function of the configuration $\mathbf{y}$ and image $\mathbf{X}$ such that $h(\mathbf{y}) = \exp(-U(\mathbf{y},\mathbf{X}))$. One way to build U is to define potentials for each object interactions (Ortner, 2004; Crăciun, 2015):

$$U(\mathbf{y},\mathbf{X}) = \sum_{y\in\mathbf{y}} V_{\text{ext.}}(y,\mathbf{X}) + \sum_{\substack{y,y'\in\mathbf{y}\\ y\sim y'}} v_{\text{int.}}(y,y'). \tag{3.23}$$

In our work we rather use the following model, the choice of which we explain in Section 4.1:

$$U(\mathbf{y},\mathbf{X}) = \sum_{y\in\mathbf{y}} V\left(y,\mathbf{X},\mathcal{N}^{\mathbf{y}}_{\{y\}}\right) \tag{3.24}$$

$$= \sum_{y\in\mathbf{y}}\sum_{e\in\xi} V_e\left(y,\mathbf{X},\mathcal{N}^{\mathbf{y}}_{\{y\}}\right), \tag{3.25}$$

where V is the energy of a point composed of energy terms V_e, ξ being the set of energy terms. We purposefully introduce a new neighborhood notation $\mathcal{N}^{\mathbf{y}}_{\{y\}}$ as opposed to $\partial^{\mathbf{y}}_{\{y\}}$, as with this model, the object interaction relation and Markovianity relation may differ.

3.2.1.1 Markovianity relation

We denote $\underset{n}{\sim}$ the interaction relation $y \underset{n}{\sim} y' \Leftrightarrow ||y-y'|| < d_{max}$ resulting in neighborhood $\mathcal{N}^{\mathbf{y}}_{\{y\}}$. We need to determine the relation $\underset{m}{\sim}$ (with associated neighborhood $\partial^{\mathbf{y}}_{\{y\}}$) for which the Point

Process in Markovian. Given Definition 3.1.6, the dependency of ratio $h(\mathbf{y} \cup \{u\})/h(\mathbf{y})$ will give us the minimal requirement for $\underset{m}{\sim}$ (i.e. the most restrictive relation that still ensures Markovianity of the process):

$$\begin{aligned}\frac{h(\mathbf{y} \cup \{u\})}{h(\mathbf{y})} &= \exp\left(U(\mathbf{y}, \mathbf{X}) - U(\mathbf{y} \cup \{u\})\right) \\ &= \exp\left(\sum_{y \in \mathbf{y}} \left[V\left(y, \mathbf{X}, \mathcal{N}^{\mathbf{y}}_{\{y\}}\right) - V\left(y, \mathbf{X}, \mathcal{N}^{\mathbf{y} \cup \{u\}}_{\{y\}}\right)\right] - V\left(u, \mathbf{X}, \mathcal{N}^{\mathbf{y} \cup \{u\}}_{\{u\}}\right)\right) \\ &= \exp\left(\sum_{y \in \mathcal{N}^{\mathbf{y}}_{\{u\}}} \left[V\left(y, \mathbf{X}, \mathcal{N}^{\mathbf{y}}_{\{y\}}\right) - V\left(y, \mathbf{X}, \mathcal{N}^{\mathbf{y} \cup \{u\}}_{\{y\}}\right)\right] - V\left(u, \mathbf{X}, \mathcal{N}^{\mathbf{y}}_{\{u\}}\right)\right)\end{aligned}$$

using the fact that $y \notin \mathcal{N}^{\mathbf{y}}_{\{u\}} \implies \mathcal{N}^{\mathbf{y}}_{\{y\}} = \mathcal{N}^{\mathbf{y} \cup \{u\}}_{\{y\}}$ (as $\underset{n}{\sim}$ is symmetric). This shows the density ratio depends on u, and the first and second degree neighbors of u. Defining the relation for Markovianity such as $y \underset{m}{\sim} y' \Leftrightarrow \exists y'' \in \mathbf{y}, y \underset{n}{\sim} y'' \underset{n}{\sim} y'$ is impractical and depends on $\mathbf{y}$; with $\underset{n}{\sim}$ derived from d_{max}, the following is sufficient to have Markovianity w.r.t. relation $\underset{m}{\sim}$:

$$y \underset{m}{\sim} y' \Leftrightarrow ||y - y'|| < 2d_{max}. \tag{3.26}$$

3.2.1.2 Local stability

To build converging simulations of our Point Process, we need to check for Condition 3.1.2; i.e. we want to find $M \in \mathbb{R}$ such that $h(\mathbf{y} \cup \{u\})/h(\mathbf{y}) \leq M$.

We suppose all V_e, $e \in \xi$ are built such that:

$$\exists A > 0 \, |V_e(y, \mathbf{X}, \mathcal{N}^{\mathbf{y}}_{\{y\}})| < A, \, \forall y \in \mathbf{y}, \, \forall \mathbf{y} \in \mathcal{Y}, \, \forall e \in \xi. \tag{3.27}$$

Thus, $|V(y, \mathbf{X}, \mathcal{N}^{\mathbf{y}}_{\{y\}})| < A|\xi|$. From the previous results on the Markovianity relation we have that:

$$\frac{h(\mathbf{y} \cup \{u\})}{h(\mathbf{y})} \leq \exp\left(2A|\xi| \left|\mathcal{N}^{\mathbf{y}}_{\{y\}}\right| + A|\xi|\right). \tag{3.28}$$

We are left to find an upper bound for the number of neighbors. (Ortner, 2004) use an exclusion energy that effectively forbids configurations with a number of neighbors above a certain threshold. In our case we use a limit on objects per cell (see Section 3.3.2.4 and 5.1.2 on parallelization) $n_{c,max}$. In turn this sets the maximum number of neighbors a point can get up to $4n_{c,max} - 1$ (the worst case being a point in the corner of a cell with neighbors in the 3 neighboring cells of that corner, as illustrated in Figure 3.4). Finally, we show we meet the Condition 3.1.2 as such:

$$\frac{h(\mathbf{y} \cup \{u\})}{h(\mathbf{y})} \leq M, \; M = \exp\left(A|\xi|(8n_{c,max} - 1)\right). \tag{3.29}$$

3.3 Sampling the Point Process

3.3.1 Markov chains

To sample configurations from the Point Process, we need to build a Markov chain that converges towards the Point Process distribution.

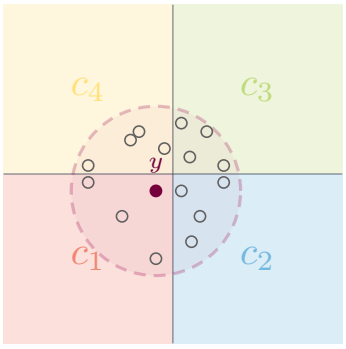


Figure 3.4: Edge case for maximum number of neighbors. Current point is labeled y with its neighborhood represented as a large dotted circle. Other points in $\mathbf{y}$ are represented as small hollow circles. In this illustration $n_{c,max} = 4$ for all cells c.

Definition 3.3.1. Let (Y_t) be a sequence of random variables, with values taken from a space $\mathcal{Y}$ associated with as sigma-algebra $\mathcal{B}$. (Y_n) is a **Markov chain** if

$$p(Y_{t+1} \in A | Y_t = \mathbf{y}_t, \ldots, Y_0 = \mathbf{y}_0) = p(Y_{t+1} \in A | Y_t = \mathbf{y}_t), \forall A \in \mathcal{B}. \tag{3.30}$$

In short, the evolution of the Markov chain only depends on the current state. We will consider homogeneous Markov chains, defined as such:

Definition 3.3.2. A Markov chain is **homogeneous** if the evolution does not depend on the time parameter t.

Definition 3.3.3. A **transition kernel** is a function K defined on $\mathcal{Y} \times \mathcal{B}$ such that:

- $\forall x \in \Omega,\ K(x, \cdot)$ is a probability measure.
- $\forall A \in \mathcal{B},\ K(\cdot, A)$ is measurable.

The transition kernel K for Markov chain $(Y_t)_{t \in \mathbb{N}}$ is such that

$$K(\mathbf{y}, A) = p(Y_{t+1} \in A | Y_t = \mathbf{y}). \tag{3.31}$$

The Markov chain needs to meet some other properties in order to use it for sampling:

- **Stationary**: a measure π is stationary for Markov chain (Y_t) with transition kernel K if:

$$\pi(A) = \int K(\mathbf{y}, A)\pi(d\mathbf{y}), \forall A \in \mathcal{B}. \tag{3.32}$$

- **Reversible**: a Markov chain is reversible if its transition kernel K is such as

$$\int_A K(\mathbf{y}, B)\pi(d\mathbf{y}) = \int_B K(\mathbf{y}', A)\pi(d\mathbf{y}'), \forall A, B \in \mathcal{B}. \tag{3.33}$$

- **Irreducible**: a Markov chain is irreducible if there is a positive probability to reach any $\mathbf{y}$ from any other $\mathbf{y}'$ in a finite number of steps:

$$\forall \mathbf{y}, \mathbf{y}' \in \mathcal{Y}, \exists k < \infty,\ p(Y_{t+k} = \mathbf{y} | Y_t = \mathbf{y}') > 0. \tag{3.34}$$

- **Aperiodic**: a Markov chain is aperiodic if it exhibits no cycles, i.e.:

$$\text{GCD}\left\{d : p(Y_d = \mathbf{y}|Y_0 = \mathbf{y})\right\} = 1, \tag{3.35}$$

  where GCD is the Greatest Common Divisor.

- **Harris recurrent**: a Markov chain is Harris recurrent, if for any $A \in \mathcal{B}$ such that $\pi(A) > 0$,

$$p(\exists t : y_t \in A|Y_0 = \mathbf{y}) = 1, \forall \mathbf{y} \in \mathcal{Y}. \tag{3.36}$$

- **Ergodic**: an aperiodic Harris recurrent Markov chain is ergodic and converges towards π:

$$\left\|K^t(x, \cdot) - \pi(\cdot)\right\| \xrightarrow[t\to\infty]{} 0, \forall x \in \mathcal{Y}, \tag{3.37}$$

  with $\|\cdot\|$ such as $\|\mu_1 - \mu_2\| = \sup_A |\mu_1(A) - \mu_2(A)|$.

3.3.2 Reversible Jump Monte Carlo Markov chain

(Green, 1995) build this ergodic Markov chain with a mixture of perturbation kernels $Q_m(\mathbf{y}, A)$, $m \in \mathcal{Q}$ such that :

$$Q(\mathbf{y}, A) = \sum_{m\in\mathcal{Q}} Q_m(\mathbf{y}, A). \tag{3.38}$$

It is named Reversible Jump Monte Carlo Markov chain (RJMCMC). For a given state of the Markov Chain $Y_t = \mathbf{y}$, the Metropolis-Hastings update scheme goes as follows:

1. With probability $Q_m(\mathbf{y}, \mathcal{Y})$ choose a kernel Q_m or with probability $1 - \sum_m Q_m(\mathbf{y}, \mathcal{Y})$ let the state unchanged.
2. Simulate $\mathbf{y}'$ with the normalized kernel :

$$\mathbf{y}' \sim \frac{Q_m(\mathbf{y}, \cdot)}{Q_m(\mathbf{y}, \mathcal{Y})}. \tag{3.39}$$

3. Compute the Green ratio $r(\mathbf{y}, \mathbf{y}')$, see (3.47).
4. Accept perturbation from $\mathbf{y}$ to $\mathbf{y}'$ with probability $\alpha(\mathbf{y}, \mathbf{y}') = \min(1, r(\mathbf{y}, \mathbf{y}'))$

As such the *transition* kernel K for the Markov chain is a function of the *perturbation* kernel Q and the acceptance probability α.

Thus, to be able to perform such a procedure, each kernel Q_m needs to have the following properties:

1. We must know the probability of picking Q_m; $Q_m(\mathbf{y}, \mathcal{Y})$.
2. The probabilities of picking each kernel sum to at most 1; $\sum_{m\in\mathcal{Q}} Q_m(\mathbf{y}, \mathcal{Y}) \leq 1$, $\forall \mathbf{y} \in \mathcal{Y}$
3. We can sample state perturbations from the normalized kernel density $Q_m(\mathbf{y}, .)/Q_m(\mathbf{y}, \mathcal{Y})$ for any configuration $\mathbf{y} \in \mathcal{Y}$.

4. There exists a symmetric measure $\psi_m(d\mathbf{y}, d\mathbf{y}')$, such that $\pi(d\mathbf{y})Q_m(\mathbf{y}, d\mathbf{y}')$ is absolutely continuous w.r.t. $\psi_m(d\mathbf{y}, d\mathbf{y}')$. The associated Radon-Nikodym derivative is denoted D_m;

$$D_m(\mathbf{y}, \mathbf{y}') = \frac{\pi(d\mathbf{y})Q_m(\mathbf{y}, d\mathbf{y}')}{\psi_m(d\mathbf{y}, d\mathbf{y}')}. \tag{3.40}$$

3.3.2.1 Detailed Balance

The last condition above is here to ensure the reversibility condition of the Markov Chain is met (implying that π is the stationary measure of the Markov Chain Y_t). For the chain to be reversible, its probability to go from a set $A \in \mathcal{B}$ towards a set $B \in \mathcal{B}$ needs to be the same at going from B to A (see (3.33)). This is known as *detailed balance* (DB) condition. As the kernel Q is a sum of kernels Q_m, we only need to show DB is maintained for each kernel Q_m (Green, 1995; Ortner, 2004). The transition sub-kernel for Q_m is

$$K_m(\mathbf{y}, A) = \int_A Q_m(\mathbf{y}, d\mathbf{y}')\alpha(\mathbf{y}, \mathbf{y}'). \tag{3.41}$$

To verify DB we then need :

$$\int_A \int_B Q_m(\mathbf{y}, d\mathbf{y}')\alpha(\mathbf{y}, \mathbf{y}')\pi(d\mathbf{y}) = \int_A \int_B Q_m(\mathbf{y}', d\mathbf{y})\alpha(\mathbf{y}', \mathbf{y})\pi(d\mathbf{y}'). \tag{3.42}$$

To find the function α that verifies this equation, (Green, 1995) proposes to find a symmetric measure ψ_m on $\mathcal{Y} \times \mathcal{Y}$, such that πQ_m is *absolutely continuous* w.r.t. ψ_m; i.e.:

$$\psi_m(A, B) = 0 \implies \pi Q_m(A, B). \tag{3.43}$$

The **Radon-Nikodym theorem** then tells us a unique function D_m exists called the *Radon-Nikodym* derivative (or density) of πQ_m w.r.t. ψ_m, such that :

$$\int_A Q_m(\mathbf{y}, B)\pi(d\mathbf{y}) = \int_A \int_B D_m(\mathbf{y}, \mathbf{y}')\psi_m(d\mathbf{y}, d\mathbf{y}'). \tag{3.44}$$

Then we can replace πQ_m in (3.42) to get:

$$\int_A \int_B \alpha(\mathbf{y}, \mathbf{y}')D_m(\mathbf{y}, \mathbf{y}')\psi_m(d\mathbf{y}, d\mathbf{y}') = \int_A \int_B \alpha(\mathbf{y}', \mathbf{y})D_m(\mathbf{y}', \mathbf{y})\psi_m(d\mathbf{y}', d\mathbf{y}). \tag{3.45}$$

Since we built ψ_m to be symmetric, we only need:

$$\alpha(\mathbf{y}, \mathbf{y}')D_m(\mathbf{y}, \mathbf{y}') = \alpha(\mathbf{y}', \mathbf{y})D_m(\mathbf{y}', \mathbf{y}). \tag{3.46}$$

(Green, 1995) proposes to take:

$$\alpha(\mathbf{y}, \mathbf{y}') = \min\left\{1, \frac{D(\mathbf{y}, \mathbf{y}')}{D(\mathbf{y}', \mathbf{y})}\right\}. \tag{3.47}$$

3.3.2.2 Balancing in terms of random numbers

In more recent works (Green & Hastie, 2009) propose an alternative derivation of α for the Markov Chain in order to maintain detailed balance.

The perturbation is defined as follows: From current configuration $\mathbf{y}$, choose deterministic function τ_m with probability $j_m(\mathbf{y})$. It generates the new proposed state $\mathbf{x}$ as $(\mathbf{x}, \delta') = \tau_m(\mathbf{y}, \delta)$, where δ is an r-dimentional random number to generate $\mathbf{x}$ from $\mathbf{y}$, and δ' the r'-dimentional random number used to generate $\mathbf{y}$ from $\mathbf{x}$ via τ', the reverse transform from τ_m (i.e. $\tau'_m(\mathbf{x}, \delta') = (\mathbf{y}, \delta)$). Random numbers δ and δ' are respectively generated with known densities g and g'. Move from $\mathbf{y}$ to $\mathbf{x}$ accepted with probability $\alpha(\mathbf{y}, \mathbf{x})$. The Detailed Balance is them given as:

$$\int_{\mathbf{y},\mathbf{x}\in A\times B} h(\mathbf{y})j_m(\mathbf{y})g(\delta)\alpha(\mathbf{y},\mathbf{x})d\mathbf{y}d\delta = \int_{\mathbf{y},\mathbf{x}\in A\times B} h(\mathbf{x})j_m(\mathbf{x})g'(\delta')\alpha(\mathbf{x},\mathbf{y})d\mathbf{x}d\delta'. \tag{3.48}$$

If transform τ_m from $(\mathbf{y}, \delta)$ to $(\mathbf{x}, \delta')$ and τ'_m are differentiable, we can apply the change-of-variable formula to the right-hand side of the equation. The equation now holds if

$$h(\mathbf{y})j_m(\mathbf{y})g(\delta)\alpha(\mathbf{y},\mathbf{x}) = h(\mathbf{x})j_m(\mathbf{x})g'(\delta')\alpha(\mathbf{x},\mathbf{y})\left|\frac{\partial(\mathbf{x},\delta')}{\partial(\mathbf{y},\delta)}\right|. \tag{3.49}$$

A valid choice for α is then:

$$\alpha(\mathbf{y},\mathbf{x}) = \min\left\{1, \frac{h(\mathbf{x})j_m(\mathbf{x})g'(\delta')}{h(\mathbf{y})j_m(\mathbf{y})g(\delta)}\left|\frac{\partial(\mathbf{x},\delta')}{\partial(\mathbf{y},\delta)}\right|\right\}, \tag{3.50}$$

with $\left|\frac{\partial(\mathbf{x},\delta')}{\partial(\mathbf{y},\delta)}\right|$ the determinant of the Jacobian matrix of τ_m.

This approach works across dimensions, provided that the transformation from $(\mathbf{y}, \delta)$ to $(\mathbf{x}, \delta')$ remains a diffeomorphism (Green & Hastie, 2009). Denoting with n and n' the dimensions $\mathbf{y}$ and $\mathbf{x}$ respectively, the diffeomorphism requires that $n + r = n' + r'$. Note that either one or both of r or r' might be 0.

3.3.2.3 Perturbation kernels

Here we (re)build the basic perturbation kernels for the RJMCMC. Note that the birth and death kernel is the minimal requirement for the Markov chain to converge to the stationary distribution. The demonstration for the birth and death kernel is sourced from (Geyer & Møller, 1994; Lacoste, 2004; Ortner, 2004; Crăciun, 2015).

Birth and Death. The birth and death kernel adds and removes point in the configuration. It is essential as it allows moving across dimensions in $\mathcal{Y}$. This kernel itself is a mix of two sub kernels: a birth kernel (that adds new points) and a death kernel (that removes points). The most basic birth and death kernel proposes new points y with density $\nu(\cdot)/\nu(\mathcal{S})$ and removes points by picking one uniformly in $\mathbf{y}$. The resulting kernel is as follows, with p_B and p_D the probabilities to pick birth or death respectively:

$$Q(\mathbf{y},\mathbf{y}') = p_\mathrm{B}(\mathbf{y})Q_\mathrm{B}(\mathbf{y},\mathbf{y}') + p_\mathrm{D}Q_\mathrm{D}(\mathbf{y},\mathbf{y}') \tag{3.51}$$

Both kernels are defined as:

$$Q_\mathrm{B}(\mathbf{y}, A) = \int_{y\in\mathcal{S}} \mathbb{1}_A(\mathbf{y}\cup\{y\})\frac{\nu(dy)}{|\mathcal{S}|}, \tag{3.52}$$

$$Q_\mathrm{D}(\mathbf{y}, A) = \sum_{y\in\mathbf{y}} \mathbb{1}_A(\mathbf{y}\setminus\{y\})\frac{1}{n(\mathbf{y})}. \tag{3.53}$$

We now have to find the symmetric measure on $\mathcal{Y} \times \mathcal{Y}$. Note that both parts of the kernel only move between $\mathcal{Y}_n$ to $\mathcal{Y}_{n+1}$, thus we can restrict the study to subsets $A_n \subseteq \mathcal{Y}_n$ and $B_{n+1} \subseteq \mathcal{Y}_{n+1}$. We can then write measures $\pi Q(A_n, B_{n+1})$ and $\pi Q(B_{n+1}, A_n)$, with μ the density of the reference Poisson Point Process

$$\int_{A_n} Q(\mathbf{y}, B_{n+1})\pi(d\mathbf{y}) = p_{\mathrm{B}} \int_{A_n} \left(\int_{y \in \mathcal{S}} \mathbb{1}_{B_{n+1}}(\mathbf{y} \cup \{y\}) \frac{\nu(dy)}{|\mathcal{S}|} \right) f(\mathbf{y})\mu(d\mathbf{y}) \tag{3.54}$$

$$\int_{B_{n+1}} Q(\mathbf{y}, A_n)\pi(d\mathbf{y}) = p_{\mathrm{D}} \int_{B_{n+1}} \left(\sum_{y \in \mathbf{y}} \mathbb{1}_{A_n}(\mathbf{y} \setminus \{y\}) \frac{1}{n(\mathbf{y})} \right) f(\mathbf{y})\mu(d\mathbf{y}) \tag{3.55}$$

Splitting ψ_n into ψ_n^+ and ψ_n^- respectively defined on $\mathcal{Y}_n \times \mathcal{Y}_{n+1}$ and $\mathcal{Y}_{n+1} \times \mathcal{Y}_n$, we define

$$\psi_n^+(A_n, B_{n+1}) = \int_{A_n} \int_{y \in \mathcal{S}} \mathbb{1}_{B_{n+1}}(\mathbf{y} \cup \{y\})\nu(dy)\mu(d\mathbf{y}) \tag{3.56}$$

$$\psi_n^-(B_{n+1}, A_n) = \int_{B_{n+1}} \sum_{y \in \mathbf{y}} \mathbb{1}_{A_n}(\mathbf{y} \setminus \{y\})\mu(d\mathbf{y}) \tag{3.57}$$

We show the symmetry of ψ_n^+ and ψ_n^- by decomposing the Poisson process measure μ using (3.4):

$$\begin{aligned}
\psi_n^-(B_{n+1}, A_n) &= \frac{\exp(-\nu(\mathcal{S}))}{(n+1)!} \int_{\mathcal{S}^{n+1}} \sum_{y \in \mathbf{y}} \mathbb{1}_{A_n}(\mathbf{y} \setminus \{y\}) \mathbb{1}_{B_{n+1}}(\mathbf{y}) \nu^{n+1}(\mathbf{y}) \\
&= \frac{\exp(-\nu(\mathcal{S}))}{(n+1)!} \int_{\mathcal{S}^{n+1}} (n+1) \mathbb{1}_{A_n}(\{y^1, \dots, y^n\}) \mathbb{1}_{B_{n+1}}(\{y^1, \dots, y^{n+1}\}) \nu^{n+1}(\mathbf{y}) \\
&= \frac{\exp(-\nu(\mathcal{S}))}{n!} \int_{\mathcal{S}^n} \int_{\mathcal{S}} \mathbb{1}_{B_{n+1}}(\mathbf{x}) \mathbb{1}_{A_n}(\mathbf{x} \cup \{u\}) \nu^n(d\mathbf{x}) \nu(du) \\
&= \psi_n^+(A_n, B_{n+1}).
\end{aligned}$$

Then we can define ψ on $\mathcal{Y} \times \mathcal{Y}$; more specifically on we define ψ on $\bigcup_{n=0}^{\infty} \{\{\mathcal{Y}_n \times \mathcal{Y}_{n+1}\} \cup \{\mathcal{Y}_{n+1} \times \mathcal{Y}_n\}\}$: On each $\{\mathcal{Y}_n \times \mathcal{Y}_{n+1}\} \cup \{\mathcal{Y}_{n+1} \times \mathcal{Y}_n\}$, ψ is equal to ψ_n. We have $\psi(A, B) = 0 \implies \pi Q(A, B) = 0$. And the Radon-Nikodym derivatives given for birth and death (deduced respectively from (3.54), (3.56) and (3.55), (3.57)), are respectively:

$$D(\mathbf{y}, \mathbf{y} \cup \{u\}) = \frac{p_{\mathrm{B}} f(\mathbf{y})}{\nu(\mathcal{S})}, \tag{3.58}$$

$$D(\mathbf{y}, \mathbf{y} \setminus \{u\}) = \frac{p_{\mathrm{D}} f(\mathbf{y})}{n(\mathbf{y})}. \tag{3.59}$$

The associated Green ratios for birth and death are then

$$r(\mathbf{y}, \mathbf{y} \cup \{u\}) = \frac{p_{\mathrm{D}}}{p_{\mathrm{B}}} \frac{\nu(\mathcal{S})}{n(\mathbf{y}) + 1} \frac{h(\mathbf{y} \cup \{u\})}{h(\mathbf{y})}, \tag{3.60}$$

$$r(\mathbf{y}, \mathbf{y} \setminus \{u\}) = \frac{p_{\mathrm{B}}}{p_{\mathrm{D}}} \frac{n(\mathbf{y})}{\nu(\mathcal{S})} \frac{h(\mathbf{y} \setminus \{u\})}{h(\mathbf{y})}, \tag{3.61}$$

with h the unnormalized density of the process.

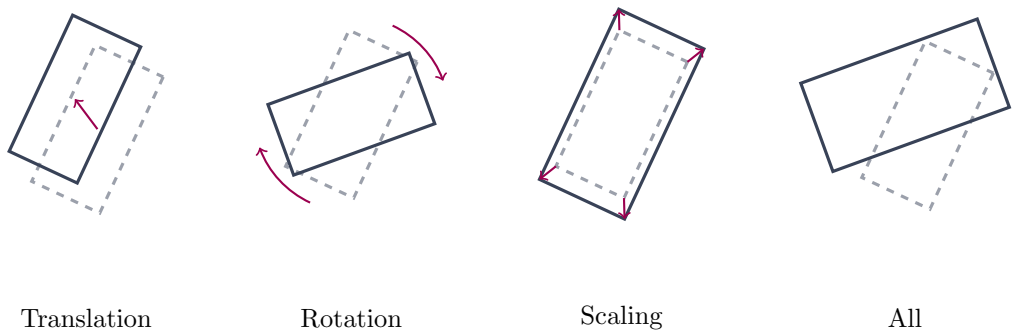


Figure 3.5: Examples of local perturbations. "All" combines all transforms at once.

Local perturbation kernel. This perturbation does not change the dimension of the current state, the number of points is fixed. It modifies the parameters of an object, for instance translating, rotating or scaling as illustrated in Figure 3.5.

While (Ortner, 2004; Crăciun, 2015) use the symmetric measure approach to derive acceptance probability α, we here use the formalism from (Green & Hastie, 2009) described in Section 3.3.2.2 to show the same result.

We consider a Marked Point Process with $\dim(\mathcal{M})$ marks (e.g. 3 for a rectangle or ellipse) and denote $d = \dim(\mathcal{M}) + \dim(\mathcal{S})$ the dimension of each point (e.g. $d = 5$ for rectangles in 2D space). We use the random number formalism as described in Section 3.3.2.2. The kernel to transition from $\mathbf{y}$ to $\mathbf{x}$ works as follows:

1. The transform kernel is picked with probability p_{T}.
2. The kth object y is picked uniformly in $\mathbf{y}$
3. Generate a perturbation $\delta \in \mathbb{R}^{d+2}$
4. Proposed configuration is defined as:

$$(\mathbf{x}, \delta') = \tau_k(\mathbf{y}, \delta) = (\{y^1, \ldots, y^k + \delta, \ldots\}, \delta), \tag{3.62}$$

thus $\tau_k(\mathbf{x}, \delta') = (\{x^1, \ldots, x^k - \delta', \ldots\}, \delta')$: then $\delta' = \delta$.

Here we consider a set of $n(\mathbf{y})$ sub kernels, each one modifying the kth object. Each sub-kernel is picked with probability $j_k(\mathbf{y}) = j_k(\mathbf{x}) = \frac{p_{\mathrm{T}}}{n(\mathbf{y})}$. The density of random variable δ is $g(\delta)$ and the one for δ' is $g'(\delta') = g'(\delta) = g(-\delta)$. Then Equation (3.50) gives us:

$$\alpha(\mathbf{y}, \mathbf{x}) = \min\left\{1, \frac{h(\mathbf{x})g(-\delta)}{h(\mathbf{y})g(\delta)} \left|\frac{\partial(\mathbf{x}, \delta')}{\partial(\mathbf{y}, \delta)}\right|\right\} \tag{3.63}$$

The Jacobian is equal to 1, as shown in Appendix A. Thus:

$$\alpha(\mathbf{y}, \mathbf{x}) = \min\left\{1, \frac{h(\mathbf{x})g(-\delta)}{h(\mathbf{y})g(\delta)}\right\}. \tag{3.64}$$

For α to be non-zero for a given δ, we need $g(-\delta) > 0$; i.e. there exists a way to reverse the move defined by δ. The basic approach is to pick a density symmetric around zero, such that $g(\delta) = g(-\delta)$ (e.g. Gaussian distribution centered in 0); it also simplifies the acceptance probability. The above-described approach is still valid if we pick the object y to transform in $\mathbf{y}$ with a non-uniform distribution, requiring to change $j_k(\mathbf{y})$ and $j_k(\mathbf{x})$ accordingly.

3.3.2.4 Parallel sampling

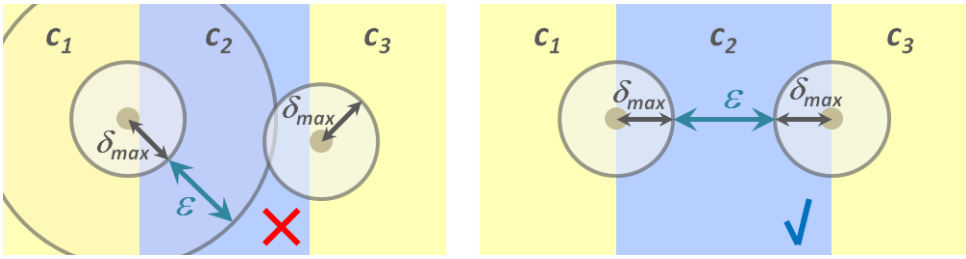


Figure 3.6: Independence of cells. On the left the two cells are not independent as c_2 is not wide enough. On the right cells c_1 and c_3 are independent. *From (Verdié & Lafarge, 2014) (Here ϵ corresponds to $2d_{max}$).*

From the definition of a Markov Point Process (Definition 3.1.6) and the Green ratio (e.g. birth (3.60)), we see that the ratio $h(\mathbf{y}')/h(\mathbf{y})$ only depends on added point y and its neighborhood according to $\sim$. (Verdié & Lafarge, 2014) show that two sequential perturbations for which the neighborhoods do not intersect can be processed in any order (the resulting Green ratio from each move not depending on the other ones), thus can be done in parallel (see Figure 3.6).

For a Point Process derived from an energy $U(\mathbf{y}) = \sum_{y \in \mathbf{y}} V_1(y) + \sum_{y' \sim y} V_2(y, y')$, two cells c and c' are independent if and only if:

$$\min_{u \in c, y \in c'} \|u - y\|_2 \geq d_{max} + 2\delta_{max}, \tag{3.65}$$

with d_{max} the interaction radius between points (for $\sim$) and δ_{max} the maximum translation distance performed by any kernel. Each set of *mutually independent cells* is called a *mic-set*. While it works with a regular grid partitioning of space, (Verdié & Lafarge, 2014) use a data driven quad tree structure to efficiently propose points for non-uniform distributions.

The general transition kernel Q is now a mixture of the usual sub-kernels, restricted to a cell c. Kernel Q_m restricted to c is denoted $Q_{c,m}$:

$$\forall \mathbf{y} \in \mathcal{Y},\ Q(\mathbf{y}, \cdot) = \sum_{c} \sum_{m \in \mathcal{Q}} p_{c,m} Q_{c,m}(\mathbf{y}, \cdot), \tag{3.66}$$

with $p_{c,m}$ the probability to pick cell c and kernel m.

3.3.3 Jump diffusion

Intra-dimensional kernels (i.e. kernels that operate at a fixed dimension) such as the local perturbation kernel (see Section 3.3.2.3) propose transformations randomly to explore nearby configuration of points. However, the transformations are done with no consideration of the energy landscape; in (3.64) the perturbation proposal density g is not tied to the current configuration $\mathbf{y}$ or density h.

In Section 5.1.1 we propose a data driven local transform kernel inspired from (Ortner, 2004; Descombes, 2013). It leverages a pre-computed position energy map to sample perturbation δ. However, this proves costly as it requires producing and sampling from a distribution derived from the position energy map around the current point y (which needs to be extracted from the whole map, and normalized locally). Moreover, it only takes into account a part of the energy

model; interactions are not taken into account, thus it might propose perturbations δ that result in higher energies due to overlaps. Ideally we would use the local gradient of the energy to update y: on one hand, it avoids building a density from a large energy map around the current point at each iteration; the computation is local. On the other we consider the whole energy model, not just a subset of it.

Jump-diffusion is introduced by (Grenander & Miller, 1994) and was successfully applied to object detection and tracking (Srivastava, Grenander, Jensen, & Miller, 2002), segmentation (Han, Tu, & Zhu, 2004), or geometric object extraction (Lafarge et al., 2010) since then.

Jump-diffusion mixes Langevin dynamics (Geman & Hwang, 1986) (also mentioned as diffusion) and the MCMC (Green, 1995). The diffusion explores at fixed number of points, i.e. $\mathcal{Y}_n$, while jumps, passes between subspaces of $\mathcal{Y}$.

The jump process consists of the birth and death kernel of the RJMCMC. Kernels such as the local transform kernel (Section 3.3.2.3) are no longer needed as the diffusion fulfills the same purpose.

The diffusion process differs from the usual MCMC procedure as it does not involve an accept/reject mechanism. For a Point Process such that $h(\mathbf{y}) \propto \exp(-U(\mathbf{y}))$, the following diffusion equation converges towards $\pi(d\mathbf{y})$:

$$d\mathbf{y}_t = -\nabla U(\mathbf{y}_t)dt + W_t, \tag{3.67}$$

where W_t is a Brownian motion. This equation is defined over continuous time, the discrete diffusion, with $t \in \mathbb{N}$ is given by:

$$\mathbf{y}_{t+1} = \mathbf{y}_t + -\gamma\nabla U(\mathbf{y}_t) + \sqrt{2T(t)}w_t,\ w_t \sim \mathcal{N}(0, \gamma), \tag{3.68}$$

where γ is the step size and $T(t)$ serves as a relaxation temperature; at higher T the Brownian motion allows exploring more of $\mathcal{Y}_n$, while at low T the Brownian motion becomes negligible and the diffusion dynamics acts as gradient descent. While the jump-diffusion is usually more effective than the standard MCMC sampler (Descombes, 2013), it requires being able to compute the gradient of the energy U at any configuration $\mathbf{y}$, with the density function being Lipschitz-continuous (Geman & Hwang, 1986).

Definition 3.3.4. A real valued function f is **Lipschitz-continuous** if there exists a positive real constant k such that for all real x_1 and x_2:

$$|f(x_1) - f(x_2)| \leq k|x_1 - x_2|. \tag{3.69}$$

3.3.4 Simulated annealing

The RJMCMC procedure described here will — given the proper conditions — run a Markov chain for which the stationary measure is π. In order to get the best fitting configuration $\mathbf{y}^*$ (see (3.22)) — the configuration that maximizes the posterior density in the case of a Bayesian model — we use *simulated annealing* to gradually collapse the Markov chain stationary distribution into the single global maximum of h.

Simulated annealing consists in running a Markov chain $(\mathbf{y}_t)_{t\in\mathbb{N}}$ with density $h_t(\mathbf{y}) = h^{1/T_t}(\mathbf{y})$ as the temperature T_t decreases to 0. Simulated annealing is loosely based on annealing in metallurgy, where the heating and cooling of a metal can alter its physical properties.

At high temperature the chain can explore more of the configurations space $\mathcal{Y}$, the stationary density gets closer to the uniform density over. At temperatures near 0, only perturbations that increase the density h are accepted. The annealing scheme starts at high temperature to be able to reach all states of $\mathcal{Y}$, then the temperature slowly decreases (as to maintain stability and explore different modes), as the chain reaches the global maximum only small steps are accepted until the stopping condition is reached.

The chain is proved to converge to the global maximum for Reversible Jump MCMC (Stoica, Gregori, & Mateu, 2005) and birth and death process (Robert & Casella, 2004), but for carefully chosen decrease function.

Logarithmic decrease.

$$T_t = \frac{C}{\log(t+1)}. \tag{3.70}$$

This decrease scheme ensures the convergence if the constant C is larger than the deepest local maximum. In practice this decrease scheme is very slow, often time logarithmic decrease is replaced by geometric decrease.

Geometric decrease. Given as:

$$T_{t+1} = \alpha T_t, \tag{3.71}$$

with $\alpha \in [0, 1]$ driving the speed of the decrease; usually α is picked very close to 1. Contrary to the logarithmic scheme, this one is not proven to converge. Lower α yields faster temperature decrease, at the cost of breaking away from the convergence guarantees of low speed temperature decrease.

Others. (Ortner, Descombes, & Zerubia, 2007) propose to use *adaptive* annealing, that adapts the schedule depending on some statistics of the chain at step t. (Guilmeau, Chouzenoux, & Elvira, 2021) propose to mix Fast Simulated Annealing (that modifies the acceptance probability function) and Sequential Monte Carlo (that simulates a weighted population of particles) into a new scheme.

3.3.5 Stopping conditions

In order to get the exact optimal configuration $\mathbf{y}^*$, the RJMCMC needs to run for an infinite amount of steps. In practice, we do not have that much time. For some cases *exact simulations* (Lieshout, 2000) allow to ensure the simulated Markov chain has reached its theoretical optimum. While this might be necessary for some statistical inference tasks, the application to object detection requires some compromises towards lower inference times.

(Robert & Casella, 2004) propose using statistics on the configurations such as

$$\bar{s}_t = \frac{1}{t} \sum_{k=0}^{t} s(\mathbf{y}_t), \tag{3.72}$$

where $s(\mathbf{y}_t)$ is some measurable characteristic of the configuration at step t; e.g. the number of points $n(\mathbf{y}_t)$ or the energy $U(\mathbf{y}_t|\mathbf{X})$. (Lacoste, 2004) compute this empirical mean every N (i.e. average for values $t \equiv 0 \mod N$) as to limit the autocorrelation effect of the Markov chain. Then, if the empirical mean $\bar{s}_t$ does not change more than a set ϵ_s for more than n_{test} sequential computations of $\bar{s}_t$, the chain is stopped. For more stability one can monitor more than one statistic.

One might also monitor the normalized energy change $(U(\mathbf{y}') - U(\mathbf{y}))/U(\mathbf{y})$, the running average of the acceptance ratio (or number of acceptances), and process those as the configurations' statistics above to produce a stopping condition.

For our application we rather set a fixed amount of iterations, although scaling it with the size of the image space $\mathcal{S}$ to explore and the amount of parallelization. While this means we sometimes might run more steps than necessary, it alleviates the running of the numerous parameters involved in using multiple statistics to define a good stopping condition.

(Robert & Casella, 2004) also proposes running parallel chains and computing both inter-chains and intra-chains statistics to monitor convergence.

3.4 CNN fundamentals

Definition 3.4.1. A Convolutional Neural Network (CNN) is a type of artificial neural network that uses convolutions to extract information from data with grid like topology (Goodfellow, Bengio, & Courville, 2016). These models use a combination of *convolutions* and *pooling* operations to extract patterns from the data.

For this approach the data need only be in a grid like structure; while 2D image are the most usual application, it can be generalized to 3D for instance with electron microscopy scans (Hallou, Yevick, Dumitrascu, & Uhlmann, 2021). The grid might also not be isotropic, for instance in image sequence segmentation (LaLonde et al., 2018) the third dimension represents time, and in audio classification (Hershey et al., 2017) one axis of the spectrogram represents time while the other is frequency. Closer to our subject, multispectral satellite images (see Chapter 1) can be treated as 3D data, with two spatial dimensions on the last corresponding to electromagnetic frequency.

The convolution operator applies a cross-correlation of the input image (or tensor) and a kernel, the output is often referred to as a feature map (Goodfellow et al., 2016). In short the cross-correlation of the kernel on the input, returns a signal that can be interpreted as proportional to the presence of the pattern represented by the kernel at every location of the image. The input and output can be composed of one or several channels (also referred to as features): e.g. the color channels for RGB images. Output and input number of features may be different; an input tensor of size (H, W, C_{int}) (height, width, number of features/channels) with a kernel of size $(K, K, C_{\text{int}}, C_{\text{out}})$ will produce a tensor of size (H, W, C_{out}), modulo some padding applied to the input.

The pooling operator reduces the size of the input image or tensor by applying an aggregation function on chunks of the image. For instance for every two-by-two block of pixels, the *Max pooling* operation keeps only the maximum value of the block to produce the output map. This reduces the spatial size of the input, allowing to aggregate spatial information.

The sequential combination of convolutions and pooling allows alternating between pattern extraction and spatial aggregation, so that steps after steps the extracted patterns grow from low level textures to larger features in the image. The surge in popularity of CNN models stems from the ability to learn the convolution kernels for a specific objective (defined by a loss function to minimize). This is enabled by the ability to back-propagate the gradient of from the loss to the kernel parameters that can them be updated within a gradient descent learning scheme.

One such model is the Unet (Ronneberger et al., 2015), it combines convolutions and pooling layers with upsampling (or *up-convolutions*) that allows to regain spatial resolution, in a sense reversing the pooling layers. The *downward path*, alternating convolution blocks with pooling,

progressively extracts bigger size patterns in the image. The *downward path*, using convolutions and upsampling, rebuilds the spatial resolution of the output progressively, with the help of feature maps transferred from the same depth level of the downward path. This simple architecture allows for fine detail outputs (the output is of same spatial resolution of the input), and higher scale pattern matching (thanks to the depth of the network). This makes it a good fit for pixel wise classification or dense feature extraction.

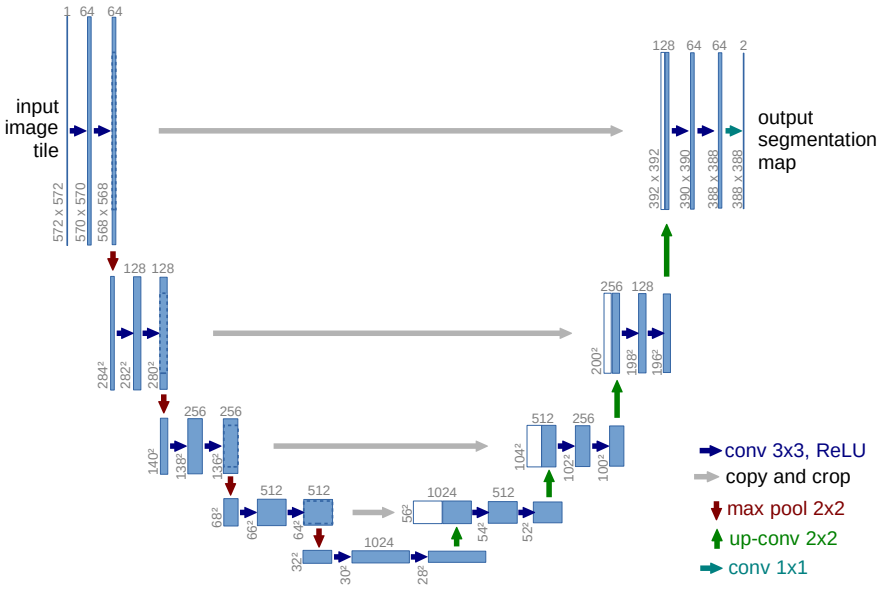


Figure 3.7: Unet architecture. *From (Ronneberger et al., 2015).*

Some deep neural architectures make use of *densely connected layers*;

Definition 3.4.2. A **densely connected** layer, computes an output $\mathbf{B}$ of shape $(\ldots, C_{out})$ from an input $\mathbf{A}$ of shape $(\ldots, C_{in})$ via a matrix multiplication with a weight tensor $\mathbf{W}$ of shape (C_{in}, C_{out}) as $\mathbf{B} = \mathbf{AW}$.

For instance a classical approach in the first image classification networks such as (Krizhevsky et al., 2012) is to flatten the image after the last convolution layer; a feature map of shape (H', W', F) gets flattened to a vector of features of size $(H'W'F)$, with F the number of features. In turn this vector of features is fed through a sequence of dense layers (in our example $C_{in} = H'W'F$) to finally output a vector of class probabilities through a Softmax. First, this architecture sets the network input to be of fixed size; since H' and W' are a direct function of the CNN model and the input image size (H, W). Also, as the processed image gets bigger, the number of elements in the weight $\mathbf{W}$ of the dense layer scales quadratically to the image size; for a bigger image of size (sH, sW), we get weight tensor $\mathbf{W}$ of shape $(s^2H'W'F, C_{out})$.

As satellite images are big and also of varying size, the use of dense layers is prohibitively expensive in terms of number of parameters and computational cost (if applied over all pixels). This is why Fully Convolutional Networks are proposed as such:

Definition 3.4.3. A **Fully Convolutional Network** (FCN) is a type of CNN architecture making use of *locally connected layers* such as convolution pooling or upsampling (Shelhamer, Long, & Darrell, 2017).

Avoiding densely connected layers allows for lower parameter counts, and ensures any input size can be used. Lastly the properties of convolutions and pooling blocks make this type of architectures inherently translation invariant. This last property is very useful in the context of biological imagery or remote sensing for instance, where images are large and the processing demands to be translation invariant, as the elements of interest are not necessarily centered as they most often are in *usual* images (e.g. photographs, taken at eye-level, with a centered subject).

Remark 3.4.1 – The translation invariance is valid if we disregard edge effects. These can be dealt with proper padding or cropping since the size of the edge effect can be computed. To compute a result on big images that do not fit in memory, one has to prepare patches of the source image with padding for each patch. The inference results on each patch can then be cropped to remove edge effects and stitched back together without any difference to inferring on the whole image at once (up to float precision errors). See Appendix B for more details.

By construction these FCN architectures are good at extracting patterns and texture. However, to build longer range interactions between objects within the image requires further increasing the depth at an increased cost in number of parameters and computation. Attention mechanism — originally introduced for natural language processing (Vaswani et al., 2017) and then applied to object detection (Carion et al., 2020) — propose a mechanism to relate each pixel with every other pixel in a feature map. This kind of mechanism is done at a great parameter and computational cost too.

While a lot of work is being done on tuning CNN architecture to any task, in our work we focus on complementing a basic FCN model with the Point Process framework described in the first sections of this chapter; we will not go much further in depth in the design of neural networks but rather consider what we can extract from those.

3.5 Conclusion

In this chapter we first presented the building blocks of Point Process models. Those allow modeling configurations of points as a random variable that depend on the underlying image. This model is defined by a Gibbs density, i.e. through an energy function that measures the compatibility of a configuration $\mathbf{y}$ with the image $\mathbf{X}$. At inference, for a given image $\mathbf{X}$, the estimated configuration $\hat{\mathbf{y}}$ is obtained by approximating the minimum of that energy function, through a Monte Carlo Markov Chain sampling method combined with simulated annealing.

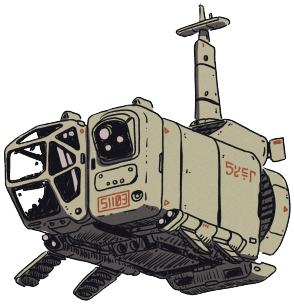
Meanwhile, Convolutional Neural Networks introduce a broad category of models to transform images into new representations. The key breakthrough of these approaches is to learn the convolution filter parameters on a training dataset through gradient descent.

While Point Processes model the high level objects and their interactions (here our objects of interest: vehicles), the CNN methods shine in performing pixel level transformations. Contrast measures try to bridge this pixel-to-object gap inherent to the Point Process approach, but remain quite tedious to tune and often fail on more complex applications (see Section 4.2.1.1). This mo-

tivates us to use CNN methods to transform the image into a representation that is easier to then factor into the energy function of the Point Process, in a way reducing this pixel-to-object gap.

PART

Contributions



CHAPTER 4

Building an Energy Based Model for object detection

Dans ce chapitre, nous construisons le Processus Ponctuel à travers sa fonction d'énergie. Nous montrons d'abord les limites des mesures basées sur le contraste pour les images satellitaires, puis procédons à la construction d'un terme d'attache aux données à partir de la sortie d'un CNN. Nous proposons plusieurs manières de construire une carte de potentiel ; nous constatons qu'une approche à base de vecteurs pointant vers les centres, puis divergence donne les meilleurs résultats. Nous montrons également comment réinterpréter une sortie de pseudo-probabilité classique d'un CNN en une énergie pour le Processus Ponctuel. De plus, nous proposons un ensemble d'a priori pour régulariser les configurations de véhicules, tout en étant prudent sur le choix de l'opérateur d'agrégation des multiples interactions d'un objet. Enfin, nous discutons de la façon dont l'inférence du CNN est factorisée dans le pipeline de calcul d'énergie pour un coût de calcul limité.

In this chapter we build the Point Process through its energy function. First we show the limitations of contrast based measures for satellite images, and proceed to build a data term from the output of a CNN. We propose several ways to build a potential map; we find that a center-pointing-vector to divergence approach gives the best results. We also show how to re-interpret a classical pseudo-probability output of a CNN into an energy for the Point Process. Furthermore, we propose a set of priors to regularize configurations of vehicles, while being careful about the choice of the aggregation operator over the multiple interactions of an object. Lastly, we discuss how the CNN inference is factorized within the energy computation pipeline for a limited computational cost.

4.1 Point Process as an energy model 69
4.2 Data terms from a CNN 70
4.2.1 Contrast measures 71
4.2.1.1 Limitations of contrast measures: example 72
4.2.2 CNN data terms for unmarked points 73
4.2.2.1 Contrast measures on CNN output 73
4.2.2.2 Inferring vector fields 76
4.2.2.3 Interpreting CNN outputs as energy 80
4.2.3 Data energy on marks 80
4.3 Priors on configurations 83
4.3.1 Point priors 83
4.3.1.1 Area and ratio priors 83
4.3.1.2 Joint area and ratio prior 84
4.3.2 Interaction priors 86
4.3.2.1 Aggregation operators 86
4.3.2.2 Non overlapping prior 86
4.3.2.3 Alignment prior 87
4.3.2.4 Repulsive and attractive priors 87
4.3.2.5 Zero neighborhood prior 87
4.3.3 Triplet priors 88
4.4 Resulting model and discussions 89
4.4.1 Model pipeline 89
4.4.2 Discussions 89
4.4.2.1 Continuity 89
4.4.2.2 Mixing of energy terms 91
4.4.2.3 Towards generic (learnable) energy terms 92
4.5 Conclusion 92

4.1 Point Process as an energy model

In this chapter we build a Point Process for detecting small objects in remotely sensed images. This Point Process is derived from an energy function via a Gibbs density (see (3.9)). For a given image $\mathbf{X}$ and configuration $\mathbf{y}$, the unnormalized density of the Point Process is defined as:

$$h(\mathbf{y}) = \exp\left(-U(\mathbf{y}, \mathbf{X})\right). \tag{4.1}$$

We now consider a Point Process of rectangles parametrized as follows: A point y in $\mathbf{y}$, has two spatial coordinates in $\mathcal{S}$, y_i and y_j. The marks $(y_a, y_b, y_\alpha) \in \mathcal{M}$ encode respectively the width, length and angle relative to the $\overrightarrow{i}$ vector as shown in Figure 4.1. The marks are bounded, and the angles are defined modulo π such that $\mathcal{M} = [a_{\min}, a_{\max}] \times [b_{\min}, b_{\max}] \times [0, \pi]$. Describing rectangles using sizes $(y_a y_b)$ and ratios (y_a/y_b) is equivalent.

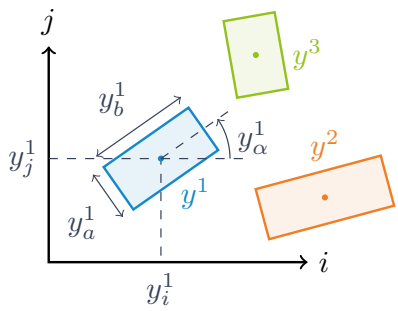


Figure 4.1: Parametrization of points for a marked Point Process of rectangles.

Classical formulation of a Gibbs Point Process performs the sum of an energy function over the cliques of configuration $\mathbf{y}$ (3.9). But formulating the energy as a sum of energies per point — that may depend on their neighborhood — is equivalent up to the size of the cliques in question (see Section 3.2.1.1):

$$\boxed{U(\mathbf{y}, \mathbf{X}) = \sum_{y \in \mathbf{y}} V\left(y, \mathbf{X}, \mathcal{N}_{\{y\}}^{\mathbf{y}}\right).} \tag{4.2}$$

For object detection, the energy of a point $V(y, \mathbf{X}, \mathcal{N}_{\{y\}}^{\mathbf{y}})$ should encompass both the fitness of point y against the image $\mathbf{X}$, and the prior knowledge on objects and their interaction. For that matter we write the energy of a point, as a sum of several energy terms:

$$\boxed{V\left(y, \mathbf{X}, \mathcal{N}_{\{y\}}^{\mathbf{y}}\right) = w_0 + \sum_{e \in \xi} w_e V_e\left(y, \mathbf{X}, \mathcal{N}_{\{y\}}^{\mathbf{y}}\right),} \tag{4.3}$$

with w_0, w_e some weight parameters, and ξ the set of energy terms, split into the following categories:

- *Internal / prior energies* encode the internal coherence of the point configuration. Either at a single point level ($e \in \xi_{prior-point}$) or considering the interactions of a point with its neighborhood ($e \in \xi_{prior-interact.}$).

- *External / data energies* ($e \in \xi_{data}$) reflects how good the point fits the image data $\mathbf{X}$. With this formulation, we measure the correspondence of a single point y against the image $\mathbf{X}$. The Bayesian approach mentioned in Section 3.2, measuring the likelihood of the image $\mathbf{X}$ given the configuration $\mathbf{y}$, would require a term over the $V_e(\mathbf{y}, \mathbf{X})$.

Formulating the energy of a point V as a sum of energy terms V_e — albeit classical, see (Ortner et al., 2007; Lafarge et al., 2010; Descombes, 2017) — remains a modeling choice; we discuss other possibilities in Section 4.4.2.2.

In the following example we show how the above-described framework allows for easy combinations of simple Point Processes to create more complex patterns.

Example 4.1.1 – Let 3 points processes Φ_1, Φ_2, Φ_3 on $\mathcal{S} \times \mathcal{M} = [0,1]^2 \times [0.01, 0.1]$, each defined to exhibit one basic behavior, and derived from a single energy term V_1, V_2, V_3. We denote y_i, y_j, y_r respectively the position in the two spatial axes, and the radius of each point. The processes are defined as such:

- *For Φ_1, points in the white pixels of image $\mathbf{X}$ are more likely to be rejected. For a point y the energy corresponds to the value of image $\mathbf{X}$ (values in $[0,1]$) at location (y_i, y_j):*

$$V_1(y, \mathbf{X}) = \mathbf{X}[y_i, y_j].$$

- *Process Φ_2 avoids overlaps; with $d(y,u)$ the distance between y and u, energy term V_2 checks if the gap between the two circles is greater than zero with all neighbors, if not it assigns an energy of 1:*

$$V_2(y, \mathcal{N}^{\mathbf{y}}_{\{y\}}) = \max_{u \in \mathcal{N}^{\mathbf{y}}_{\{y\}}} \{\mathbb{1}_{d(y,u)-y_r-y'_r<0}(y)\}.$$

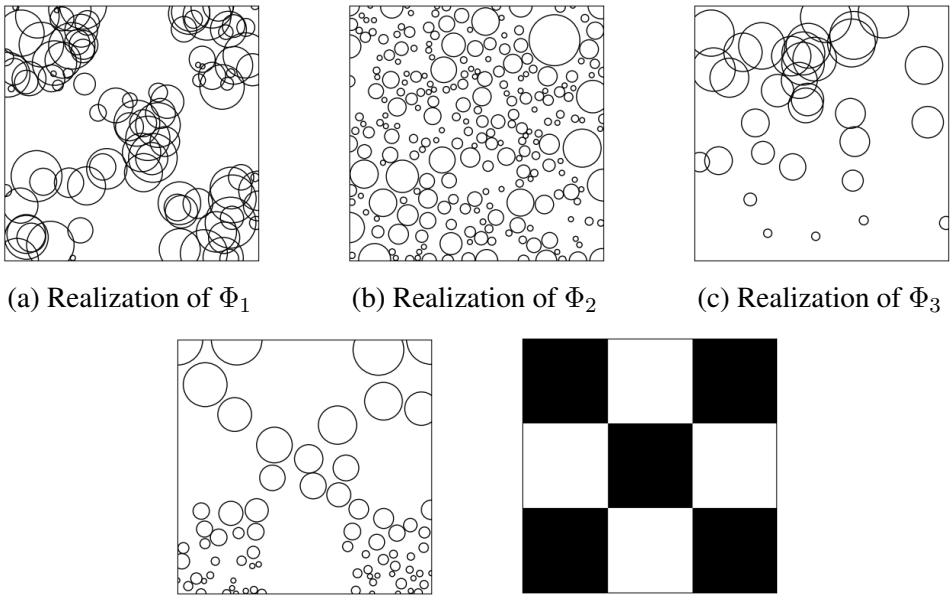
- *Process Φ_3 favors circles of growing size as points get higher in the image, within a margin 0.01:*

$$V_3(y) = \mathbb{1}_{|y_r-(0.99y_j+0.01)|>0.01}(y).$$

Finally, we define a composite process Φ_4, from the sum of the energies V_1, V_2, V_3 (or product of densities). As illustrated in Figure 4.2, Point Process Φ_4 exhibits a combination of all the constraints set on each Φ_1, Φ_2, Φ_3.

4.2 Data terms from a CNN

The data terms of the Point Process aim to measure the fitness of each point $y \in \mathbf{y}$ against the image data $\mathbf{X}$. First we look at contrast measures and their limitations, then we describe our contribution to use Convolution Neural Networks (CNN, introduced in Section 3.4) for these data terms.



(a) Realization of Φ_1 (b) Realization of Φ_2 (c) Realization of Φ_3

(d) Realization of Φ_4 (e) Image $\mathbf{X}$

Figure 4.2: Realizations of the processes described in Example 4.1.1. The radius of each circle corresponds to the radius marks y_r.

4.2.1 Contrast measures

Point Process models in (Lacoste et al., 2005; Ortner et al., 2007; Crăciun, 2015; Descombes, 2017) use contrast measures to build a data potential for each point. The idea is to compare statistical properties of pixels values of image $\mathbf{X}$ inside and around the object y.

The idea is to compare the distributions inside and outside the shape formed by y. For example (Lacoste et al., 2005) use the T-test on the values inside shape $i(y)$ and on its contour $c(y)$:

$$V(y, \mathbf{X}) \propto \frac{|\mu_{i(y)} - \mu_{c(y)}|}{\sqrt{\frac{\sigma^2_{i(y)}}{n_{i(y)}} + \frac{\sigma^2_{c(y)}}{n_{c(y)}}}}, \tag{4.4}$$

with $\mu_{i(y)}, \sigma_{i(y)}$ the mean and variance of values of image $\mathbf{X}$ in interior $i(y)$, similarly for contour $s(y)$.

Alternatively (Kulikova et al., 2011) propose to check the image gradient $\nabla\mathbf{X}$ against the normal vector $\overrightarrow{n_y}(t)$ along the contour line $s_y(t)$, $t \in [0, 1]$:

$$V(y, \mathbf{X}) \propto \int_{t\in[0,1]} \overrightarrow{n_y}(t) \cdot \frac{\nabla\mathbf{X}[s_y(t)]}{\sqrt{|\nabla\mathbf{X}[s_y(t)]|^2 + \epsilon}} dt, \tag{4.5}$$

where $\nabla\mathbf{X}[s_y(t)]$ is the image gradient at contour coordinate t.

While there are many other formulations for contrast measure, most rely on the assumption of simple backgrounds, limited object of interest variations, and having little to no high contrast background elements. Moreover, the definition of interior and outside contour of the object y is to be fine-tuned to the application; e.g. the gap and thickness of considered contours (see Figure

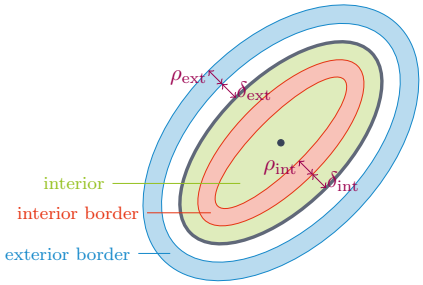


Figure 4.3: Sampling areas to compute contrast for an ellipse. Parameters $\rho_{\text{int}}, \rho_{\text{ext}}, \delta_{\text{int}}, \delta_{\text{ext}}$ drive the size and locations of these surfaces.

4.3). Similarly, in order to deal with heterogeneity inside the object (Crăciun, 2015) measure the contrast between an interior contour and exterior one (red and blue in Figure 4.3); as the visual features inside the boats (e.g. windows, decks, etc...) increase the variance of inside pixels.

On the other hand CNN models have shown to be very efficient at extracting features from images for object detection and classification. In the following section, we will show how to interpret a CNN based object detector output to obtain an energy that measures the fitness of a configuration against an image. It allows us to go past the contrast measure design by utilizing a pre-trained CNN output. The next example shows a basic comparison between contrast measures and a CNN model.

4.2.1.1 Limitations of contrast measures: example

We compare the performances of contrast measures on both remote sensing images and synthetic data. We evaluate the performance of each by generating a set of object proposals from the image, 99% of which do not correspond to an object, 1% are actual objects of interest. For a contrast measure to be effective, it should be able to score high on objects of interest and low on negative proposals. For every contrast threshold we can assign positive and negative detections to each proposal, thus compute precision and recall values. Computing precision and recall for every threshold gives us the precision-recall curve.

We compare the two contrast measures given in (4.4) and (4.5) against our CNN model (see next sections) and a measure that always returns a constant value[1].

This comparison is performed on both a synthetic image and a sample from the dataset DOTA (Xia et al., 2018) (sampled at 0.5m resolution). The synthetic data consists of bright or dark rectangles on a gray background with some additive noise. The data and results are shows in Figure 4.4.

The contrast measure performs excellently on the synthetic data, but struggles on real images. Indeed, this image contains a lot of high contrast elements that are not objects of interest; air-conditioning units, pavement features, trees etc...

[1]For heavily imbalance datasets such a classifier can get decent results (e.g. if we invert the positive to negative proportions); we introduce it as a baseline check.

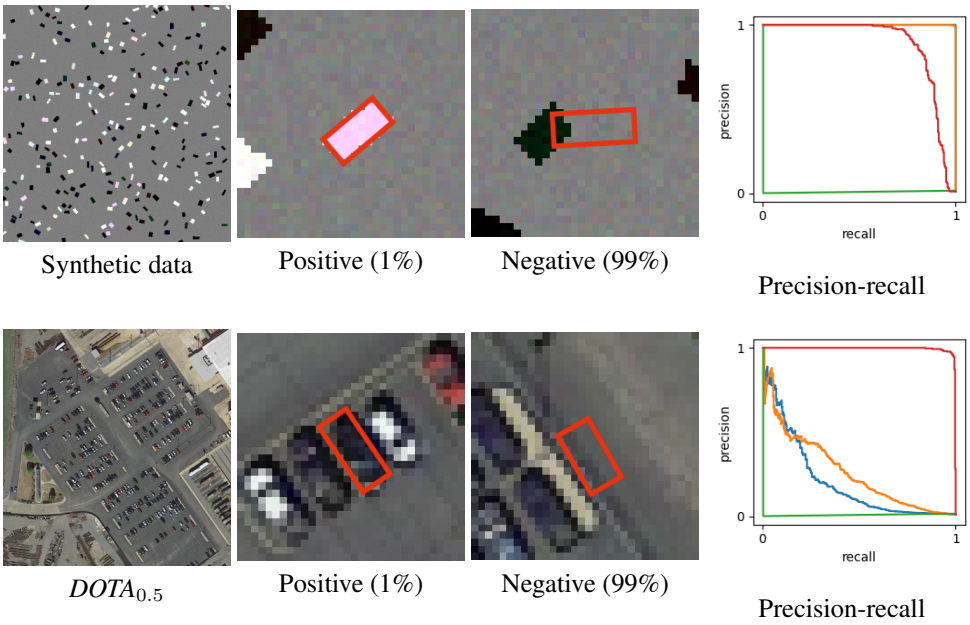


Figure 4.4: Comparing contrast measures on synthetic (first line) and real data (second line). Positive and negative samples, represent respectively 1% and 99% of the samples for which the metric is computed. For precision-recall; blue: T-test; orange: image-gradient; red: CNN output; green: constant value. On the top right plot, blue and orange plots are overlapping.

4.2.2 CNN data terms for unmarked points

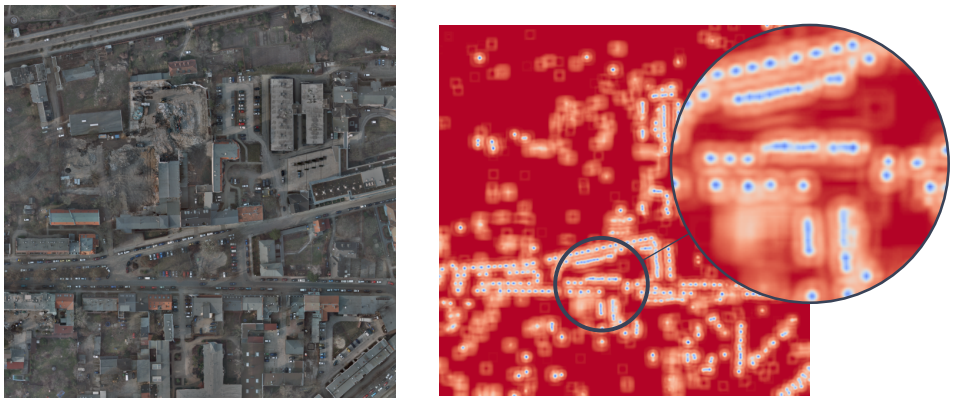
4.2.2.1 Contrast measures on CNN output

To build a data term from a CNN for the Point Process, we aim at inferring from a given image $\mathbf{X}$ (Figure 4.5 (a)) a map of potentials $\mathbf{V}_{pos}$ (Figure 4.5 (b)), so that the data energy V_{pos} of a point y can be computed by simply looking up the value of $\mathbf{V}_{pos}$ at the corresponding position.

Previous works on marked Point Processes using contrast measures have shown good results when applied to high contrast images where background clutter is limited. Thus, our first approach consists in applying a tailored contrast measure in an image transformed by a FCN model into a more simple representation. Ideally the image transformation model should produce simple contrasted patterns on object positions, while simplifying background to be as close as possible to uniformity. Applying the pattern matching would in turn provide potential map $\mathbf{V}_{pos}$. The following approach was published in (Mabon et al., 2021).

Inferring heatmaps. First we must determine the output of our CNN. We need to build target (or Ground Truth, GT) maps with desired properties on training data, for the model to be able to reproduce such maps on unseen data. Object detection models such as (T. Zhao, Liu, Celik, & Li, 2021; Yi et al., 2021; Huang, Li, Xia, & Tao, 2022) use heatmaps to identify object centers.

> **Definition 4.2.1.** For object detection, a **heatmap** refers to a map (often 2 dimensional) of probability-like values measuring the presence of an element or event; in our case object centers.



(a) Input image (b) Potential $\mathbf{V}_{pos}$

Figure 4.5: Example of energy map: (a) input image $\mathbf{X}$ from COWC (Mundhenk et al., 2016); (b) results of $\mathbf{F}$ passed through function q_{pos}, red is high positive potential, blue is negative. We show a zoom on part of the image where nearby blobs clump together.

Remark 4.2.1 – Neural network models — even if returning values in $[0, 1]$ — produce only estimates of the likelihood: as (C. Guo et al., 2017) shows, these measures are often poorly calibrated and should rather be considered as scores. We will also refer to those as probability estimates.

Training the CNN model to infer an *exact* map such as it returns 1 at object centers and zeros elsewhere (see Figure 4.6) proves to be impractical; the image data and annotations are sometimes off by a few pixels (making the sharp target map incorrect) and the labels are too imbalanced to properly train the model (i.e. the ratio of center pixels to background is close to zero).

Relaxing the heatmap. We propose *relaxing* the training target, in learning to infer a map $\mathbf{H}$ such that for every pixel ρ in the image:

$$\mathbf{H}[\rho] = \max\left\{1 - \frac{d(\rho, \mathbf{y}^{GT})}{r_{max}}, 0\right\}, \tag{4.6}$$

where $d(\rho, \mathbf{y}^{GT})$ is the distance of pixel ρ to the nearest point in $\mathbf{y}^{GT}$ and r_{max} a threshold term (see Figure 4.6). This relaxing produces a relaxed heatmap target to train out model on.

Balancing. The works in (M. Bai & Urtasun, 2017) on inference of *discrete watershed* map for instance segmentation propose a solution to alleviate class imbalance issues. Their approach requires them to infer a distance map in the likes of (4.6); they propose to *discretize* the distance values into n_c distinct bins or classes (see Figure 4.6). We now have to use the CNN to estimate the most likely class (or value bin). This allows to re-weight the loss over each class to deal with the imbalance.

Definition 4.2.2. In image processing, a **watershed** (Beucher & Lantuejoul, 1979; Vincent & Soille, 1991) is a transform defined on a grayscale image, where lower levels (lower pixel

brightness) of the image get progressively flooded to form a segmentation of the image. By extension the *watershed map* refers to the image the watershed method is applied on.

The ground truth (or training target) map is then, for every pixel ρ:

$$\mathbf{H}_{dscr}[\rho, c] = \begin{cases} 1 & \text{if } \mathbf{H}[\rho] \in \left[\frac{c-1}{n_c}, \frac{c}{n_c}\right] \\ 0 & \text{else.} \end{cases}, \forall c = 1, \ldots, n_c. \tag{4.7}$$

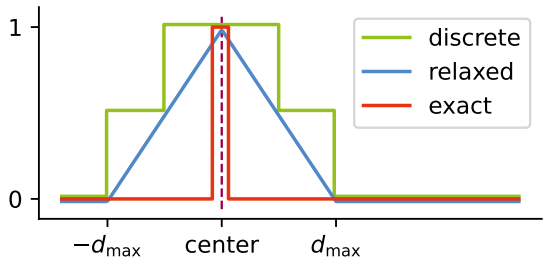


Figure 4.6: Relaxation and discretization of the target map illustrated for a single object in 1D. For the discrete plot, values are scaled from range $[0, n_c]$ to $[0, 1]$ for ease of reading.

Estimation and training. The Unet estimates scores $\widehat{\mathbf{H}}_{dscr}$ from image $\mathbf{X}$, for each discrete class of values c, by the application of a Softmax (Goodfellow et al., 2016, 6.2.2.3 Softmax Units for Multinoulli Output Distributions) on the Unet output $\mathbf{Z}$ of size (H, W, n_c) as :

$$\text{Softmax}(\mathbf{Z}[\rho])_c = \frac{\exp\left(\mathbf{Z}[\rho, c]\right)}{\sum_{c'=1}^{n_c} \exp\left(Z[\rho, c']\right)}. \tag{4.8}$$

The CNN model is trained to minimize the cross entropy loss (Kline & Berardi, 2005):

$$\mathcal{L}(\mathbf{H}_{dscr}, \widehat{\mathbf{H}}_{dscr}) = -\sum_{\rho} \sum_{c=1}^{n_c} w_c \mathbf{H}_{dscr}[\rho, c] \log\left(\widehat{\mathbf{H}}_{dscr}[\rho, c]\right), \tag{4.9}$$

with $w_c = 1 - \frac{1}{n_c} \sum_{\rho} \mathbf{H}_{dscr}[\rho, c]$ a weighting factor introduced to amplify the loss on the less frequent classes (i.e. foreground classes in our case).

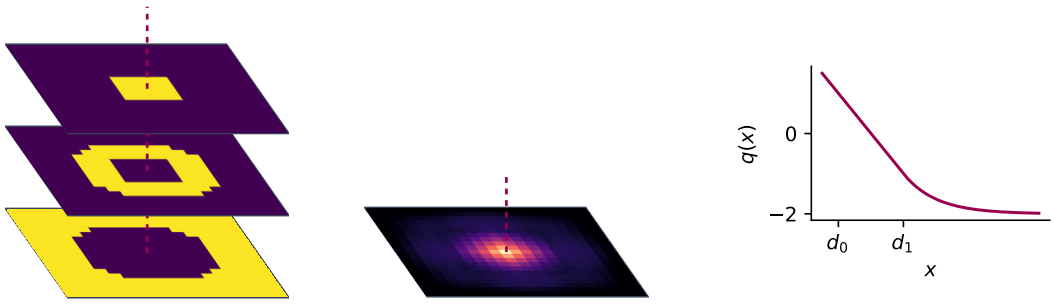
Building the data potential. Since we know by construction of the target map the aspect of the map around object centers, we devise a filter $\mathbf{K}$ as illustrated in Figure 4.7 (a) to look for object centers when cross-correlated with the output map $\widehat{\mathbf{H}}_{dscr}$. When the filter pattern matches the pattern in the output map we get a high response, otherwise it is low, as shown in Figure 4.7 (b). This approach is akin to using a contrast measure as the filter is looking at a specifically shaped pattern of contrast in the image. Using cross-correlation to look for this pattern allows to pre-compute the contrast for every pixel of the image easily. We denote $\mathbf{F}[\rho] = (\mathbf{K} * \widehat{\mathbf{H}}_{dscr})[\rho]$

the cross-correlation response at pixel ρ, and thus define the data potential[2] as :

$$V_{pos}(y, \mathbf{X}) = q_{pos}\left(\mathbf{F}[\rho_y]\right), \tag{4.10}$$

$$q_{pos}(x) = \begin{cases} \frac{-2(x-d_0)}{d_1-d_0} + 1 & \text{if } x < d_1 \\ \exp\left(\frac{-2(x-d_0)}{d_1-d_0} + 2\right) - 2 & \text{else,} \end{cases} \tag{4.11}$$

where ρ_y is the pixel containing y. The quality function and its two parameters d_0, d_1 is illustrated in Figure 4.7. With the data potential defined as such, the map F can be pre-computed for a given image $\mathbf{X}$ from the Unet output, to be then used for any computation of $U(\mathbf{y}, \mathbf{X})$, simplifying the contrast computation to the sampling of an array.



(a) Filter (b) Filter response (c) $q_{pos}(x)$

Figure 4.7: Filter and response for energy potential

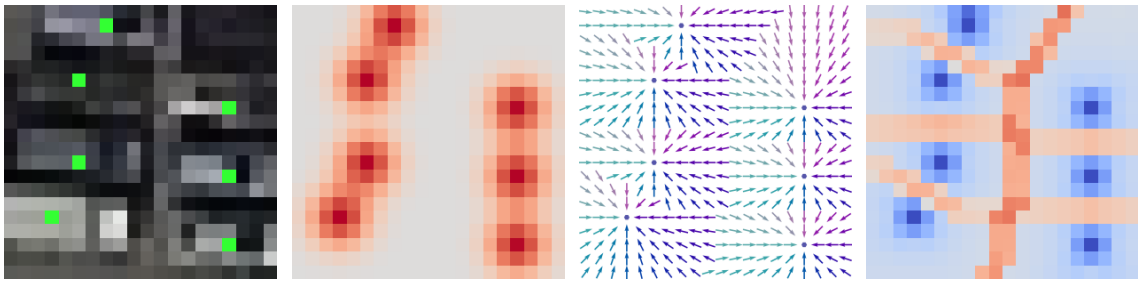
Limitations. While this first approach allows to extract an energy map, we notice that nearby *blobs* formed by objects in the energy map can sometimes clump together. We illustrate this in Figure 4.5; the clumping of nearby detections into one long blob is quite pronounced for cars parked side-to-side (as opposed to front-to-back). Moreover, the energy pits around detected centers are quite wide — as a side effect of the relaxation — making the positioning of object imprecise.

> *Remark 4.2.2 – The focal loss (Lin, Goyal, et al., 2017) propose to weight the loss for imbalanced data such as heatmap without any discretization. In our case, learning to infer the heatmap* $\mathbf{H}$ *directly with focal loss does not improve significantly on the clumping issue.*

4.2.2.2 Inferring vector fields

The previous approach to build a data energy is limited in how it can efficiently separate nearby instances. Here we propose an approach based on vectors and their divergence to better pinpoint the location of objects. The following approach has been published in (Mabon et al., 2022b) and (Mabon, Ortner, & Zerubia, 2022c).

[2]Denoting *pos* for position, as the model can include a potential for marks too, that will be discussed in a further Section.



(a) (b) (c) (d)

Figure 4.8: (a) Closeup image with centers overlay; (b) centers heatmap $\mathbf{H}$; (c) vector field $\mathbf{H}_\nabla$; (d) divergence. For (b) and (d): red> 0, blue< 0.

Vectors point to object center. To do instance separation (Neven, Brabandere, Proesmans, & Van Gool, 2019) propose to infer for each pixel of the image the vector from the pixel location to the center of the object the pixel belongs to. Outside the objects the vectors are not defined.

With our small objects, we rather choose to extend the range at which these vectors are inferred outside the area of the object; within a radius r_{max} the model has to infer the unit vector that points towards the nearest object center as illustrated in Figure 4.8. In short, we look to infer the map:

$$\mathbf{H}_\nabla[\rho] = \begin{cases} \frac{1}{\|y-\rho\|_2} \begin{bmatrix} y_i - \rho_i \\ y_j - \rho_j \end{bmatrix}, \; y = \arg\min_{y' \in \mathbf{y}^{GT}} \{\|y' - \rho\|_2\} & \text{if } \mathbf{H}_M[\rho] = 1 \\ \begin{bmatrix} 0 \\ 0 \end{bmatrix} & \text{otherwise} \end{cases} \tag{4.12}$$

The map $\mathbf{H}_M$ is a binary map, equal to one if less than r_{max} away from an object center, else it is zero:

$$\mathbf{H}_M[\rho] = \begin{cases} 1 & \text{if } \arg\min_{y' \in \mathbf{y}^{GT}} \{\|y' - \rho\|_2\} \leq r_{max} \\ 0 & \text{otherwise.} \end{cases} \tag{4.13}$$

Remark 4.2.3 – Note that $\mathbf{H}_\nabla$ can be obtained from the image gradient of $\mathbf{H}$ defined in (4.6):

$$\mathbf{H}_\nabla[\rho] = \begin{cases} \frac{\nabla \mathbf{H}[\rho]}{\|\nabla \mathbf{H}[\rho]\|_2} & \textit{if } \mathbf{H}_M[\rho] = 1 \\ \begin{bmatrix} 0 \\ 0 \end{bmatrix} & \textit{otherwise} \end{cases}, \quad \nabla \mathbf{H}[\rho] = \begin{bmatrix} \frac{\partial \mathbf{H}}{\partial i}[\rho] \\ \frac{\partial \mathbf{H}}{\partial j}[\rho] \end{bmatrix}. \tag{4.14}$$

The gradient is defined over continuous space, but can be easily approximated in discrete pixel space by the following cross correlation:

$$\nabla \mathbf{H} \simeq \begin{bmatrix} \begin{bmatrix} -1 & 0 & +1 \end{bmatrix} * \mathbf{H} \\ \begin{bmatrix} -1 \\ 0 \\ +1 \end{bmatrix} * \mathbf{H} \end{bmatrix}$$

Inference and training. To produce a field of vectors $\widehat{\mathbf{H}}_\nabla$ from image $\mathbf{X}$, we modify the Unet to output two tensors $\mathbf{Z}_\nabla$ and $\mathbf{Z}_M$ of respective sizes $(H, W, 2)$ and $(H, W, 1)$. The output $\mathbf{Z}_M$ allows estimating $\mathbf{H}_M$ as such:

$$\widehat{\mathbf{H}}_M[\rho] = \sigma\left(\mathbf{Z}_M[\rho]\right). \tag{4.15}$$

Thus, the estimate of $\mathbf{H}_\nabla$ can be obtained as:

$$\widehat{\mathbf{H}}_\nabla[\rho] = \widehat{\mathbf{H}}_M[\rho]\frac{\mathbf{Z}_\nabla[\rho]}{\|\mathbf{Z}_\nabla[\rho]\|_2}, \tag{4.16}$$

with $\|\mathbf{Z}_\nabla[\rho]\|_2$ the norm of 2-dimensional vector $\mathbf{Z}_\nabla[\rho]$. The normalizing allows to consistently maintain unit vectors in the neighborhood of objects, while the product with $\widehat{\mathbf{H}}_M[\rho]$ sets to zero all out-of-range pixels. The model is trained to minimize the Mean Square Error (MSE) between the estimated and ground truth vectors (inside the r_{max}-neighborhood of each point) while minimizing the Binary Cross Entropy (BCE) between the mask and its estimation:

$$\begin{aligned}\mathcal{L}(\widehat{\mathbf{H}}_\nabla, \widehat{\mathbf{H}}_M, \mathbf{H}_\nabla, \mathbf{H}_M) = &\overbrace{\sum_\rho \mathbf{H}_M[\rho]\left\|\widehat{\mathbf{H}}_\nabla[\rho] - \mathbf{H}_\nabla[\rho]\right\|_2^2}^{\text{MSE}(\widehat{\mathbf{H}}_\nabla, \mathbf{H}_\nabla) \text{ where } \mathbf{H}_M[\rho]=1} \\ &\underbrace{-\sum_\rho \mathbf{H}_M[\rho]\ln(\widehat{\mathbf{H}}_M[\rho])}_{\text{BCE}(\widehat{\mathbf{H}}_M, \mathbf{H}_M)}.\end{aligned} \tag{4.17}$$

We provide some details about the training procedures in Appendix B.

Remark 4.2.4 – In practice, we infer a single tensor $\mathbf{Z}$ *of size* $(H, W, 3)$ *and then extract the relevant features/channels to obtain* $\mathbf{Z}_\nabla$ *and* $\mathbf{Z}_M$ *of respective sizes* $(H, W, 2)$ *and* $(H, W, 1)$.

Divergence as data potential. Given the inferred $\widehat{\mathbf{H}}_\nabla$ for image $\mathbf{X}$, we need to build a scalar data potential for each element of $\mathbf{y}$. By construction the center of each object corresponds to a point where vectors converge. It can be seen clearly in Figure 4.8 (c) and (d). Thus, we use the divergence operator to extract object locations; the object centers get negative values, while separation lines in between get positive values (see Figure 4.8 (d)). Denoting $\widehat{\mathbf{H}}_{div} = \operatorname{div}\widehat{\mathbf{H}}_\nabla$:

$$V_{pos}(y, \mathbf{X}) = \widehat{\mathbf{H}}_{div}[\rho_y]. \tag{4.18}$$

We show the potential map computed over a real unseen (i.e. not trained on) image in Figure 4.9.

Computing the divergence. The divergence in continuous space over a vector field $\mathbf{F}$ in 2D is defined as:

$$\operatorname{div}\mathbf{F} = \frac{\partial \mathbf{F}_i}{\partial i} + \frac{\partial \mathbf{F}_j}{\partial j}. \tag{4.19}$$

The divergence field over in raster grid coordinates can be approximated using the finite difference approximation for derivatives:

$$\frac{\partial \mathbf{F}_i}{\partial i}[y_i] \simeq \frac{\mathbf{F}_i[y_i+1] - \mathbf{F}_i[y_i-1]}{2}. \tag{4.20}$$

The approximate derivate in an image can be obtained by convolution using a Prewitt (Prewitt, 1970) filter:

$$\frac{\partial \mathbf{F}_i}{\partial i} \simeq \begin{bmatrix} +1 & +1 & +1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} * \mathbf{F}_i, \quad \frac{\partial \mathbf{F}_j}{\partial j} \simeq \begin{bmatrix} -1 & 0 & +1 \\ -1 & 0 & +1 \\ -1 & 0 & +1 \end{bmatrix} * \mathbf{F}_j. \tag{4.21}$$

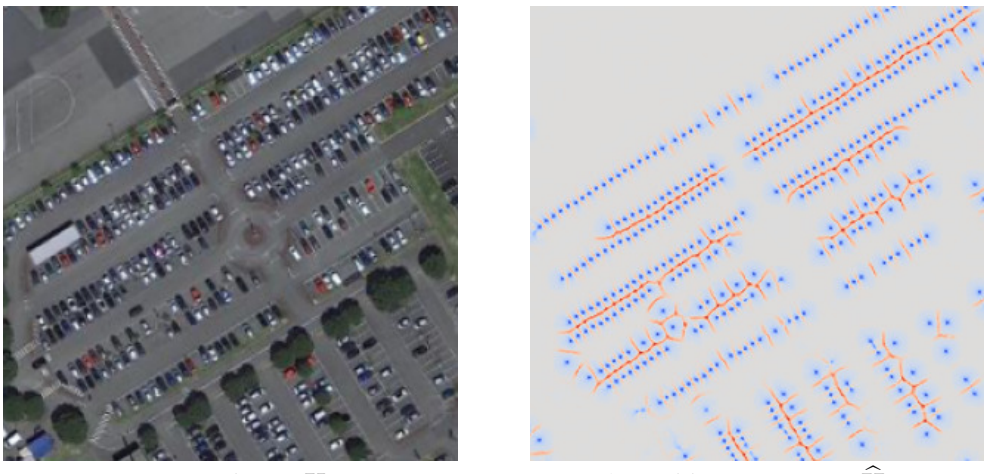
The computation of the divergence for a vector field defined in a raster grid is detailed in Appendix I.3.

Interpolation. The energy described in (4.18) only samples the value of the $\widehat{\mathbf{H}}_{div}$ at integer locations (exact pixel locations). However, points y in configuration $\mathbf{y}$ have real-valued locations $(y_i, y_j) \in \mathbb{R}^2$. Thus, we propose interpolating values in between exact pixel locations, defining:

$$V_{pos}(y, \mathbf{X}) = \widehat{\mathbf{H}}_{div}[y_i, y_j], \tag{4.22}$$

with $\widehat{\mathbf{H}}_{div}[y_i, y_j]$ the bilinear interpolation of $\widehat{\mathbf{H}}_{div}$ at location (y_i, y_j).

Remark 4.2.5 – This vector inference approach can be adapted to produce score of object centers. Given tensor $\widehat{\mathbf{H}}_{div}$, we can produce an approximated probability $\hat{p}(y|\mathbf{X}) = \sigma(a\widehat{\mathbf{H}}_{div} + b)$, where sigma is the sigmoid function and a, b two scalar parameters (e.g. from a linear block in the neural network). As the divergence operator is differentiable, the parameters a, b can be learned during the training of the whole model.



(a) Input image $\mathbf{X}$ (b) Position energy map $\widehat{\mathbf{H}}_{div}$

Figure 4.9: Example of inferred energy map. Image from DOTA (Xia et al., 2018) with corresponding inferred energy map. Blue areas corresponds to negative potentials while red to positive potentials.

In this section we contributed in proposing an energy map built from the divergence of a field of vector inferred by a CNN, allowing for better separability of small tightly-packed instances.

4.2.2.3 Interpreting CNN outputs as energy

Until now, we proposed several network architectures to produce a map of potentials for the data term of our Point Process. In this section we propose a general framework to use the output of any CNN model (within some specifications) to produce the required potential. This was published in (Mabon et al., 2023a) and (Mabon, Ortner, & Zerubia, 2023b) and to be submitted in (Mabon, Ortner, & Zerubia, 2023). The core idea stems from (Grathwohl et al., 2019) where they propose to interpret classifier model as Energy Based Models (EBM).

Here we consider a class of CNN model that infers a map of scores of centers for every pixel of the image such as the heatmap based models (Huang et al., 2022). The model described in the previous section can also be adapted to produce probability estimates (scores) as explained in Remark 4.2.5. This class of models produce a probability estimates of center for pixel ρ from a tensor $\widehat{\mathbf{Z}}_{pos}$ inferred by the CNN. The probability estimate $\hat{p}(\rho)$ is obtained through the sigmoid function as:

$$\hat{p}(\rho) = \sigma\left(\widehat{\mathbf{Z}}_{pos}[\rho]\right) = \frac{1}{1+\exp\left(-\widehat{\mathbf{Z}}_{pos}[\rho]\right)}. \tag{4.23}$$

Meanwhile, if we consider the simplest Point Process model derived from $U(\mathbf{y},\mathbf{X}) = \sum_{y\in\mathbf{y}} V_{pos}(y,\mathbf{X})$, Definition 3.1.7 and Theorem 3.1.2 give us:

$$\begin{aligned} p\left(N_\Phi(dy) = 1 | \Phi \cap (dy)^c = \mathbf{y} \cap (dy)^c\right) &= \frac{h(\{u\}\cup\mathbf{y})}{h(\mathbf{y})} dy \\ &= \exp(-V_{pos}(y,\mathbf{X}))dy \end{aligned} \tag{4.24}$$

By considering (4.24) over the whole pixel ρ, and relating it the probability estimate in (4.23), we propose to define V_{pos} such as:

$$\exp(-V_{pos}(y,\mathbf{X})) = \sigma\left(\widehat{\mathbf{Z}}_{pos}[\rho(y)]\right). \tag{4.25}$$

This yields:

$$V_{pos}\left(y,\mathbf{X}\right) = q_{pos}\left(\widehat{\mathbf{Z}}_{pos}[y_i,y_j]\right) \tag{4.26}$$

$$q_{pos}(x) = \ln\left(1+\exp\left(-x+t_{pos}\right)\right), \forall x \in \mathbb{R}, \tag{4.27}$$

where t_{pos} is a scalar parameter we introduce to offset the inflection point of function $x \mapsto \ln(1+\exp(-x))$ as shown in Figure 4.10. Once again we can perform bilinear interpolation to have the potential be defined for any real-valued coordinate in $\mathcal{S}$ instead of discrete pixels.

4.2.3 Data energy on marks

For now, we only considered potentials on the position of the objects, in this section we propose a potential on the marks of the Point Process.

First we discretize each mark κ ($\kappa \in \{a,b,\alpha\}$ in the case of a Point Process (PP) of rectangles), into n_κ classes (or value bins) in the range $[\kappa_{\min},\kappa_{\max}]$. We define the integer class of a value v for mark κ as:

$$c_\kappa(v) = n_\kappa \frac{v-\kappa_{\min}}{\kappa_{\max}-\kappa_{\min}}, \forall v \in [\kappa_{\min},\kappa_{\max}], \tag{4.28}$$

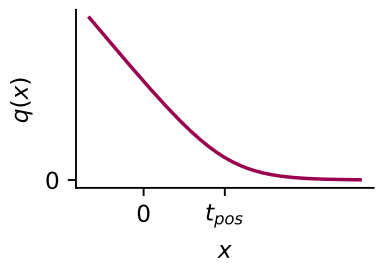


Figure 4.10: Illustration of function from (4.27) and its inflection point t_{pos}.

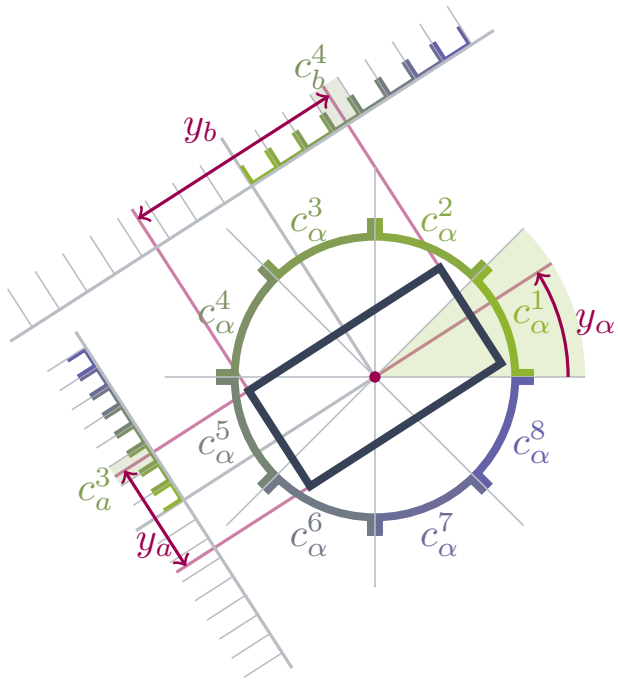


Figure 4.11: Mark discretization for $n_\kappa = 8$: here the angle α belongs to class $c_\alpha^1 = \lceil c_\alpha(y_\alpha) \rceil$, the width a to class $c_a^3 = \lceil c_a(y_a) \rceil$ etc...

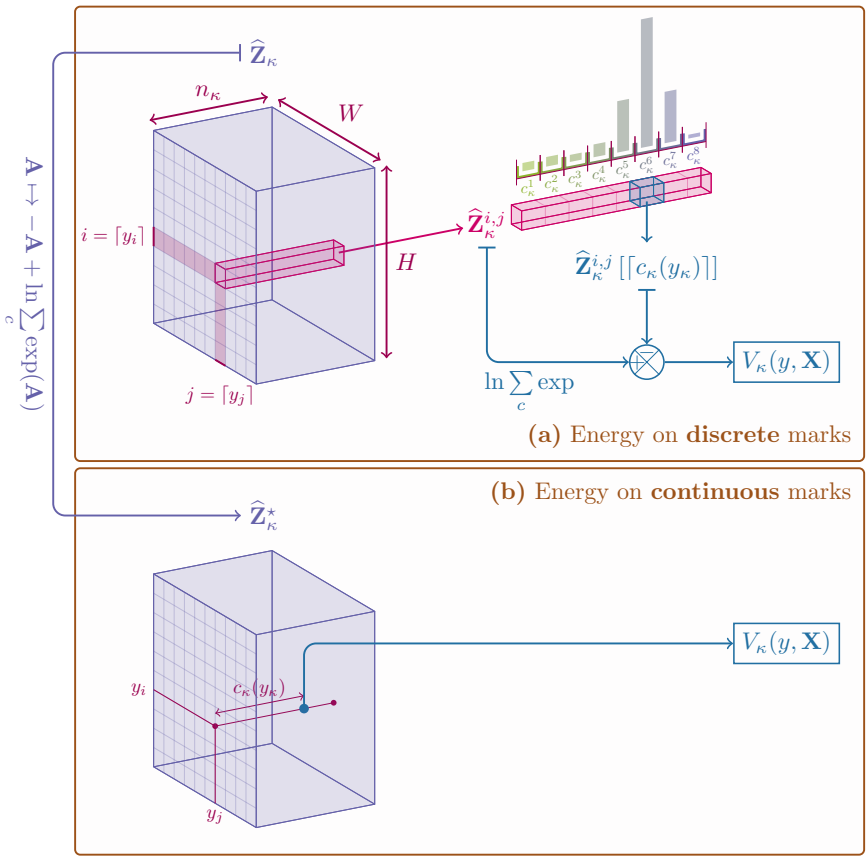


Figure 4.12: Energy over discrete marks (a) and continuous marks (b).

with $\lceil c_\kappa(v) \rceil$ the corresponding integer class. We illustrate the discretization in Figure 4.11: here we pick $n_\kappa = 8$ for $\kappa \in \{a, b, \alpha\}$, a low value for illustration purposes (in practice we pick $n_\kappa = 32$). For instance, in this illustration we have $\lceil c_\alpha(y_\alpha) \rceil = c_\alpha^1$, $\lceil c_a(y_a) \rceil = c_a^3$ etc...

Now, supposing we have a model trained to classify the mark of an object at a given position $(i, j) \in \mathcal{S}$, such model will produce a probability estimate of mark κ to be in class $c \in [\![1, n_\kappa]\!]$, from model output $\widehat{\mathbf{Z}}_\kappa^{i,j} \in \mathbb{R}^{n_\kappa}$ as :

$$\hat{p}(c|i, j, \mathbf{X}) = \text{Softmax}(\widehat{\mathbf{Z}}_\kappa^{i,j})_c = \frac{\exp\left(\widehat{\mathbf{Z}}_\kappa^{i,j}[c]\right)}{\sum_{c'=1}^{n_\kappa} \exp\left(\widehat{\mathbf{Z}}_\kappa^{i,j}[c']\right)}. \tag{4.29}$$

As in (Grathwohl et al., 2019), we can reinterpret the model output into a potential, giving:

$$V_\kappa(y, \mathbf{X}) = -\widehat{\mathbf{Z}}_\kappa^{i,j}[\lceil c_\kappa(y_\kappa) \rceil] + \ln\left(\sum_{c=1}^{n_\kappa} \exp\left(\widehat{\mathbf{Z}}_\kappa^{i,j}[c]\right)\right), \tag{4.30}$$

we justify this formulation in Appendix C. As the CNN outputs a tensor $\widehat{\mathbf{Z}}_\kappa$ of shape (H, W, n_κ), we get vector $\widehat{\mathbf{Z}}_\kappa^{i,j}$ by sampling tensor $\widehat{\mathbf{Z}}_\kappa$ at location $(\lceil y_i \rceil, \lceil y_j \rceil)$, as illustrated in Figure 4.12 (a).

Interpolation. Same as the first proposed position energies, the energy on marks defined in 4.30 is defined over the discrete values of the mark. As the marks live in a continuous space, we

propose using interpolation to defined V_κ in between discrete mark values:

$$V_\kappa(y, \mathbf{X}) = \widehat{\mathbf{Z}}^\star_\kappa[y_i, y_j, c_\kappa(y_\kappa)] \tag{4.31}$$

$$\widehat{\mathbf{Z}}^\star_\kappa[i,j] = -\widehat{\mathbf{Z}}_\kappa[i,j] + \ln \sum_{c=1}^{n_\kappa} \exp\left(\widehat{\mathbf{Z}}_\kappa[i,j,c]\right), \forall i,j \in \mathcal{S}_d. \tag{4.32}$$

The above, illustrated in Figure 4.12 (b), has two benefits: first (4.31) defines the energy for continuous values of the marks using bilinear interpolation, thus ensuring Lipschitz continuity of the energy. Second we can pre-compute (4.32) once for a given image; making the mark energy computation a simple value lookup and interpolation for each point. This pre-computation is quite fast, as it is defined as an operation over a tensor, which is implicitly parallelized by tensor computation libraries such as (*PyTorch*, n.d.).

Remark 4.2.6 – (Grathwohl et al., 2019) propose to use the extra degree of freedom of the model output to measure the probability of the observation itself (not its class); indeed the $\mathrm{Softmax}$ *function is invariant to any offset on (*$\mathrm{Softmax}(\mathbf{A} + b) = \mathrm{Softmax}(\mathbf{A})$, $b \in \mathbb{R}$*) which provides a degree of freedom in* $\mathbf{A}$.

In our case we can use $\ln\left(\sum_{c=1}^{n_\kappa} \exp\left(\widehat{\mathbf{Z}}^{i,j}_\kappa[c]\right)\right)$ *as a position potential; measuring the probability of an object at that location regardless of the class. While this would allow to join the position and mark potentials, our experiments have not been conclusive for the moment.*

Our contribution in the last two subsections, is to propose to use any classical CNN for detection as a data term for the Point Process (for both position and marks), based on the suggestion of (Grathwohl et al., 2019) to interpret classifiers as energy models.

4.3 Priors on configurations

With the data potentials defined, we focus now on the design of priors to regularize the configurations of objects. We first detail the potentials that apply to single points, then those that measure the interactions. Some priors introduced in this section are illustrated in Figure 4.13.

4.3.1 Point priors

4.3.1.1 Area and ratio priors

As our objects of interest have quite consistent sizes, we favor configurations where shape parameters fit within some expected range. The ratio prior, compares the point y width to length ratio against a parameter μ_{ratio}

$$V_{ratio}(y) = -\exp\left(-\frac{\left(\frac{y_a}{y_b} - \mu_{ratio}\right)^2}{2\sigma^2_{ratio}}\right). \tag{4.33}$$

Similarly, we define a prior on areas V_{area} by replacing y_a/y_b by $y_a y_b$ with corresponding $\mu_{area}, \sigma_{area}$.

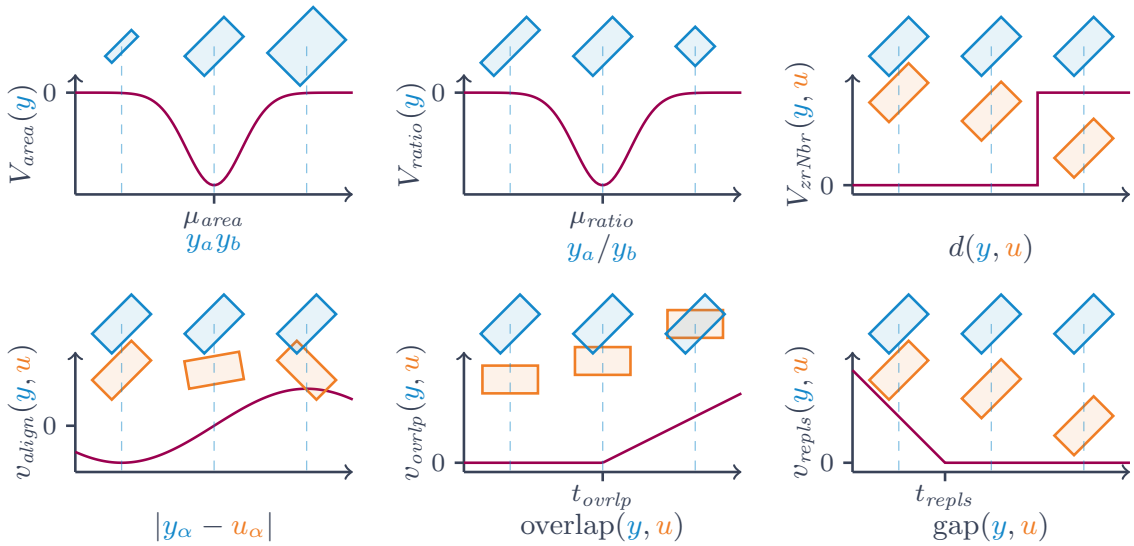


Figure 4.13: Illustration of the priors.

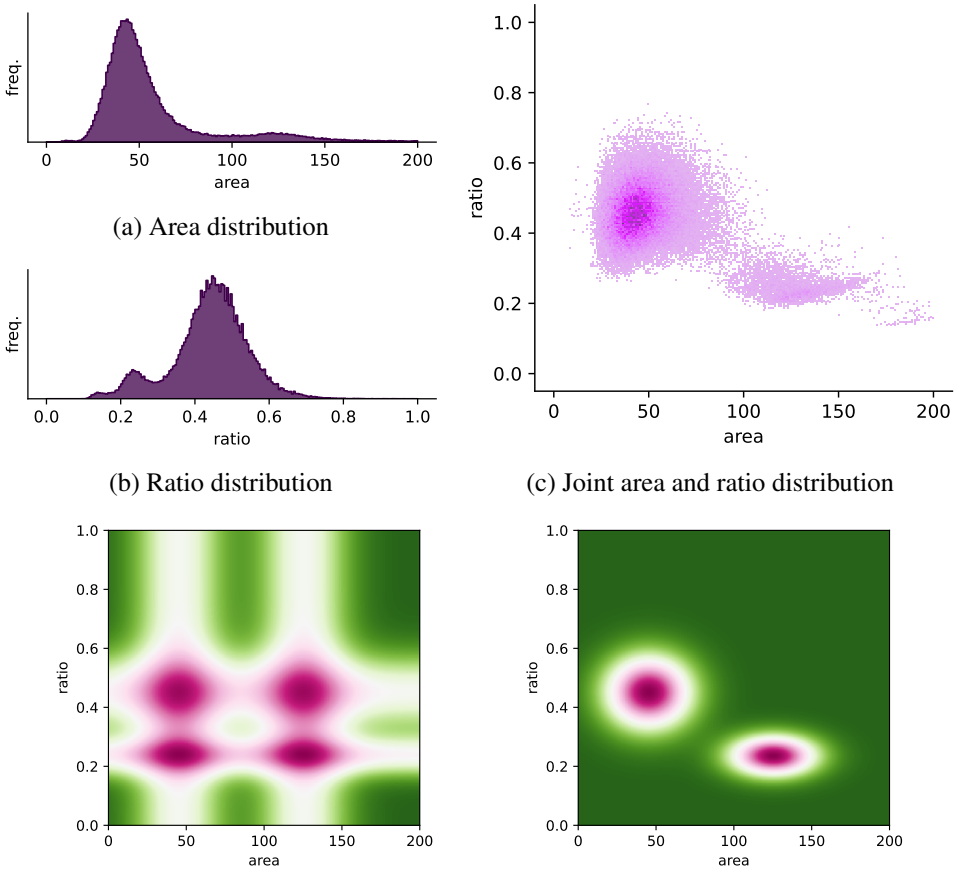
4.3.1.2 Joint area and ratio prior

The previous two energy terms V_{ratio} and V_{area} have a limitation: Let us consider objects that fall into two groups with typical ratio and area parameters $(\mu_{area,1}, \mu_{ratio,1})$ for the first group and $(\mu_{area,2}, \mu_{ratio,2})$ for the second. With previously defined area and ratio potentials, we could define two of each. However, this would create four attraction points, favoring points of geometries close to $(\mu_{area,1}, \mu_{ratio,1})$ and $(\mu_{area,2}, \mu_{ratio,2})$ as needed, but also shapes close to $(\mu_{area,1}, \mu_{ratio,2})$ and $(\mu_{area,2}, \mu_{ratio,1})$ that may not exist in reality. This is illustrated in Figure 4.14 (d).

To deal with this issue we propose a joint area and prior potential, that combine the two to favor points around $(\mu_{area}, \mu_{ratio})$ in the area-ratio subspace as illustrated in Figure 4.14 (e). This joint area and ratio prior is defined as:

$$V_{jntAR}(y) = -\exp\left(-\frac{\left(\frac{y_a}{y_b} - \mu_{ratio}\right)^2}{2\sigma_{ratio}^2} - \frac{(y_a y_b - \mu_{area})^2}{2\sigma_{area}^2}\right). \tag{4.34}$$

Measured joint distribution. Plotting the effective distribution of area and prior over the data, two modes can be identified for each (see Figure 4.14 (a) and (b)). However, using the V_{area} and V_{ratio} priors would create four attractive points. As can be seen in Figure 4.14 (c), the data exhibits only two modes (cars and trucks); justifying the use of two joint priors V_{jntAR}. One approach is to set parameters $(\mu_{area}, \mu_{ratio})$ manually from looking at the joint distribution in Figure 4.14 (c); however we propose in Section 5.2.2, to automatically estimate these parameters from the data, in a unified estimation procedure. We show the results of a model using these automatically estimated parameters in Section 6.2.2.



(a) Area distribution

(b) Ratio distribution

(c) Joint area and ratio distribution

(d) Four separate priors

(e) Two joint priors

Figure 4.14: Distribution of area and ratio over the training data. (a) and (b) show distributions of areas and ratios over the data, and (c) their joint distribution. In (d) we propose a setup with two ratio priors plus two area priors, while (e) uses two joint priors (green is high energy, pink is low).

4.3.2 Interaction priors

4.3.2.1 Aggregation operators

Some energy models for Point Processes sum all the interactions over the whole configuration as seen in (3.23). However, we find this creates some problems with attractive potentials: Objects sometimes accumulate the negative potentials from the interactions, as the number of interactions grows quadratically relative to the number of points. Meanwhile, isolated points remain at high energy.

For interaction priors (V_e, $e \in \xi_{prior-interact.}$), we propose using a carefully chosen aggregation operator $\mathcal{F}_e$ along a pair-interaction potential v as such:

$$V_e\left(y, \mathcal{N}^{\mathbf{y}}_{\{y\}}\right) = \mathcal{F}_e\left\{v_e(y,u),\ u \in \mathcal{N}^{\mathbf{y}}_{\{y\}}\right\},\ e \in \xi_{prior-interact.}. \tag{4.35}$$

To avoid the quadratic increase of interaction potentials, we choose $\mathcal{F}_e$ such that:

$$\forall A > 0,\ \mathcal{F}_e : \mathbb{P}([-A, A]) \mapsto [-A, A], \tag{4.36}$$

where $\mathbb{P}([-A, A])$ is the power set (set of all subsets) of $[-A, A]$; i.e. if v_e is bounded by A, V_e is too. For instance if $\mathcal{F}_e$ is a maximum, minimum, or average the above condition is met (as in Example 4.1.1). If it is a sum, then the model is equivalent to the above-mentioned (3.23) (up to a factor 2 on the interaction potentials, as each interaction would be counted twice).

In practice this helps to build some versatile priors; e.g. "the object needs to be aligned with at least one object", is easily enforceable with a $\min$ over all alignment potentials. Meanwhile, it helps avoid accumulation of points in attractive neighborhoods. We further discuss the choice of aggregation function $\mathcal{F}_e$ in Section G.

4.3.2.2 Non overlapping prior

The non-Bayesian approach mentioned in Section 3.2, may lead to the accumulation of objects in areas with low data potential without a strong prior on object not overlapping. For the objects we study — in most situations — no overlap is permitted. We propose a simple prior to penalize overlaps:

$$\begin{aligned} V_{ovrlp}(y, \mathcal{N}^{\mathbf{y}}_{\{y\}}) &= \max_{u \in \mathcal{N}^{\mathbf{y}}_{\{y\}}} \{v_{ovrlp}(y,u)\} \\ v_{ovrlp}(y,u) &= \max\left\{0, \frac{\mathrm{Area}(u \cap y)}{\min\{\mathrm{Area}(u), \mathrm{Area}(y)\}} - t_{ovrlp}\right\}, \end{aligned} \tag{4.37}$$

where t_{ovrlp} is a positive threshold that may allow some permeability between objects. We use a maximum to aggregate the interaction potentials v_{ovrlp} as we aim at penalizing overlap with *any* neighboring objects.

We find there is no simple closed-form formula for the area of intersection of two oriented rectangles. Thus, in practice, we approximate the intersection measure as described in Appendix D.

4.3.2.3 Alignment prior

The objects of interest (small vehicles), tend to be aligned: either parked with the same heading, or circulating in parallel lanes. We introduce a prior that favors aligned configurations

$$\begin{aligned} V_{align}\left(y|\mathcal{N}^{\mathbf{y}}_{\{y\}}\right) &= \min_{u\in\mathcal{N}^{\mathbf{y}}_{\{y\}}} \{v_{align}(y,u)\} \\ v_{align}(y,u) &= -\cos\left(|y_\alpha - u_\alpha| - t_{align}\right), \end{aligned} \tag{4.38}$$

where $t_{align} \in \{0, \pi/2\}$ is a parameter picked to either favor parallel objects (if 0) or orthogonal objects (if $\pi/2$). The aggregation operation over all the interaction potentials v_{align} is picked to be a minimum to favor being aligned with *at least* one object.

4.3.2.4 Repulsive and attractive priors

Vehicles often maintain a certain distance, especially when parked close. We introduce two concurring priors based on the gap between two rectangles (see (D.16) in Appendix D), one repulsive, the other attractive; with carefully picked thresholds t_{repls}, t_{attrc} we can favor objects within a specific distance range (see Figure 4.15):

$$\begin{aligned} V_{repls}\left(y|\mathcal{N}^{\mathbf{y}}_{\{y\}}\right) &= \max_{u\in\mathcal{N}^{\mathbf{y}}_{\{y\}}} \{v_{repls}(u,v)\} \\ v_{repls}(u,v) &= \max\left\{0, 1 - \frac{\mathrm{Gap}(y,u) - t_{repls}}{d_{max} - t_{repls}}\right\}, \end{aligned} \tag{4.39}$$

$$\begin{aligned} V_{attrc}\left(y|\mathcal{N}^{\mathbf{y}}_{\{y\}}\right) &= \min_{u\in\mathcal{N}^{\mathbf{y}}_{\{y\}}} \{v_{attrc}(u,v)\} \\ v_{attrc}(u,v) &= \max\left\{0, \frac{\mathrm{Gap}(y,u) - t_{attrc}}{d_{max} - t_{attrc}}\right\}, \end{aligned} \tag{4.40}$$

with $\mathrm{Gap}(y,u)$, measuring the smallest distance between the two shapes, as defined in (D.16).

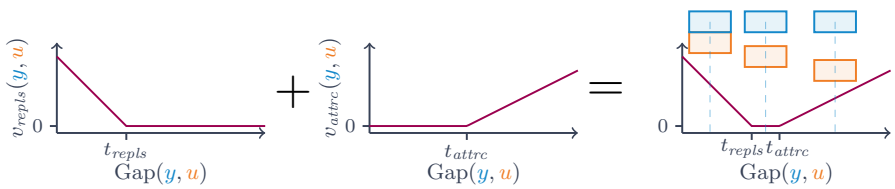


Figure 4.15: Combining attractive and repulsive priors.

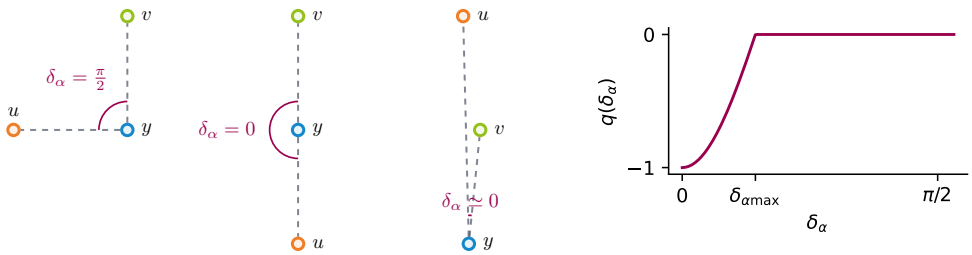
4.3.2.5 Zero neighborhood prior

For all the interaction priors defined above, the negative of positive potentials added to interacting points might offset the overall energy values of interacting points over points that have no interactions. For that matter we introduce a simple prior aimed at compensating this effect; the potential is 1 when no neighbor is present, else 0:

$$V_{zrNbr}(y, \mathcal{N}^{\mathbf{y}}_{\{y\}}) = \mathbb{1}_{|\mathcal{N}^{\mathbf{y}}_{\{y\}}|=0}(y). \tag{4.41}$$

Here we pick a value of 1 as it we will be able to scale it as needed using the energy weight w_{zrNbr} (see (4.3)). This extra term acts similarly as the offset term w_0, but applies only to objects without neighbors.

4.3.3 Triplet priors



(a) Example triplet alignments

(b) Plot of $q(\delta_\alpha)$

Figure 4.16: Illustration of triplet alignment prior.

In (Mabon et al., 2021) we propose a prior on triplets of points. In this work we consider a non-marked Point Process (i.e. no orientation for the detected objects); there is no possibility to define alignment with only two objects. Thus, we look at the angle formed by triplets of vehicles within the neighborhood $\mathcal{N}^{\mathbf{y}}_{\{y\}}$ of each object y in configuration $\mathbf{y}$. We observe that vehicles are most often stopped or circulating in lines; that is why we choose to enforce an angle of $0 \mod \pi$. This prior is no longer needed once dealing with marked objects, as the simple alignment prior already favors this type of configurations.

Denoting $\measuredangle uyv$ the value (in radiant) of the angle formed by the triplet of points u, y, v (as shown in Figure 4.16 (a)), we define $\delta_\alpha(y, u, v)$ the absolute angle, and the triplet alignment potential v_{triAl} as follows:

$$\delta_\alpha(y, u, v) = \min\{|\measuredangle uyv|, |\pi - \measuredangle uyv|\}, \tag{4.42}$$

$$v_{triAl}(y, u, v) = q_{triAl}\left(\delta_\alpha(y, u, v)\right). \tag{4.43}$$

To turn the absolute difference δ_α into a potential, we use the following quality function, parametrized by $\delta_{\alpha\,\max}$, which sets a margin for the angle as illustrated in Figure 4.16 (b):

$$q_{triAl}(\delta) = -\frac{1}{\delta^2_{\alpha\,\max}}\left(\frac{1 + \delta^2_{\alpha\,\max}}{1 + \min\{\delta, \delta_{\alpha\,\max}\}^2} - 1\right) \tag{4.44}$$

Finally, the potential for point y and its neighborhood $\mathcal{N}^{\mathbf{y}}_{\{y\}}$ takes the minimum over all potentials; this favors points that are aligned within *at least* one triplet:

$$V_{triAl}(y, \mathcal{N}^{\mathbf{y}}_{\{y\}}) = \begin{cases} \min\limits_{\substack{u,v \in \mathcal{N}^{\mathbf{y}}_{\{y\}} \\ u \neq v}} \{v_{triAl}(y, u, v)\} & \text{if } |\mathcal{N}^{\mathbf{y}}_{\{y\}}| \geq 2 \\ 0 & \text{else.} \end{cases} \tag{4.45}$$

4.4 Resulting model and discussions

4.4.1 Model pipeline

With all potentials defined above, we can compute the energy for any configuration $\mathbf{y}$. We compute the energy as follows:

1. For a given image $\mathbf{X}$, pre-compute the potential maps once:
   (a) Pass the image $\mathbf{X}$ through the CNN model to produce $\widehat{\mathbf{Z}}_{pos}$ and $\widehat{\mathbf{Z}}_{\kappa}$.
2. For any configuration $\mathbf{y} \in \mathcal{Y}$ relative to image $\mathbf{X}$:
   (a) Compute data terms for each $y \in \mathbf{y}$ given $\widehat{\mathbf{Z}}_{pos}$ and $\widehat{\mathbf{Z}}_{\kappa}$.
   (b) Compute each prior term for each $y \in \mathbf{y}$; each depends at most on $\mathcal{N}^{\mathbf{y}}_{\{y\}}$.
   (c) Combine all terms into $U(\mathbf{y}, \mathbf{X})$ as in (4.2) (4.3).

Within a MCMC framework, in which the energy has to be computed for different configurations with the same image, only steps 2 (a) to (c) have to be repeated as the image and inferred tensors are unchanged (no need to redo step 1 each time). We illustrate this pipeline in Figure 4.17: the pre-computed block corresponds to the results from step 1 that needs to be computed only once per image. The blocks $\kappa \in \{a, b, \alpha\}$, $e \in \xi_{prior-interact.}$, $e \in \xi_{prior-point}$ are repeated for each respective mark, point prior and interaction prior. For the interaction prior the distance matrix is only computed once for all interaction priors. For more precision on the actual implementation of this pipeline we refer the reader to Appendix I.

4.4.2 Discussions

4.4.2.1 Continuity

In Section 3.3.3, we mention that the energy function has to be Lipschitz-continuous so that diffusion can be used to sample configurations. The data energy terms are made continuous through interpolation, while the local prior terms are continuous by construction. The interaction energies v_e are continuous too. However, object y moving continuously into (or out of) the neighborhood of another point u creates a discontinuity as it crosses the boundary of the neighborhood $\mathcal{N}^{\mathbf{y}}_{\{u\}}$. It is made obvious when simplifying to only two points, we get:

$$V_e(y, \{u\}) = \mathbb{1}_{d(u,y) \leq d_{max}}(y) v_e(y, u). \tag{4.46}$$

Supposing v_e is Lipschitz-continuous, the above is made discontinuous at $d(u, y) = d_{max}$ by the step function induced by the indicator $\mathbb{1}$. As such our energy model is Lipschitz-continuous for one object y everywhere in $\mathcal{S} \times \mathcal{M}$ except in set $\cup_{u \in \mathbf{y} \setminus \{y\}} \{x,\ x \in \mathcal{S},\ d(x, u) = d_{max}\}$, i.e. the union of boundaries of each neighborhood, as shown in mauve in Figure 4.18[3].

[3] Within a neighborhood (or intersection of neighborhoods) there are no discontinuities.

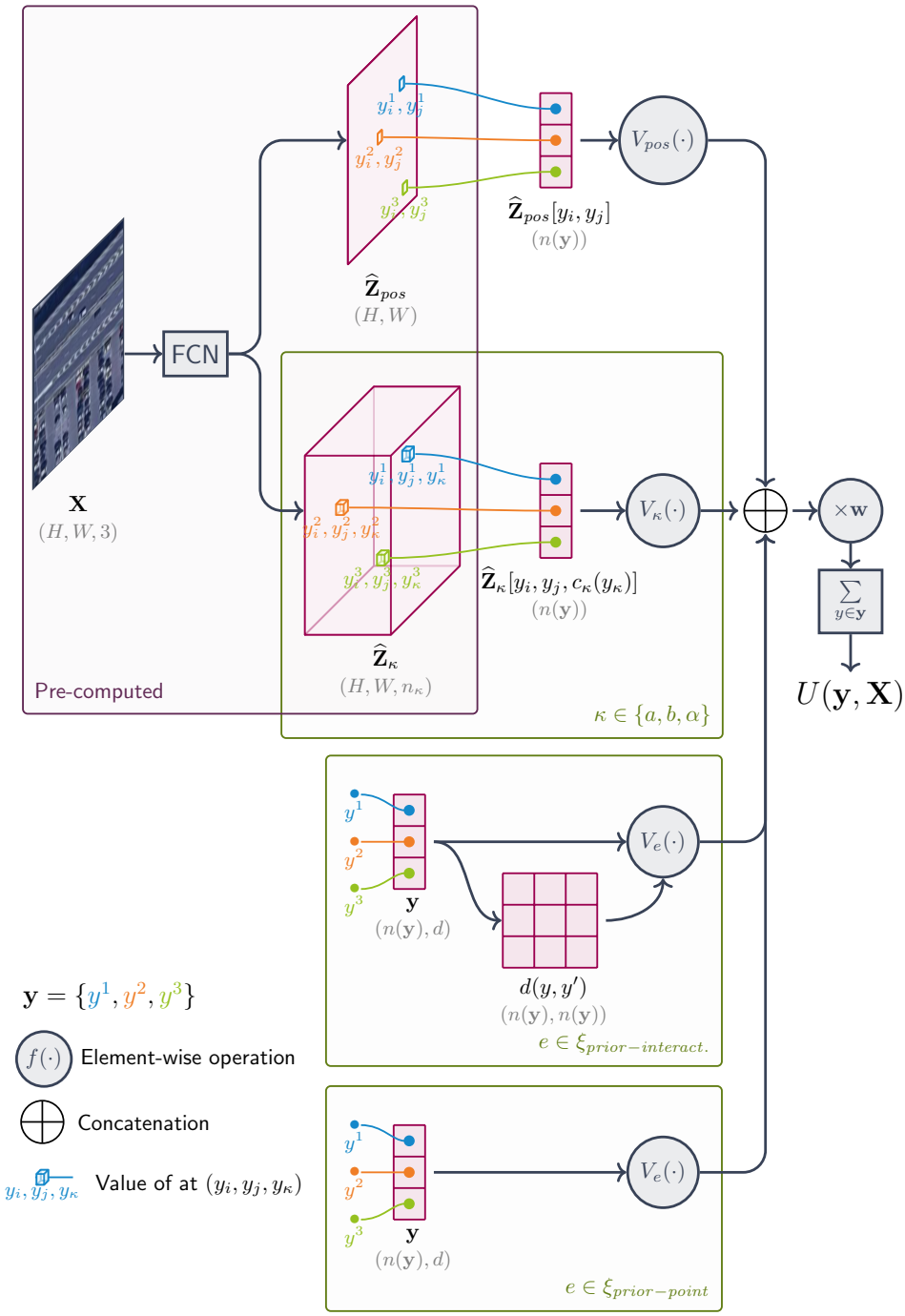


Figure 4.17: Energy model pipeline. The *pre-computed* section can be reused for computing different $\mathbf{y} \in \mathcal{Y}$ for a given image $\mathbf{X}$. $d(y, y')$ is the distance matrix.

This could be solved by incorporating a smooth weighting function over each interaction potential v_e, rewriting (4.35) as:

$$V_e\left(y, \mathcal{N}^{\mathbf{y}}_{\{y\}}\right) = \mathcal{F}_e\left\{\varsigma(d(u,y))v_e(y,u),\ u \in \mathcal{N}^{\mathbf{y}}_{\{y\}}\right\}, \tag{4.47}$$

with $d(u,y)$ the distance between y and u, and ς a Lipschitz-continuous function such that $\varsigma(d) = 0$ for $d \geq d_{max}$ and $\varsigma(d) > 0$ for $d \in [0, d_{max}]$.

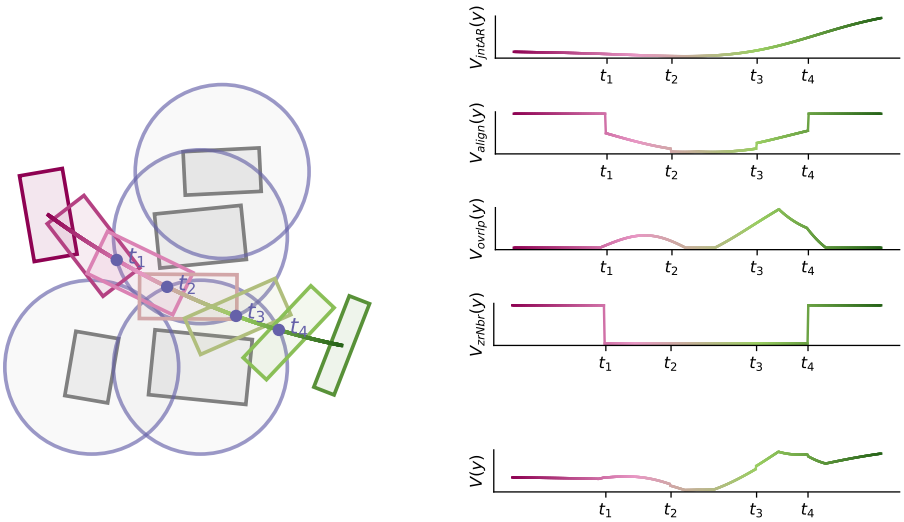


Figure 4.18: Energy continuity illustrated for a simple example: on the left, we move continuously one object through space and marks (trajectory from violet to green). The energy terms V_e are plotted on the right. The total energy of a point (bottom plot on the right) is $V(y) = V_{jntAR} + V_{align} + 8V_{ovrlp} - 0.5V_{zrNbr}$. In mauve, we represent the discontinuity surface in $\mathcal{S}$, with $t_1, \ldots, t_4$ the points where the path of the sliding object intersects. The interaction radius d_{max} is set at a low value for illustration purposes.

4.4.2.2 Mixing of energy terms

The classical approach for mixing energy terms via a weighted sum (see (4.3)) has its limitations as illustrated in Section 4.3.1.2. In (Mabon et al., 2022a) we propose using a *hierarchical* model, where the priors are only taken into account if the position potential reaches a certain threshold:

$$V\left(y, \mathbf{X}, \mathcal{N}^{\mathbf{y}}_{\{y\}}\right) = w_{pos}V_{pos}(y) + \mathbb{1}_{V_{pos}(y)<t_{pos}}\left(\sum_{e\in\xi,\, e\neq pos} w_e V_e\left(y, \mathbf{X}, \mathcal{N}^{\mathbf{y}}_{\{y\}}\right)\right).$$

The latter induces some more discontinuities in the energy model that do not go well with the jump-diffusion sampling (Geman & Hwang, 1986) (see Section 3.3.3).

Alternatively, we can see the function V can be any function that transforms the vector of all energy terms into a scalar energy:

$$\begin{aligned} V &: \mathbb{R}^{|\xi|} \to \mathbb{R} \\ \mathbf{v}_y &\mapsto V(\mathbf{v}_y), \quad \mathbf{v}_y = \begin{bmatrix} V_{e_1}(y) & \ldots & V_{e_{|\xi|}}(y) \end{bmatrix}^\top \end{aligned} \tag{4.48}$$

For instance one could use a Multi Layer Perceptron (MLP) parametrized by θ, such that $V(\mathbf{v}_y) = \mathrm{MLP}_\theta(\mathbf{v}_y)$. As the MLP is a universal approximator (given enough hidden layers) (Hornik, Stinchcombe, & White, 1989) it could better approach the optimal mixing function. The criterion for *optimality* will be discussed in Section 5.2.

4.4.2.3 Towards generic (learnable) energy terms

In this chapter we proposed a series of priors with parameters to either set manually of estimate from the data. The selection and design of the energy prior functions to put in the total energy U is by itself a prior on the model, as it restricts the space of energies $U(\ldots,\theta)$ generated by parameters θ. As prospective work, we propose in Appendix G to push further the parametrization of the energy priors in order to bypass the function-design-induced constraints, while allowing to build energy functions that better fit the data. We first go through a design of per-point priors with Multi Layer Perceptron (MLP), and then show that attention mechanisms can be used to learn a generic aggregation function for interaction priors.

4.5 Conclusion

In this Chapter we built the Point Process through its energy function, using a CNN output for the data term and introducing multiple priors. We summarize our contributions as follows:

- A first data model for non-marked Point Processes, using a *heatmap* inferred by a CNN (Section 4.2.2.1).
- We further refine this approach by inferring a map of vectors and computing the divergence; it achieves better instance separation (Section 4.2.2.2).
- Then we propose a way to interpret most CNN outputs as energies that we can use on our model, both for object positions and marks (Section 4.2.2.3).
- All the above data terms based on a CNN output benefit from replacing the contrast measures computation by a simple value lookup and interpolation (Section 4.2.2.2 and 4.2.3), thanks to the pre-computation of energy maps (Section 4.4.1).
- Joint priors on object parameters to avoid favoring non-existing shapes when dealing with multiple shape modes (e.g. cars and trucks) (Section 4.3.1.2).
- Carefully chosen aggregation functions on interaction priors to avoid energy explosion and favor specific configurations (e.g. alignment with *at least* one other object, no overlap with *any* other object) (Section 4.3.2.1).
- As prospective work, we propose a way to solve discontinuities in object interactions (Section 4.4.2.1) and new models to mix the energy terms (Section 4.4.2.2).

CHAPTER 5

Point Processes as Energy Based Models

Dans ce chapitre, nous passons en revue les procédures d'échantillonnage et d'estimation des paramètres pour le Processus Ponctuel défini au chapitre 4. Nous proposons d'exploiter les cartes de potentiels fournies par le CNN, en élaborant un noyau de perturbation local et un noyau de naissance, tous deux construits à partir des données. De plus, nous proposons un schéma de calcul parallèle modifié pour échantillonner de nouveaux points et perturbations qui se concentre sur les zones de l'image à forte densité d'objets. Pour l'estimation des paramètres du Processus Ponctuel, nous proposons d'abord une approche fondée sur les séparateurs à vaste marge pour résoudre les limitations des méthodes précédentes. La deuxième approche et contribution majeure consiste à appliquer des méthodes d'apprentissage de modèles basés sur l'énergie pour estimer tous les paramètres du Processus Ponctuel au sein d'un algorithme unifié par divergence contrastive. Enfin, nous proposons un score d'objet pour le calcul de métriques utilisant l'intensité de Papangelou.

In this chapter we go through the sampling and parameter estimation procedures for the Point Process defined in Chapter 4. We propose to leverage the potential maps provided by the CNN, by building a data driven local perturbation kernel and birth kernel. Moreover, we propose a modified parallel computation scheme to sample new points and perturbation that focuses on areas of the image with high object density. For the estimation of the Point Process parameters, we first propose an approach based on Support Vector Machines to alleviate limitations of previous methods. The second approach and major contribution of this thesis, consists in applying Energy Based Model learning methods to estimate all parameters of the Point Process within one unified algorithm through contrastive divergence. Finally, we propose an object score for metrics computation based on the Papangelou intensity.

5.1 Sampling **95**
5.1.1 Data driven kernels 95
5.1.1.1 Local perturbation based on data 95
5.1.1.2 Birth with density 96
5.1.1.3 Jump diffusion 98
5.1.2 Parallel sampling 99
5.1.3 Sampling method 101
5.2 Parameters estimation **102**
5.2.1 Learning energy weights with local perturbations 103
5.2.1.1 Linear Programming 103
5.2.1.2 Margin maximization 104
5.2.2 Maximum likelihood learning 105
5.2.2.1 Contrastive divergence 106
5.2.2.2 Replay buffer 107
5.2.2.3 Effect of temperature on samples 108
5.2.2.4 Algorithm 110
5.2.2.5 Discussions 111
5.3 Papangelou intensity as a score **112**
5.3.1 Computing the detection score 112
5.3.2 Contrastive divergence loss and Papangelou intensity 113
5.4 Conclusion **114**

5.1 Sampling

In Chapter 4 we built our Point Process model through the definition of its energy function U. This section presents the improvements over the sampling procedures introduced in Section 3.3. We leverage the pre-computed energy potential maps provided by the CNN model to build more efficient sampling. First by using local perturbations based on the potential map adapted from (Ortner, 2004; Descombes, 2013), which we published in (Mabon et al., 2021). We adapt the birth map from (Lacoste et al., 2005) to our model, and propose using a truncated energy model to sample new points (Mabon et al., 2023a). We then adapt parallel Point Process sampling from (Verdié & Lafarge, 2012) with cell picking based on the potential maps, as published in (Mabon et al., 2023a) and to be submitted in (Mabon et al., 2023). Finally, the resulting sampling procedure is presented in Algorithm 5.1, based on Jump Diffusion while leveraging automatic gradient computation.

5.1.1 Data driven kernels

5.1.1.1 Local perturbation based on data

The local perturbation kernel introduced in Section 3.3.2.3 usually uses a zero centered Gaussian distribution to generate the perturbation δ on a point y on configuration $\mathbf{y}$. When computing the total energy we first infer a position potential map $\mathbf{V}_{pos}$ so that we simply have to sample values from it to get V_{pos}. It is defined as:

$$\mathbf{V}_{pos}[\rho] = V_{pos}(\rho, \mathbf{X}), \tag{5.1}$$

for any pixel $\rho \in [\![0, H]\!] \times [\![0, W]\!]$. Similarly, we can define $\mathbf{V}_\kappa$ for each mark $\kappa \in \{a, b, \alpha\}$. Given that we have access to the pre-computed position potential map, we could intuitively use it to make better move propositions within the Markov chain.

In order to propose more relevant perturbations, we propose to use this potential map. From (3.64) we have:

$$\alpha(\mathbf{y}, \mathbf{x}) = \min\left\{1, \frac{h(\mathbf{x})g'(\delta')}{h(\mathbf{y})g(\delta)}\right\},$$

Where $g(\delta)$ is the density for picking perturbation vector δ and $g'(\delta')$ the density of the reverse move.

In (Mabon et al., 2021) we propose a data-based translation kernel adapted from (Ortner, 2004; Descombes, 2013) using the position potential map $\mathbf{V}_{pos}$. This proposal is applied to a non-marked Point Process (thus only performing translations) but can be generalized to a marked Point Process too.

For a point y in configuration $\mathbf{y}$, $y \in \mathcal{S}$, we generate a translation vector $\delta \in [\![-\delta_{max}, \delta_{max}]\!]^2$, with $\delta_{max} \in \mathbb{N}^+$ the maximum move distance.

Vector δ is sampled from $\delta \sim g(\delta)$, with:

$$g(\delta) = \frac{\exp(-w_{pos}\mathbf{V}_{pos}\left[\lfloor y \rfloor + \delta\right])}{\sum_{\delta' \in [\![-\delta_{max}, \delta_{max}]\!]^2} \exp(-w_{pos}\mathbf{V}_{pos}\left[\lfloor y \rfloor + \delta'\right])}, \tag{5.2}$$

With $\lfloor y \rfloor$ the integer coordinates vector of point y. In practice, we build a tensor $\mathbf{D}_{y,\mathbf{x}}$ of size $(2\delta_{max}+1, 2\delta_{max}+1)$ such that:

$$\begin{aligned} \mathbf{D}_{y,\mathbf{x}}[i,j] &= \frac{1}{Z}\exp\left(-w_{pos}\mathbf{V}_{pos}\left[\lfloor y \rfloor + \begin{bmatrix} i \\ j \end{bmatrix}\right]\right), \; i,j \in [\![-\delta_{max}, \delta_{max}]\!]^2, \\ Z &= \sum_{\delta' \in [\![-\delta_{max},\delta_{max}]\!]^2} \exp(-w_{pos}\mathbf{V}_{pos}\left[\lfloor y \rfloor + \delta'\right]). \end{aligned} \tag{5.3}$$

2D tensor $\mathbf{D}_{y,\mathbf{x}}$ defines a density over the raster space that approximates g. Note that for the return move from $y' = y + \delta$ to δ, the density is $g'(\delta') = \mathbf{D}_{y',\mathbf{x}}[-\delta_i, -\delta_j]$.[1] To ensure this kernel can access all real-valued positions within δ_{max}, δ is drawn as $\delta \sim \mathbf{D}_{y,\mathbf{x}}$, then we set:

$$\delta \leftarrow \delta + \begin{bmatrix} \mathcal{U}([0,1]) \\ \mathcal{U}([0,1]). \end{bmatrix} \tag{5.4}$$

The change in densities g and g' gets balanced out in $\alpha(\mathbf{y}, \mathbf{x})$. The resulting kernel has the following acceptance ratio and is illustrated in Figure 5.1:

$$\alpha(\mathbf{y}, \mathbf{x}) = \min\left\{1, \frac{h(\mathbf{x})\mathbf{D}_{y',\mathbf{x}}[-\lfloor \delta_i \rfloor, -\lfloor \delta_j \rfloor]}{h(\mathbf{y})\mathbf{D}_{y,\mathbf{x}}[\lfloor \delta_i \rfloor, \lfloor \delta_j \rfloor]}\right\}.$$

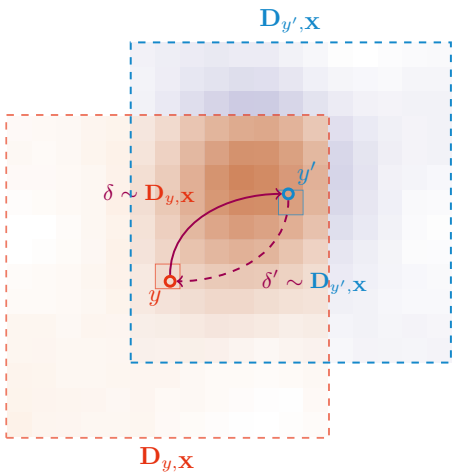


Figure 5.1: Data-driven translation kernel applied to y. Perturbation δ is generated from the 2D density map $\mathbf{D}_{y,\mathbf{x}}$. The reverse mode follows density $\mathbf{D}_{y',\mathbf{x}}$ which is centered on y'. The red and blue centered squares represent the integer positions $\lfloor y \rfloor$ and $\lfloor y' \rfloor$ for y and y'.

5.1.1.2 Birth with density

The uniform birth kernel introduced in Section 3.3.2.3 proposes points uniformly in $\mathcal{S} \times \mathcal{M}$. However, the density and scattering of objects in the image might not be uniform, making a lot of

[1] We use negative array indexing for simplicity, one can either offset all indexing by δ_{max} for all indexing to be positive, or use the Python negative indexing convention $\mathbf{A}[-i] = \mathbf{A}[\text{len}(\mathbf{A}) - i]$.

the object proposals superfluous. In their work, (Crăciun, 2015) use a binary map to restrict the space $\mathcal{S}$ into a smaller subspace; boats being most likely on water, the binary map corresponds to a detection of water bodies. In (Lacoste et al., 2005), the authors use a map of potentials to build a proposal density for the birth kernel. This proposal density is built from a pre-computation of the contrast measure for numerous possible object positions and parameters (i.e. data potentials in that case).

A density birth kernel Q_{B} that proposes a new point u to configuration $\mathbf{y}$ with density $d(u)$, has the following Green ratio:

$$\begin{aligned} r(\mathbf{y}, \mathbf{y} \cup \{u\}) &= \frac{p_{\mathrm{D}}}{p_{\mathrm{B}}} \frac{Q_{\mathrm{D}}(\mathbf{y} \cup \{u\} \to \mathbf{y})}{Q_{\mathrm{B}}(\mathbf{y} \to \mathbf{y} \cup \{u\})} \frac{h(\mathbf{y} \cup \{u\})}{h(\mathbf{y})} \\ &= \frac{p_{\mathrm{D}}}{p_{\mathrm{B}}} \frac{\lambda}{d(u)(n(\mathbf{y})+1)} \frac{h(\mathbf{y} \cup \{u\})}{h(\mathbf{y})}. \end{aligned} \tag{5.5}$$

As in (Lacoste et al., 2005), we propose using the data energy to drive this density d using the pre-computed position and mark tensors $\mathbf{V}_{pos}$ and $\mathbf{V}_{\kappa}$.

Discrete points space. The position and mark energy maps $\mathbf{V}_{pos}$ and $\mathbf{V}_{\kappa}$, are *raster* maps (i.e. defined over integer positions), meanwhile the density has to be defined on $\mathcal{S} \times \mathcal{M}$. As it is simple to sample in a discrete finite space, we define a discrete equivalent of the space $\mathcal{S} \times \mathcal{M}$ to sample in. The discretization stems from the pixel space in the image, and mark discretization performed in Section 4.2.3:

$$\begin{aligned} \mathcal{S}_d &= [\![0, H]\!] \times [\![0, W]\!], \\ \mathcal{M}_d &= \prod_{\kappa \in \{a,b,\alpha\}} \{c(\kappa_{\max} - \kappa_{\min}) + \kappa_{\min},\ c = 0, \ldots, n_{\kappa}\}. \end{aligned} \tag{5.6}$$

Sampling from truncated energy. Ideally we would propose a point u to the current configuration $\mathbf{y}$, sampled with the marginal density $p(u|\mathbf{y})$ as is done within Gibbs sampling. While we cannot easily compute the marginal density of u knowing $\mathbf{y}$ we can approximate it through the position potential of point y, with $p(u|\mathbf{y}) \propto h(\mathbf{y} \cup \{u\})/h(\mathbf{y})$:

$$\frac{h(\mathbf{y} \cup \{u\})}{h(\mathbf{y})} = \exp\left(U(\mathbf{y}, \mathbf{X}) - U(\mathbf{y} \cup \{u\}, \mathbf{X})\right) \tag{5.7}$$

$$= \exp\left(\underbrace{-V(u, \mathcal{N}^{\mathbf{y}}_{\{u\}}, \mathbf{X})}_{\text{added point energy}} + \sum_{y \in \mathcal{N}^{\mathbf{y}}_{\{u\}}} \underbrace{V(y, \mathcal{N}^{\mathbf{y}}_{\{y\}}, \mathbf{X}) - V(y, \mathcal{N}^{\mathbf{y} \cup \{u\}}_{\{y\}}, \mathbf{X})}_{\text{energy variation of existing points}} \right) \tag{5.8}$$

$$\simeq \exp\left(-V(u, \mathcal{N}^{\mathbf{y}}_{\{u\}}, \mathbf{X})\right) \tag{5.9}$$

$$\simeq \exp\left(-w_{pos} V_{pos}(u, \mathbf{X}) - \sum_{\kappa \in \{a,b,\alpha\}} w_{\kappa} V_{\kappa}(u, \mathbf{X})\right) \tag{5.10}$$

In (5.9) we first neglect the energy variations of the existing points, moreover the cardinality of $\mathcal{N}^{\mathbf{y}}_{\{u\}}$ is limited by the number of object in a d_{max} radius. In (5.10) we simplify the energy of point u to the easy-to-sample energy terms: the position and marks potentials. While for instance the

non-overlap prior is of great importance, it makes the sampling of a new point u more computationally expensive. That way our truncated law for point proposal only depends on the image $\mathbf{X}$.

We derive the following density for sampling new points ρ in discrete space $\mathcal{S}_d \times \mathcal{M}_d$:

$$\widetilde{d}(\rho) = \frac{1}{Z} \exp\left(-\frac{w_{pos}\mathbf{V}_{pos}[\rho] + \sum_{\kappa\in\{a,b,\alpha\}} w_\kappa \mathbf{V}_\kappa[\rho]}{T_d} \right), \tag{5.11}$$

where Z is a normalizing constant (which we can compute easily as $\widetilde{d}$ is defined over a discrete finite space). Density $\widetilde{d}(\rho)$ can be computed for all pixels ρ in $\mathcal{S}_d \times \mathcal{M}_d$, as the normalizing constant is simply a sum over HWn_κ^3 elements; thus this density can be easily sampled from. Temperature parameter $T_d > 0$ is used to influence the density: high T_d tends towards a uniform distribution, (i.e. getting closer to the uniform birth kernel); low T_d makes the birth kernel only propose low data potential points (i.e. likely higher acceptance rate at the cost of less space exploration). Eventually, to sample a point u in continuous space $\mathcal{S} \times \mathcal{M}$ we proceed as follows:

1. Sample a pixel ρ in $\mathcal{S}_d \times \mathcal{M}_d$ with density $\widetilde{d}$.
2. Sample u in ρ uniformly ($u \in \mathcal{S} \times \mathcal{M}$).

This resulting sampling density d of u is:

$$d(u) = \frac{1}{|\rho_u|}\widetilde{d}(\rho_u), \tag{5.12}$$

with ρ_u the pixel in $\mathcal{S}_d \times \mathcal{M}_d$ containing u, and $|\rho_u|$ the measure of pixel ρ_u ($= 1$ if the unit of measure is set to a pixel).

5.1.1.3 Jump diffusion

For sampling the Point Process we use the Jump Diffusion mechanism introduced in Section 3.3.3. It alternates diffusion with birth and death kernels to explore the whole configuration space $\mathcal{Y}$. As a reminder the discrete diffusion as introduced (3.68) gives:

$$\mathbf{y}_{t+1} = \mathbf{y}_t - \gamma\nabla U(\mathbf{y}_t) + \sqrt{2T(t)}w_t,\ w_t \sim \mathcal{N}(0,\gamma).$$

The gradient of the energy ∇U is computed thanks to an automatic differentiation engine (see following paragraph on implementation 5.1.2), lifting the burden of manual gradient derivation, and allowing to implement and test new energy functions easily (as long as those are defined using the set of operations for which the automatic differentiation is available).

Diffusion can be seen as a continuation of the local perturbation based on data. While with the latter we approximate the energy locally with the position potential to sample a new position for a point, the diffusion approximates the energy locally as $U(\mathbf{y}+\delta) \simeq U(\mathbf{y}) + \frac{\partial U(\mathbf{y})}{\partial \mathbf{y}}\delta$ and samples a small perturbation in the vicinity of $\mathbf{y}$.

For most of the energy model, the gradient computation is well-defined: for both single point priors ($V_e, e \in \xi_{prior-point}$) and interaction priors ($V_e, e \in \xi_{prior-interact.}$) minus the discontinuities discussed in Section 4.4.2.1. For the data terms ($V_e, e \in \xi_{data}$), the computation of the energy

corresponds to looking up a value at a position in a tensor $\mathbf{Z}$. Thanks to the interpolation used to obtain values for float pixel coordinates, the operation is made differentiable, both relative to the configuration $\mathbf{y}$ and relative to the potential map $\mathbf{Z}$[2].

In practice, as parallelization requires some maximum spatial displacement on points (see next Section 5.1.2), the step $\Delta C_c(\mathbf{x}_t)$ from $\mathbf{x}_t$ to $\mathbf{x}_{t+1}$ (Alg. 5.1 line 9) is clipped; For each point $y \in \mathbf{x}_t$ updated to y', we bound the i and j components of $\Delta C_c(\mathbf{x}_t)$ within $[-\delta_{max}, \delta_{max}]$:

$$\Delta C_c(\mathbf{x}_t)_i \leftarrow \min\left(\delta_{max}, \max\left(-\delta_{max}, \Delta C_c(\mathbf{x}_t)_i\right)\right),$$

ensuring $|y_i - y'_i| \leq \delta_{max}$, and similarly for j.

5.1.2 Parallel sampling

In Section 3.3.2.4 we define the basics of parallelization of the Point Process samples, which allows performing perturbations over multiple independent cells at once. With parallelization, we aim to take advantage of the spatial Markovianity of the Point Process; i.e., points further than a certain distance have no effect on each other's energy. Contrary to (Verdié & Lafarge, 2014) we use a constant cell size instead of quadtrees.

We build each cell to be a square of size d_c, each cell is assigned to a mic-set s, such that no two cells in set s are neighbors (considering the 8-connectivity). To build such cells on a 2D plane requires four colors as illustrated in Figure 5.2. We demonstrate in Appendix E.1 that the minimum cell size necessary to maintain independence of the moves in cells of the same mic-sec needs to be such that:

$$d_c \geq 2d_{max} + 2\delta_{max}. \tag{5.13}$$

To simulate the MCMC, one could pick a mic-set s uniformly and run the perturbations in parallel over each cell c in set s. However, we might do more perturbations than necessary in some low density areas of the image. Thus, we propose a selection scheme for the cells to compute, that aims at reproducing sampling density d introduced in Section 5.1.1.2. For a given configuration $\mathbf{y}$ we do:

1. Pick a kernel Q_m with probability $Q_m(\mathbf{y}, \mathcal{Y})$,
2. pick one mic-set s with probability $p(s)$,
3. keep each cell in mic-set s with probability $p(c|s)$ to form $\tilde{s}$
4. for every cell c in set $\tilde{s}$ run kernel Q_m restricted to c (denoted $Q_{c,m}$).

The process of cell selection (step 3), allows to limit the number of cells processed at once on big images, hence limiting the computational cost while maintaining the desired sampling density d.

[2]The heavy lifting of interpolation and differentiation is handled by the Pytorch library, here the `grid_sample` operation (*Torch.Nn.Functional.Grid_sample — PyTorch 2.0 Documentation*, n.d.)

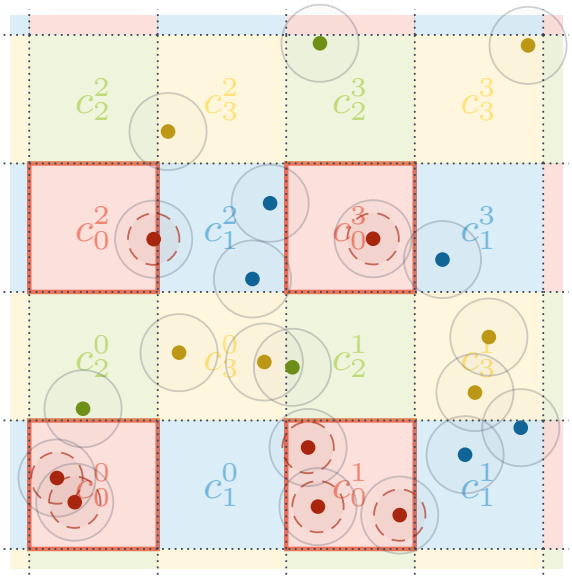


Figure 5.2: Mic sets and cells for a Point Process. Each point is represented in the color of the cell it belongs to. Cell c_s^k is the kth cell of mic-set s. We highlight cells of mic-set 0, for which we represent the radius in which each point can be moved δ_{max} (colored dashed circle). Gray semi-transparent circles represent each point interaction radius d_{max} (neighborhood $\mathcal{N}_{\{y\}}^{\mathbf{y}}$).

Probabilities of picking cells. We now build the probabilities $p(s)$ and $p(c|s)$ with which cells are picked, with the aim to fit the sampling density d introduced in (5.12). For a density birth kernel $Q_{c,\mathrm{B}}$, the kernel density over the whole image Q_{B} is now:

$$\underbrace{\frac{d(u)}{\lambda}}_{Q_{\mathrm{B}}(\mathbf{y}\to\mathbf{y}\cup\{u\})} = p(s_u)p(c_u|s_u)Q_{c_u,\mathrm{B}}(\mathbf{y}\to\mathbf{y}\cup\{u\}), \tag{5.14}$$

with c_u and s_u the cell and mic-set containing u respectively. For the above defined procedure we have the following conditions stemming from it:

- For step 2, we need $p(s) \in [0,1]$ and $\sum_s p(s) = 1$.
- For step 3, we need $p(c|s) \in [0,1]$.
- $Q_{c_u,\mathrm{B}}$ is of the form $Q_{c_u,\mathrm{B}}(\mathbf{y}\to\mathbf{y}\cup\{u\}) = \frac{d_c(u)}{\lambda}$, with $d_c(u)$ density we can sample from, with $\int_{u\in c} d_c(u)du = 1$.

We propose the following solution that verifies (5.14) and the conditions mentioned above:

$$p(s) = \int_{v\in s} d(v)dv = d(s), \tag{5.15}$$

$$p(c|s) = \frac{\int_{v\in c} d(v)dv}{\int_{v\in s} d(v)dv} = \frac{d(c)}{d(s)}, \tag{5.16}$$

$$d_c(u) = \frac{d(u)}{\int_{v\in c} d(v)dv} = \frac{d(u)}{d(c)}, \tag{5.17}$$

denoting (by extension) $d(s)$ and $d(c)$ the density d integrated over all cells of mic-set s and cell c respectively.

Skewing distributions for more parallelism. While the procedure to pick cells $\tilde{s}$ allows sampling points with density d, by construction the expected number of cells in $\tilde{s}$ is 1. In practice, we skew probability $p(c|s)$ to ensure more cells are kept for simulation, by replacing $p(c|s)$ by $p'(c|s) = \min(1, np(c|s))$, with $n > 1$. It increases the amount of parallel computations of cells (thus reducing simulation time) at the cost of deteriorating the approximation of global sampling density d.

Acceptance ratio in cells. From the mutual independence of cells we get the following property: the acceptance ratio for a move in cell c can be simplified (at least) as such:

$$\alpha(\mathbf{y}, \mathbf{y}') = \alpha(C_{\bar{c}}(\mathbf{y}), C_{\bar{c}}(\mathbf{y}')), \tag{5.18}$$

where $C_c(\mathbf{y})$ is the set of points from configuration $\mathbf{y}$ in cell c, and $\bar{c}$ the 8 cells neighboring c and c itself (thus by extension $C_{\bar{c}}(\mathbf{y})$ is points in $\mathbf{y}$ inside c or its neighbors). More details in Appendix E.2. This allows making the computation of acceptance ratio local, thus easy to perform in parallel.

Implementation. Instead of using multiple CPU threads as in (Crăciun, 2015), we defer the parallelization to the implicit multithreading capabilities of GPU (Graphical Processing Unit) when performing matrix/tensor operations. In short, we write the whole energy model in terms of tensor operations, using an extra batch dimension to process multiple cells at once. Libraries such as (*PyTorch*, n.d.) combined with CUDA (*CUDA*, 2017), allow to transparently run tensor operations over multiple threads within a GPU. On top of it, granted operations are defined properly, PyTorch allows to automatically compute the gradient of the energy model; either relative to parameters θ — which we use for parameters estimation in Section 5.2.2 — or to the configuration $\mathbf{y}$, which are used for diffusion in Section 3.3.3. We go further into details about the data structure and implementation of our approach in Appendix I.

5.1.3 Sampling method

The sampling method is summarized into Algorithm 5.1. The input for this algorithm are:

- $\mathbf{x}_0$: initial configuration;
- $\mathbf{X}$: image;
- θ: energy model parameters;
- T_0: initial temperature;
- n_s: number of samples;
- α_T: temperature decay rate (see annealing in Section 3.3.4).

Input: $\mathbf{x}_0, \mathbf{X}, \theta, T_0, n_s, \alpha_T$
for $t = 0, \ldots, n_s - 1$ **do**
  Pick *diffusion* with probability 0.8, else *jump*
  Pick mic-set s with probability $p(s)$
  Keep each c in s with probability $p(c|s)$ to make $\tilde{s}$
  $\mathbf{x}_{t+1} \leftarrow \mathbf{x}_t$ /* *unvisited cells will keep previous state* */
  for all $c \in \tilde{s}$ **do**
    if *diffusion* **then**
      $dw \sim \mathcal{N}(0, \gamma)$
      $\Delta C_c(\mathbf{x}_t) = -\gamma \nabla_{C_c(\mathbf{x}_t)} U(\mathbf{x}_t, \mathbf{X}, \theta) + dw\sqrt{2T_t}$
      $C_c(\mathbf{x}_{t+1}) \leftarrow C_c(\mathbf{x}_t) + \Delta C_c(\mathbf{x}_t)$
    else
      $Q_c \leftarrow Q_{c,\mathrm{B}}$ with probability 0.5 else $Q_{c,\mathrm{D}}$ /* *pick birth or death* */
      $\mathbf{x}' \sim Q_c(\mathbf{x} \rightarrow \cdot)$
      $r \leftarrow \frac{Q_c(\mathbf{x}' \rightarrow \mathbf{x})}{Q_c(\mathbf{x} \rightarrow \mathbf{x}')} \exp\left(-\frac{U(\mathbf{x}', \mathbf{X}, \theta) - U(\mathbf{x}, \mathbf{X}, \theta)}{T_t}\right)$ /* *compute Green ratio* */
      $C_c(\mathbf{x}_{t+1}) \leftarrow C_c(\mathbf{x}')$ with probability $\min(1, r)$
    end if
  end for
  $T_{t+1} \leftarrow \alpha_T T_t$
end for
Output: $\mathbf{x}_{n_s}$

Algorithm 5.1: Sampling method $\mathrm{Sample}(\mathbf{x}_0, \mathbf{X}, \theta, T_0, n_s, \alpha_T)$.

Alternatively, in Algorithm 5.1 we can run the diffusion kernel interrupted by jumps (birth or death) after a wait time ω, with ω following a Poisson distribution:

$$\omega \sim p(\omega) = \frac{\tau^\omega}{\omega!} \exp(-\tau), \tag{5.19}$$

where $\tau = \mathbb{E}(\omega)$ represents the expected waiting time thus controls the frequency of the jumps (Descombes, 2013).

In this section we first introduced using the potential map provided by the CNN to first build a local transform kernel based on the position potential map $\mathbf{V}_{pos}$. Then we improved on (Lacoste et al., 2005), by building a birth map from truncated model given by the CNN output instead of precomputing contrast for numerous objects in the image. In the continuation of data driven kernels, we implement the diffusion mechanism, by proposing to leverage the modern automatic differentiation engines. Finally, we adapted the parallel approach from (Verdié & Lafarge, 2012), proposing a cell selection scheme that simulates the Point Process in parallel given the density derived from the potential map.

5.2 Parameters estimation

Our model introduced throughout Chapter 4 contains several parameters that require to be set before sampling configurations. The most important of which being the relative weights on the

energies introduced in (4.3). We also have a few parameters in the various priors of the model. We denote by θ the set of parameters and by $U(\mathbf{y}, \mathbf{X}, \theta)$ the resulting energy model. The ideal parameters θ^* should be such that the ground truth configuration $\mathbf{y}^{GT}$ corresponds to the minimum of the energy U for an image $\mathbf{X}$:

$$\mathbf{y}^{GT} = \arg\min_{\mathbf{y} \in \mathcal{Y}} U\left(\mathbf{y}, \mathbf{X}, \theta^*\right). \tag{5.20}$$

We first present some previous approaches for Point Process parameter estimation, for instance using linear constraints. Then, we propose a method to learn the weights of the energies $\theta = \{w_0\} \cup \{w_e,\, e \in \xi\}$ based on margin maximization, solving the over-constraint issue of the previous approach from (Q. Yu & Medioni, 2009; Craciun et al., 2015). This was published in (Mabon et al., 2021). Eventually we adapt learning methods from energy based models, using contrastive divergence (Hinton, 2002; Teh, Welling, Osindero, & Hinton, 2003) with a replay buffer (Du & Mordatch, 2019) to estimate the Point Process parameters with gradient descent. This approach was first published in (Mabon et al., 2022a) and refined in (Mabon et al., 2023a) and (Mabon et al., 2023b).

5.2.1 Learning energy weights with local perturbations

5.2.1.1 Linear Programming

In their work (Crăciun, 2015) and (Q. Yu & Medioni, 2009) use random perturbations on the Ground Truth $\mathbf{y}^{GT}$ to generate non-valid configurations and build a set of linear constraints to estimate the weights w_e. As a matter of fact, re-formulating (5.20) locally yields:

$$\begin{aligned} U\left(\mathbf{y}^{GT}, \mathbf{X}, \theta^*\right) &< U\left(\mathbf{y}^{-}, \mathbf{X}, \theta^*\right) \\ \mathbf{y}^{-} &\sim Q^{-}\left(\mathbf{y}^{GT}, \cdot\right), \end{aligned} \tag{5.21}$$

where $\mathbf{y}^-$ is a *negative* configuration, generated from perturbation kernel Q^-. The kernel Q^- is build such that the perturbation on $\mathbf{y}^{GT}$ results in a perceptually *worse* configuration. For instance (Crăciun, 2015) use random translation, rotations and scaling (with a great amplitude) along with random addition and removal of points.

Once a certain number of linear constraints are built, (Crăciun, 2015) and (Q. Yu & Medioni, 2009) use Linear Programming to solve for the weights w_e. However, this approach has several limitations:

First, the number of constraints are to be limited, otherwise this leads to an over-constrained unsolvable problem. The latter couples badly with the imprecise nature of the ground truth; with noisy ground truth, some linear constraints can become contradictory, thus making the problem unsolvable. With noisy input data, one would resort to use more data to compensate the variance of the input; the latter Linear Programming approach does not offer this possibility. We propose to address this issue in the following Section 5.2.1.2.

Finally, how the negative sample $\mathbf{y}^-$ is built from to the ground truth depends on the design of this kernel Q^-. In practice, we find we need to iterate between crafting the perturbation Q^- and solving for w_e. For instance, given a set of negative samples $\mathbf{y}^-$, we obtain weights w_e and find out that, at inference, it exhibits some bad characteristics (e.g. too much overlap). We then proceeded to build a new set of negative configurations $\mathbf{y}^-$ via a new Q^-, with more examples to

the bad characteristic (e.g. more overlaps). We obtain a new set of weights w_e that would hopefully produce less of the negative characteristic. Section 5.2.2 proposes to address the Q^- kernel design issues.

5.2.1.2 Margin maximization

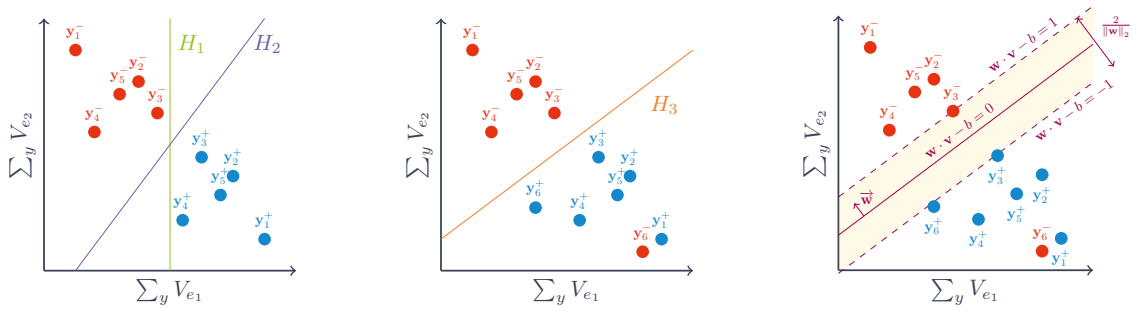
With this first approach proposed in (Mabon et al., 2021), we try to find the best linear separation between ground truth and *negative* configurations/samples.

First we consider a set of *negative* samples $\mathbf{y}^-$ generated — as with the previous method — from a kernel Q^-; $\mathbf{y}^- \sim Q^-(\mathbf{y}^{GT}, \cdot)$. We also introduce *positive* samples $\mathbf{y}^+$ that are valid configurations, obtained by small perturbations of the ground truth with a perturbation kernel Q^+: $\mathbf{y}^+ \sim Q^+(\mathbf{y}^{GT}, \cdot)$. On one hand this models the uncertainty on the ground truth data due to its noisy nature. This also help to balance the separation problem introduced later on and artificially augments the dataset. Lastly, introducing energy term $V_0(y) = 1$ and thus $\xi' = \{0\} \cup \xi$ we can rewrite (4.3) into:

$$U(\mathbf{y}, \mathbf{X}, \theta) = \sum_{y \in \mathbf{y}} \sum_{e \in \xi'} w_e V_e(y, \mathbf{X}, \mathcal{N}^{\mathbf{y}}_{\{y\}}) = \sum_{e \in \xi'} w_e \sum_{y \in \mathbf{y}} V_e(y, \dots) = \mathbf{w} \cdot \mathbf{v}$$
$$\mathbf{w} = \begin{bmatrix} w_0 \\ w_{pos} \\ \vdots \end{bmatrix}, \quad \mathbf{v} = \begin{bmatrix} \sum_{y \in \mathbf{y}} V_0(y, \dots) \\ \sum_{y \in \mathbf{y}} V_{pos}(y, \dots) \\ \vdots \end{bmatrix}. \tag{5.22}$$

In a sense, we describe any configuration $\mathbf{y}$ as a point with coordinates $\mathbf{v}$ in the *energy terms space*; configuration energy U is a linear function from this space to $\mathbb{R}$, defined as the dot product of a weight vector $\mathbf{w}$ and the coordinates vector $\mathbf{v}$ of $\mathbf{y}$ in this space.

Given a set of negative and positive samples, we look for a hyperplane defined by $\mathbf{w} \cdot \mathbf{v} - b = 0$, that best separates negative and positive samples. As shown in Figure 5.3, there might be more than one hyperplane (H_1 and H_2 in (a)) or no hyperplane that can separate all the data (H_3 in (b)).



(a) Linearly separable (b) Non-separable (c) Hyperplane and margin

Figure 5.3: Separation of energy configurations via hyperplanes in the energy terms space. For this illustration we only consider two energies V_{e_1} and V_{e_2}. (a) shows multiple possible hyperplanes H_1, H_2 that perform separation. In (b) there is no linear separation possible. (c) shows the hyperplane and associated margin; vector $\vec{\mathbf{w}}$ shows in which direction the energy increases.
Figures are adapted from (Cortes & Vapnik, 1995) and (Larhmam, 2018).

Support Vector Machine. With the Support Vector Machine (SVM) (Cortes & Vapnik, 1995) propose to find the hyperplane that maximizes the gap in between the two categories within a classification problem. We consider a set of samples $\mathbf{y}_k$, $k = 1, \ldots, n$ with associated vectors $\mathbf{v}_k$, with labels $l_k = -1$ for positive samples and $l_k = 1$ for negative samples[3].

For linearly separable data, the *Hard-margin* criterion minimize $\|\mathbf{w}\|_2^2$ (i.e. maximizes the gap $2/\|\mathbf{w}\|_2$) with constraint $l_k(\mathbf{w} \cdot \mathbf{v}_k - b) \geq 1$ for all $k = 1, \ldots, n$; i.e. all configurations lie in the correct side of the hyperplane (see Figure 5.3 (c)).

However, our data might not be separable: (Cortes & Vapnik, 1995) propose to use the *hinge loss* and minimize the following instead:

$$\gamma \|\mathbf{w}\|_2^2 + \frac{1}{n} \sum_{k=1}^{n} \max\left(0, 1 - l_k(\mathbf{w} \cdot \mathbf{v}_k - b)\right), \tag{5.23}$$

with $\gamma > 0$ a scalar setting the balance between increasing the margin and having points in the correct side.

Remark 5.2.1 – SVM are often used with nonlinear kernels, allowing for nonlinear separation functions. In our case the energy model is linear; we discuss the addition of nonlinearities in the energy mixing function in Section 4.4.2.2.

Application and limitations. Minimizing this loss on the training data gives us the weights $\mathbf{w}$ and b. In our case, parameter b in (5.23) is superfluous as the energy needs to be known up a to a constant.

This method allows learning weights w_e even from non-separable set of positive and negative samples. Moreover, we can consider many samples without risking to over-constraint the optimization. However, this method is still reliant on the two perturbation kernels Q^- and Q^+ that respectively generate the negative and positive samples. Finally, this does not allow to tune the energy term parameters such as $t_{pos}, \mu_{area}, t_{ovrlp}, \ldots$. Those have to be estimated separately or set manually.

5.2.2 Maximum likelihood learning

In this section we approach our parameter estimation problem from the angle of Energy Based Models (EBM).

Definition 5.2.1. Energy Based Model (EBM) is a form of generative model; it captures the dependencies between variables by associating a scalar energy to each configuration of variables. Importantly it allows for an implicit sample generation procedure, where sample $\mathbf{x}$ is found from $\mathbf{x} \sim \exp(-U(\mathbf{x}))$. *Learning* the EBM consists in finding the energy function that associates low energies to correct values of the variables. (LeCun et al., 2006; Du & Mordatch, 2019).

[3] We swap positive and negative labels due to the reverse ordering on energies; better configurations get lower energy. That way the equations fall back to the canonical formulations for SVM.

Formally, a Gibbs Point Process model is an Energy Based Model. The works of (LeCun et al., 2006) propose to learn EBMs by maximizing the likelihood for the data $\mathcal{D}$:

$$p\left(\mathbf{y}_1^{GT}, \ldots, \mathbf{y}_{n_\mathcal{D}}^{GT} | \mathbf{X}_1, \ldots, \mathbf{X}_{n_\mathcal{D}}, \theta\right) = \prod_{k=1}^{n_\mathcal{D}} p\left(\mathbf{y}_k^{GT} | \mathbf{X}_k, \theta\right). \tag{5.24}$$

The latter can be derived into the Negative Log-Likelihood (NLL) loss:

$$\mathcal{L}_{\text{NLL}}\left(\theta, \mathcal{D}\right) = \frac{1}{n_\mathcal{D}} \sum_{k=1}^{n_\mathcal{D}} \left(U\left(\mathbf{y}_k^{GT}, \mathbf{X}_k, \theta\right) + \beta \log \int_{\mathbf{y} \in \mathcal{Y}} \exp\left(-\beta^{-1} U\left(\mathbf{y}, \mathbf{X}_k, \theta\right)\right)\right), \tag{5.25}$$

with β a temperature parameter (from the Gibbs distribution), that has no effect on the position of the minimum. While the left part $U(\mathbf{y}_k^{GT}, \mathbf{X}_k, \theta)$ pulls down the energies of ground truth configurations, the right part $U(\mathbf{y}, \mathbf{X}_k, \theta)$ pulls up the energies of all configurations in $\mathcal{Y}$.

When minimizing the loss via gradient descent, the gradient at step n for parameters θ_n and data sample $k = 1, \ldots, n_\mathcal{D}$, is given as:

$$\frac{\partial \mathcal{L}_{\text{NLL}}\left(\theta_n, \mathbf{y}_k^{GT}, \mathbf{X}_k\right)}{\partial \theta_n} = \frac{\partial U\left(\mathbf{y}_k^{GT}, \mathbf{X}_k, \theta_n\right)}{\partial \theta_n} - \underbrace{\int_{\mathbf{y} \in \mathcal{Y}} \frac{\partial U\left(\mathbf{y}, \mathbf{X}_k, \theta_n\right)}{\partial \theta_n} p\left(\mathbf{y} | \mathbf{X}_k, \theta_n\right)}_{I_{k,n}}, \tag{5.26}$$

where $p(\mathbf{y}|\mathbf{X}_i, \theta_n)$ is given from the Gibbs distribution:

$$p(\mathbf{y}|\mathbf{X}_i, \theta_n) = \frac{\exp(-\beta^{-1} U(\mathbf{y}, \mathbf{X}_k, \theta_n))}{\int_{\mathbf{y}' \in \mathcal{Y}} \exp(-\beta^{-1} U(\mathbf{y}', \mathbf{X}_k, \theta_n))}. \tag{5.27}$$

While the integral $I_{k,n}$ remains intractable, it can be approximated through Monte Carlo sampling, where $\widetilde{\mathbf{y}}_1, \ldots, \widetilde{\mathbf{y}}_{n_s}$ are n_s samples drawn from the law defined by $p(\cdot|\mathbf{X}_k, \theta_n)$, yielding:

$$I_{k,n} \simeq \frac{1}{n_s} \sum_{s=1}^{n_s} \frac{\partial U\left(\widetilde{\mathbf{y}}_s, \mathbf{X}_k, \theta_n\right)}{\partial \theta_n}. \tag{5.28}$$

As the law defined by $p(\cdot|\mathbf{X}_k, \theta_n)$ is simply the Gibbs Point Process with energy $U(\mathbf{y}, \mathbf{X}_k, \theta_n)$ at temperature β, we can use the Point Process sampling procedure we introduced previously.

5.2.2.1 Contrastive divergence

Hinton et al. in (Hinton, 2002; Teh et al., 2003) propose to use a single sample in his *contrastive divergence* method. This method also uses few simulation steps for the Monte Carlo Markov chain (MCMC) to generate $\mathbf{y}^-$, starting from the desired answer $\mathbf{y}^{GT}$.

The general idea is to generate *contrastive samples* $\mathbf{y}^-$ that follow the density derived from $U(\cdot, \mathbf{X}, \theta_n)$ at step n of the optimization. Then we proceed to update θ_n to θ_{n+1}, by gradient descent, with the goal to minimize the energy of the *valid sample* $\mathbf{y}^{GT}$, while maximizing the energy of the *contrastive sample* $\mathbf{y}^-$ (see Figure 5.4). Alternatively we can augment the data and use *positive samples* $\mathbf{y}^+ = \mathbf{y}^{GT} + \mathcal{N}(0, \sigma^+)$ to replace $\mathbf{y}^{GT}$ as in (Du & Mordatch, 2019). The loss to minimize is then:

$$\begin{aligned} \mathcal{L}\left(\theta_n, \mathbf{y}^+, \mathbf{y}^-, \mathbf{X}\right) &= U(\mathbf{y}^+, \mathbf{X}, \theta_n) - U(\mathbf{y}^-, \mathbf{X}, \theta_n) + \gamma R_V \\ R_V &= \sum_{\mathbf{y} \in \{\mathbf{y}^+, \mathbf{y}^-\}} \frac{1}{|Y|} \sum_{y \in \mathbf{y}} \left| V\left(\mathbf{y}, \mathbf{X}, \mathcal{N}_{\{y\}}^{\mathbf{y}}, \theta_n\right) \right|, \end{aligned} \tag{5.29}$$

with $\gamma > 0$ the weight of regularization term R. We introduce the regularization term to avoid an explosion of the per-point energy $V(y, \mathbf{X}, \mathcal{N}^{\mathbf{y}}_{\{y\}}, \theta_n)$. To ensure for a sparse weighting of energies (i.e. minimize the number of non-zero weights w_e) we can introduce a new regularization term as an L^1 norm on the vectors of weights (Goodfellow et al., 2016):

$$R_1 = \sum_{e \in \xi} |w_e| \tag{5.30}$$

The broad strokes of the estimation procedure for parameters θ are as follows:

1. Pick a pair $(\mathbf{y}^{GT}, \mathbf{X})$ from data $\mathcal{D}$.
2. Generate positive sample $\mathbf{y}^+ = \mathbf{y}^{GT} + \mathcal{N}(0, \sigma^+)$.
3. Generate negative sample $\mathbf{y}^- \sim \exp(-\beta^{-1} U(\mathbf{y}^-, \mathbf{X}, \theta_n))$.
4. Compute the loss $\mathcal{L}\left(\theta_n, \mathbf{y}^+, \mathbf{y}^-, \mathbf{X}\right)$ (see (5.29)).
5. Update θ_n to θ_{n+1} according to the gradient $\nabla\mathcal{L}$, with the Stochastic Gradient Descent (SGD) method (Bottou, 2012).
6. Loop back to step 1 until convergence.

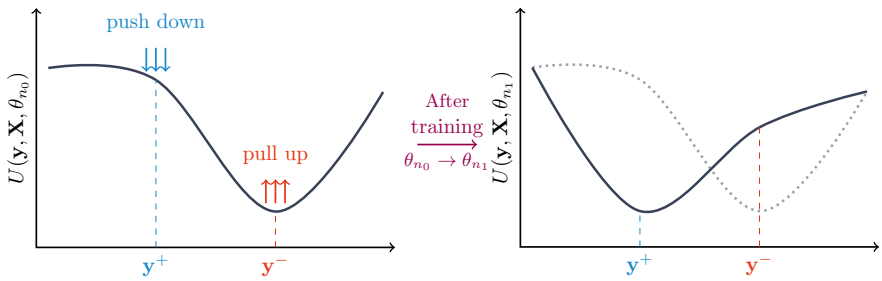


Figure 5.4: Effect of training the energy with contrastive samples generated from the current θ_n.
Figure adapted from (LeCun et al., 2006).

While the local perturbation method (Section 5.2.1) only learns from constraints with states in the neighborhood of the Ground Truth $\mathbf{y}^{GT}$ (or positive samples $\mathbf{y}^+$), the contrastive divergence method allows learning by contrast against any state in $\mathcal{Y}$. It also focuses the contrastive samples on low energy states (instead of uniformly sampling in $\mathcal{Y}$ which would be inefficient).

5.2.2.2 Replay buffer

As sampling contrastive samples $\mathbf{y}^-$ can be time-consuming, (Du & Mordatch, 2019) propose an adaptation of Hinton's method (Hinton, 2002; Teh et al., 2003), making use of a replay buffer. The replay buffer saves Markov chain results at current optimization step, to use for initialization in the next steps, thus saving computing time. This allows reducing the long simulation time necessary to pass the burn-in period of the Markov Chain (Robert & Casella, 2004) and get samples $\mathbf{y}^-$. We

use a sample from the law derived from $U(\cdot, \mathbf{X}, \theta_{n-1})$ to initialize the chain that samples the law derived from $U(\cdot, \mathbf{X}, \theta_n)$.

The use of the replay buffer within one step n of the optimization loop goes as follows:

1. With probability $p_{\mathcal{B}}$ set configuration $\mathbf{x}_0$ as a value of the replay buffer $\mathcal{B}$ picked at random. With probability $1 - p_{\mathcal{B}}$ (or if the buffer is empty) set configuration $\mathbf{x}_0$ as a random configuration in $\mathcal{Y}$.
2. Run the Markov Chain $(\mathbf{x}_t)_{t=0,\dots,n_s}$ to simulate model of energy $U(\cdot, \mathbf{X}, \theta_n)$.
3. Use $\mathbf{y}^- = \mathbf{x}_{n_s}$ as a negative sample for the loss computation and update of θ_n to θ_{n+1}.
4. Update buffer $\mathcal{B} \leftarrow \mathcal{B} \cup \{\mathbf{y}^-\}$.

The buffer allows to virtually run longer chains *across* the epochs[4] as shown in Figure 5.5 (a). At each epoch we pick the result from a previous chain and continue it for a n_s steps before saving the result to the buffer. We find it is useful to limit the size of the buffer to $n_{\mathcal{B}}$, and update it on a *first in first out* fashion; with lower size $n_{\mathcal{B}}$, the Markov chain starts more often from *recent* contrastive samples, i.e. likely closer to an energy minimum. At the limit, with $n_{\mathcal{B}} = 1$, the contrastive sample generations pick up the last sample $\mathbf{y}^-$, and updates it with n_s MCMC steps (see Figure 5.5 (b)). This yields an expect cross-epoch chain length of $\frac{n_s p_{\mathcal{B}}}{1-p_{\mathcal{B}}}$, allowing to run longer chains with fewer steps. This corresponds to the *persistent contrastive divergence* proposed by (Tieleman, 2008). We discuss the effects of the choice of $p_{\mathcal{B}}$ and n_s on the *cross-epoch* chain in Appendix F.

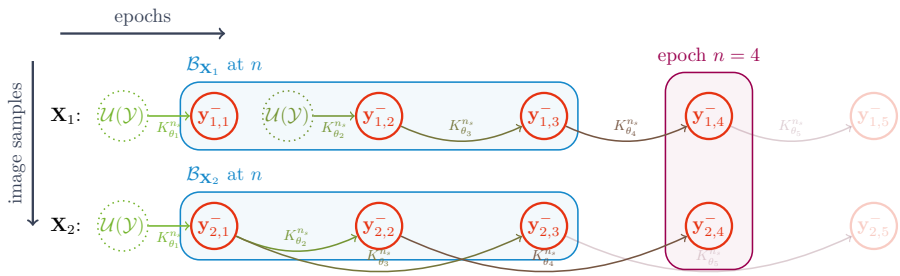
Remark 5.2.2 – Contrary to (Du & Mordatch, 2019) — where the authors perform image generation — our generated configurations $\mathbf{y}^-$ depend on the current image $\mathbf{X}_k$. Thus, we define the buffer $\mathcal{B}$ as a set of sub-buffers $\mathcal{B}_{\mathbf{X}_k}$, each associated to an image $\mathbf{X}_k$. A configuration $\mathbf{y}^-$ generated from image $\mathbf{X}_k$ get added to sub-buffer $\mathcal{B}_{\mathbf{X}_k}$.

5.2.2.3 Effect of temperature on samples

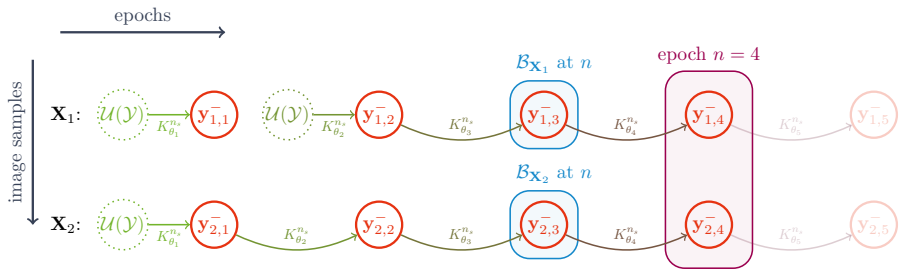
In the contrastive divergence method (Section 5.2.2.1), we introduce a temperature parameter β to generate the negative samples $\mathbf{y}^-$. This temperature parameter is the same as the one introduced for simulated annealing (Section 3.3.4) and has the same effects on the generated samples. At high temperature the samples get closer to a uniform distribution in $\mathcal{Y}$, and as temperature lowers samples get closer to local or global minima. Within the contrastive divergence optimization scheme:

- Picking a high β allows to increase the energy of configurations far away from $\mathbf{y}^{GT}$. Samples are not likely to be drawn close to $\mathbf{y}^{GT}$, thus the learned model will not discriminate efficiently around $\mathbf{y}^{GT}$.
- At low β sample are drawn consistently in local minima around $\mathbf{y}^{GT}$, the ability to discriminate against configurations that differ greatly from $\mathbf{y}^{GT}$ will be diminished.

[4] An epoch designates one iteration of the estimation procedure over the whole data $\mathcal{D}$



(a) Unlimited buffer size



(b) Buffer size $n_{\mathcal{B}} = 1$

Figure 5.5: Buffer size effect on contrastive samples. Buffers are represented in blue. In this example the Markov chain starts from $\mathbf{y}^-_{\cdot,0} \sim \mathcal{U}(\mathcal{Y})$, thus the unlimited buffer is only of size 3 at epoch $n = 4$. Configuration $\mathbf{y}^-_{k,n}$ corresponds to contrastive sample for image $\mathbf{X}_k$ at epoch n, and $K^{n_s}_{\theta_n}$ is the application of the transition kernel of the RJMCMC n_s times with parameters θ_n.

We propose using a set of β, $\{\beta_1, \beta_2, \beta_3\}$ corresponding to low, medium and high temperatures, allowing to mix samples explore $\mathcal{Y}$ while drawing some samples near local minima or $U(y, \mathbf{X}, \theta_n)$.

Remark 5.2.3 – As for the previous remark, we further separate the buffer to associate each sub-buffer to a temperature β. Now the buffer corresponding to image $\mathbf{X}$ and temperature β is $\mathcal{B}_{\mathbf{X},\beta}$.

Remark 5.2.4 – When estimating weights w_0, w_e we notice that scaling the weights by a factor ς, is equivalent to scaling the temperature by ς^{-1}:

$$\frac{U(\mathbf{y}, \mathbf{X}, \{w_0\} \cup \{w_e,\, e \in \xi\})}{T\varsigma^{-1}} = \frac{U(\mathbf{y}, \mathbf{X}, \{\varsigma w_0\} \cup \{\varsigma w_e,\, e \in \xi\})}{T}. \tag{5.31}$$

Thus picking, a temperature might have different effect depending on the weights. In practice, we find that scaling the energy model to unit variance solves this consistency issue; we compute the energy model variance given its parameters θ:

$$\mathrm{Var}(U_{T=\infty}) = \langle U^2_{T=\infty} \rangle - \langle U_{T=\infty} \rangle^2, \tag{5.32}$$

with $\langle U_{T=\infty} \rangle$ the mean of energy U over multiple samples drawn at infinite temperature (i.e. drawn from $\mathcal{U}(\mathcal{Y})$).

We then simulate the energy model $\bar{U} = U/\sqrt{\mathrm{Var}(U_{T=\infty})}$ (so that $\mathrm{Var}(\bar{U}_{T=\infty}) = 1$). This is equivalent to specifying the temperature for a unit variance energy model and scaling it to the current energy model it is being applied on. The simple procedure to compute the scaling factor is presented in Algorithm 5.2, where we pick 10000 as an arbitrary large number of samples to compute the variance on.

Input: $\theta, \mathbf{X}$
1: $\mathit{Values} \leftarrow \emptyset$
2: **for all** $k = 1, \ldots, 10000$ **do**
3: $\quad \mathbf{x} \sim \mathcal{U}(\mathcal{Y})$ /* sample random configuration */
4: $\quad \mathit{Values} \leftarrow \mathit{Values} \cup \{U(\mathbf{x}, \mathbf{X}, \theta)\}$
5: **end for**
6: $\varsigma \leftarrow \sqrt{\mathrm{Var}(\mathit{Values})}$
Output: ς

Algorithm 5.2: Computation of the energy scaling factor, ComputeScaling.

5.2.2.4 Algorithm

Resulting from the above, we have the following algorithm for inferring parameters $\widehat{\theta}$:

1: $\mathcal{B} \leftarrow \emptyset$, $n \leftarrow 0$
2: **while** not converged **do**
3: **for all** $(\mathbf{y}^{GT}, \mathbf{X})$ in $\mathcal{D}$ **do**
4: $\beta \sim \mathcal{U}(\{\beta_1, \beta_2, \beta_3\})$ ** select negative sample temperature */*
5: $\mathbf{x}_0 \sim \mathcal{U}(\mathcal{B}_{\mathbf{X},\beta})$ with probability $p_{\mathcal{B}}$, else $\mathcal{U}(\mathcal{Y})$ ** retrieve relevant buffer */*
6: $\varsigma \leftarrow \text{ComputeScaling}(\theta_n, \mathbf{X})$ ** compute energy scale, see Alg. 5.2 */*
7: $\mathbf{y}^- \leftarrow \text{Sample}\,(\mathbf{x}_0, \mathbf{X}, \theta_n, \varsigma\beta, K, 1)$ ** see* $\text{Sample}(\mathbf{x}_0, \mathbf{X}, \theta, T_0, n_s, \alpha_T)$ *in Alg. 5.1 */*
8: $\mathbf{y}^+ \leftarrow \mathbf{y}^{GT} + \mathcal{N}(0, \sigma^+)$
9: $\Delta\theta_n \leftarrow \nabla_{\theta_n}\mathcal{L}\,(\theta_n, \mathbf{y}^+, \mathbf{y}^-, \mathbf{X})$
10: Update θ_{n+1} with $\Delta\theta_n$ using SGD
11: $\mathcal{B}_{\mathbf{X},\beta} \leftarrow \{\mathbf{y}^-\} \cup \mathcal{B}_{\mathbf{X},\beta}$ ** update the buffer */*
12: $n \leftarrow n + 1$
13: **end for**
14: **end while**

Algorithm 5.3: Contrastive divergence for parameters estimation.

The sampling method at line 7 is described in Algorithm 5.1. In practice one iteration of the loop on data at line 3 is done on a mini batch of ground truth and image pair instead of a single pair; it allows smoothing the gradient between iterations by averaging on data samples.

In this section we contribute in proposing to use contrastive divergence (adapted from energy based models) to estimate the parameters of the Point Process model. The use of replay buffer (or persistent contrastive divergence) allows cutting on sampling time when sampling negative configurations. While previous approaches using linear constraints and linear programming would often be limited to learning energy weights, and risk over-constraint with too noisy or too many data points, our method does not suffer from over-constraint, and allows solving for all parameters of the model at once.

5.2.2.5 Discussions

Relating contrastive divergence to manual methods. In practice, the contrastive divergence method relates to the empirical trial and error procedure used in previous works to set $\widehat{\theta}$. Indeed, oftentimes the manual procedure to find weights w_e goes as follows:

1. the user sets some approximate values to weights $\tilde{w}_e$ within $\widetilde{\theta}$,
2. the user simulates configurations $\tilde{\mathbf{y}} \sim U(\cdot|\mathbf{X}, \widetilde{\theta})$,
3. if the inference is of good quality, the procedure ends. Otherwise, the user updates values for weights $\tilde{w}_e$, picked to counteract the observed bad configurations; (e.g., *the configuration has too much overlap* $\rightarrow$ *increase the weight of the overlap energy*). Then go back to step 2.

In a broad sense, the repeat of steps 2 and 3 maps to the alternated sampling (Alg. 5.3 line 7) and loss minimization (Alg. 5.3 lines 9 to 10). As step 3 in the manual procedure, attempts to maximize the energy of the (bad) sampled configuration, minimizing the loss in Algorithm 5.3 tends to increase $U(\mathbf{y}^-, \mathbf{X}, \theta_n)$ and decrease $U(\mathbf{y}^+, \mathbf{X}, \theta_n)$.

Learning more than just weights. For previous parameter inference method, we could only infer the weights of the linear combination of energy terms. Here, the only requirement to be able to learn a parameter is to be able to back propagate the energy model gradient up to that parameter. This allows to learn not only weights but also energy term parameters so that:

$$\theta = \{w_0\} \cup \{w_e,\, e \in \xi\} \cup \{t_{pos}, \mu_{area}, \sigma_{area}, \mu_{ratio}, \mu_{ratio}, t_{ovrlp}, t_{repls}, t_{attrc}\}.$$

We detail in Appendix I the implementation allowing such back propagation of the gradient.

It would be possible to infer the parameters of the whole data energy model (i.e., the backbone CNN), however this proves quite computationally expensive; training the CNN requires many epochs as parameters are numerous ($> 10^6$ parameters). Meanwhile, the contrastive divergence method (Algorithm 5.3) requires simulating a few Markov Chain steps in between epochs to generate contrastive samples, which increases the computational complexity of each epoch.

5.3 Papangelou intensity as a score

5.3.1 Computing the detection score

Classical CNN models for object detection yield a confidence score $s(y) \in \mathbb{R}$ for each proposed object y in the image. This confidence score is often interpreted, for each detection, as proportional to the probability of proposed element y to be a true object (true positive), $s(y) \propto p(y|\mathbf{X})$. Applying a score threshold t_s gives a set of detections, for which metrics such as precision and recall can be computed by matching the detections with the ground truth. As the threshold has to be adapted according to the need of the application of the model; i.e. some applications may require few false positive (high precision) while other need less missed detections (high recall). To assess the performance independently of the threshold selection, the Average Precision (AP) metrics sums up the performance as the area under the precision-recall curve.

With the PP approach, the probability of one proposed point being an object of interest depends on the rest of the inferred configuration $\hat{\mathbf{y}}$, thus the scoring function should too $s(y|\hat{\mathbf{y}} \setminus \{y\}) \propto p(y|\hat{\mathbf{y}} \setminus \{y\}, \mathbf{X})$. From Definition 3.1.7 we have that the Papangelou intensity is proportional to the probability of finding a point $y \in \mathbf{y}$ in a small neighborhood dy knowing the rest of the configuration $\mathbf{y} \setminus \{y\}$. We propose to use the Papangelou intensity as a score :

$$\begin{aligned} \lambda(y; \mathbf{y} \setminus \{y\}) &= \frac{h(\mathbf{y})}{h(\mathbf{y} \setminus \{y\})} \\ &= \exp\left(U(\mathbf{y} \setminus \{y\}, \mathbf{X}, \widehat{\theta}) - U(\mathbf{y}, \mathbf{X}, \widehat{\theta})\right). \end{aligned} \tag{5.33}$$

Pruning sequence. However, the dependency of the score on the current configuration yields a complication while computing the Average Precision: when applying a threshold t_s to prune the configuration $\mathbf{y}$ into $\mathbf{y}' \subset \mathbf{y}$, for any $y \in \mathbf{y}'$, the score $s(y|\mathbf{y} \setminus \{y\})$ may differ from $s(y|\mathbf{y}' \setminus \{y\})$. With a score of the form $s(y)$, that only depends on y and the image $\mathbf{X}$ — such as those from classical CNN models — the score from one object after pruning is unchanged.

In the PP case, we compute the scores by sequentially removing the lowest scoring point until none is left; i.e., we build a sequence of configurations $\mathbf{y}_1 \supset \mathbf{y}_2 \dots \mathbf{y}_{n(\hat{\mathbf{y}})-1} \supset \mathbf{y}_{n(\hat{\mathbf{y}})} \supset \emptyset$, with

$\mathbf{y}_1 = \hat{\mathbf{y}}$:

$$\mathbf{y}_{n+1} = \mathbf{y}_n \setminus \{y_n\},\ y_n = \underset{y \in \mathbf{y}_n}{\arg\min} \lambda(y; \mathbf{y}_n \setminus \{y\}), \quad n = 1, \dots, n(\hat{\mathbf{y}}) \tag{5.34}$$

$$s(y_n | \mathbf{y}_n \setminus \{y_n\}) = s(y_n | \mathbf{y}_{n+1}) = \lambda(y; \mathbf{y}_{n+1}) \tag{5.35}$$

Equation (5.34) provides a pruning order $y_1, \dots, y_{|\hat{\mathbf{y}}|}$ of points in $\hat{\mathbf{y}}$. This ordering allows to plot the precision and recall curve. Indeed, to trace a precision recall-curve, one only requires the sequence of $(\text{Recall}(t_s), \text{Precision}(t_s))$ pairs, which are obtained by sequentially pruning the lowest scoring points. Equation (5.35) provides a score to each point y_n.

5.3.2 Contrastive divergence loss and Papangelou intensity

On one hand we minimize the loss in (5.29) derived from the likelihood maximization, on the other we evaluate the performance of our detections with the scoring method in (5.35) sourced from the Papangelou intensity. In this part we show that while the two are derived differently, minimization of the loss function leads to good properties on the score function. Here we consider the simplified loss, as γ in (5.29) is small:

$$\begin{aligned} \mathcal{L}(\theta, \mathbf{y}^+, \mathbf{y}^-, \mathbf{X}) &= U(\mathbf{y}^+, \mathbf{X}, \theta) - U(\mathbf{y}^-, \mathbf{X}, \theta) \\ &= \Delta U(\mathbf{y}^- \to \mathbf{y}^+), \end{aligned} \tag{5.36}$$

denoting $\Delta U(\mathbf{y} \to \mathbf{x}) = U(\mathbf{x}) - U(\mathbf{y})$.

Theorem 3.1.2 gives us the following expression of the Papangelou intensity:

$$\begin{aligned} \lambda(u; \mathbf{y}) &= \frac{h(\mathbf{y} \cup \{u\})}{h(\mathbf{y})} \\ &= \exp(U(\mathbf{y}) - U(\mathbf{y} \cup \{u\})) \\ &= \exp(\Delta U(\mathbf{y} \cup \{u\} \to \mathbf{y})) \end{aligned} \tag{5.37}$$

Case 1: single point addition. Considering a basic pair of positive contrastive samples, in which we add a non-valid point u to $\mathbf{y}^+$:

$$\mathbf{y}^- = \mathbf{y}^+ \cup \{u\},\ u \in \mathcal{S} \times \mathcal{M},\ u \notin \mathbf{y}^+.$$

Now we have:

$$\begin{aligned} \mathcal{L}(\theta, \mathbf{y}^+, \mathbf{y}^-, \mathbf{X}) &= \Delta U(\mathbf{y}^+ \cup \{u\} \to \mathbf{y}^+) \\ &= \log(\lambda(u; \mathbf{y}^+)). \end{aligned}$$

We get the expected behavior: minimizing the loss $\mathcal{L}$ leads to minimizing the score of point u.

Case 2: single point removal. Now considering the removal of a valid point y from $\mathbf{y}^+$ to form $\mathbf{y}^-$:

$$\mathbf{y}^- = \mathbf{y}^+ \setminus \{y\},\ u \in \mathbf{y}^+.$$

Then:

$$\begin{aligned} \mathcal{L}(\theta, \mathbf{y}^+, \mathbf{y}^-, \mathbf{X}) &= \Delta U(\mathbf{y}^+ \setminus \{y\} \to \mathbf{y}^+) \\ &= -\Delta U(\mathbf{y}^+ \to \mathbf{y}^+ \setminus \{y\}) \\ &= -\log(\lambda(y; \mathbf{y}^+ \setminus \{y\})). \end{aligned}$$

Here we get the expected result that minimizing the loss lead to increasing the score of y.

Case 3: Arbitrary sequence of moves. We now consider the generic case where $\mathbf{y}^-$ is generated from a sequence of moves from $\mathbf{y}^+$. Note that a local transformation (translation, rotation, scaling etc...) can be formulated as a death followed by a birth. We built a sequence $(\mathbf{y}_k)_{k=0,\dots,n}$ of n configurations as:

$$\forall k = 1, \dots, n,\ \mathbf{y}_k = \begin{cases} \mathbf{y}_{k-1} \setminus \{y_k\} & \text{if } y_k \in \mathbf{y}^+ \\ \mathbf{y}_{k-1} \cup \{y_k\} & \text{otherwise,} \end{cases}$$

with $\mathbf{y}_0 = \mathbf{y}^+$, $\mathbf{y}^- = \mathbf{y}_n$, and y_k elements of either $\mathcal{S} \times \mathcal{M}$ or $\mathbf{y}^+$[5]. Without loss of generality we can reorder the sequence to match the pruning order defined in (5.34). The energy change for one move is given as:

$$\Delta U(\mathbf{y}_{k-1} \to \mathbf{y}_k) = \begin{cases} \log(\lambda(y_k; \mathbf{y}_{k-1} \setminus \{y_k\})) & \text{if } y_k \in \mathbf{y}^+ \\ -\log(\lambda(y_k; \mathbf{y}_{k-1})) & \text{otherwise.} \end{cases}$$

As we have (by definition) $\Delta U(\mathbf{x} \to \mathbf{x}'') = \Delta U(\mathbf{x} \to \mathbf{x}') + \Delta U(\mathbf{x}' \to \mathbf{x}'')$, the loss is given as:

$$\begin{aligned} \mathcal{L}(\theta, \mathbf{y}^+, \mathbf{y}^-, \mathbf{X}) &= -\sum_{k=1}^{n} \Delta U(\mathbf{y}_{k-1} \to \mathbf{y}_k) \\ &= \sum_{y_k \notin \mathbf{y}^+} \log(\underbrace{\lambda(y_k; \mathbf{y}_{k-1})}_{\text{(a)}}) - \sum_{y_k \in \mathbf{y}^+} \log(\underbrace{\lambda(y_k; \mathbf{y}_{k-1} \setminus \{y_k\})}_{\text{(b)}}). \end{aligned}$$

With ordering of the $y_k, \mathbf{y}_k$ matching the pruning order in (5.34) each $\lambda(y_k; \dots)$ can be matched to their respective score:

- part (a) corresponds to non-valid points added to $\mathbf{y}^+$, their score is minimized as the loss is decreased;
- part (b) corresponds to valid points removed from $\mathbf{y}^+$, their score is increased as the loss is minimized.

Hereby we show that minimization of the loss at a configuration level leads to the expected results on object scores.

5.4 Conclusion

In this chapter we presented our methods to sample and estimate parameters of the Point Process model. We summarize our contributions as follows:

- Adapting data-based perturbation kernels to our pre-computed energy maps: both for local perturbation kernels (Section 5.1.1.1) and Birth kernels by sampling from a truncated energy function (Section 5.1.1.2).
- Adapting parallel computation to focus sampling on cells with estimated high density of object (Section 5.1.2).

[5] We shall add the condition that $(\mathbf{y}_k)_{k=1,\dots,n}$ contains no more than one instance of each element of $\mathbf{y}^+$ so that the set subtraction is properly defined.

- A simple procedure to estimate energy weight parameters, based on SVM, that avoids the over-constraint issue of previous approaches (Section 5.2.1.2).
- Another parameter estimation method based on contrastive divergence initially introduced for EBM. It allows learning internal energy parameters along with energy weights in a unified procedure. We adapt the replay buffer to the Point Process model, in order to shorten sampling times during parameter estimation (Section 5.2.2).
- Using the Papangelou intensity as scoring function for the inferred points, to compute benchmark metrics (Section 5.3).

CHAPTER 6

Experimental results

Dans ce chapitre, nous appliquons sur des données synthétiques et réelles les modèles et algorithmes proposés dans la thèse. Nous présentons d'abord l'application de la divergence contrastive sur un exemple simple utilisant des données synthétiques pour démontrer ses capacités. Nous présentons ensuite nos résultats avec des Processus Ponctuels non marqués sur un premier jeu de données : Cars Overhead With Context. Un modèle plus avancé est appliqué sur le jeu de données DOTA, qui montre d'excellents résultats par rapport à d'autres approches fondées sur les CNN ; les configurations inferrées sont plus régularisées et nos modèles sont plus résilients au bruit additif.

In this chapter we apply on both synthetic and real data the models and algorithms presented in the thesis. First we present the application of the contrastive divergence on a simple example using synthetic data to demonstrate its capabilities. We then show our first results with non-marked Point Processes on a first dataset: Cars Overhead With Context. A more advanced model is applied on DOTA, which shows great results compared to other CNN based approaches; inferred configurations are more regularized and our models are more resilient to additive noise.

6.1 Parameters estimation with Contrastive Divergence 119
6.1.1 Training example: Filtering out irrelevant terms 119
6.1.2 Stable perturbation kernel 121
6.2 Results on optical data 121
6.2.1 COWC with non-marked Point Process 121
6.2.1.1 Data 121
6.2.1.2 Point Process model 123
6.2.1.3 Parameter estimation and inference 123
6.2.1.4 Results 123
6.2.2 Point Process of rectangles on DOTA and ADS data 124
6.2.2.1 Data 124
6.2.2.2 Models 125
6.2.2.3 Parameter estimation and inference 126
6.2.2.4 Results 126
6.2.2.5 Computational complexity 130
6.3 Conclusion 131

6.1 Parameters estimation with Contrastive Divergence

6.1.1 Training example: Filtering out irrelevant terms

In the following part we demonstrate a use of the contrastive divergence method for parameter estimation in a simple case, and how it is able to properly weight the energy terms within the energy model; i.e. the estimated weights reflect the relative importance of energy terms within the generative model. We proceed as follows:

1. Define an energy model.
2. Sample some configurations given a set of energy weight parameters.
3. Estimate the energy weight parameters from the generated data in order to simulate similar data.

We consider a Point Process of circles, where each point $y \in \mathbf{y}$ is such that $y \in \mathcal{S} \times \mathcal{M} = [0, H] \times [0, W] \times [r_{min}, r_{max}]$, with $y = (y_i, y_j, y_r)$. The energy model is composed of the following energy terms:

$$V_{pos}(y, \mathbf{X}) = \mathbf{X}[y_i, y_j] \tag{6.1}$$

$$V_{ovrlp}(y, \mathcal{N}^{\mathbf{y}}_{\{y\}}) = \max_{u \in \mathcal{N}^{\mathbf{y}}_{\{y\}}} \left\{ \max \left(0, \frac{d(u, y) - y_r - u_r}{y_r + u_r} \right) \right\} \tag{6.2}$$

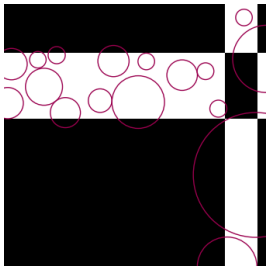
$$V_{expnd}(y) = \frac{y_r - r_{min}}{r_{max} - r_{min}} \tag{6.3}$$

$$V_{noise}(y) = v_y, \quad v_y \sim \mathcal{N}(0, 1) \tag{6.4}$$

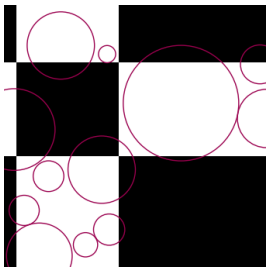
The position potential (6.1) samples the value of the image at the object center location, overlap potential (6.2) assigns high energy to overlapping circles, the *expansion* potential (6.3) favors circles of a larger size, and (6.4) gives a random energy to each point of the configuration (and as such is not a useful potential). To each we assign a corresponding weight w_{pos}, w_{ovrlp}, w_{expnd} and w_{noise}. We generate training samples by sampling the energy model with weights set as $w_{noise} = 0$, $w_{pos} = -2$, $w_{ovrlp} = 8, w_{expnd} = -1$, picked semi-arbitrarily to limit overlap while favoring big circles: some samples are shown in Figure 6.1 (a) and (b).

With these image and configuration pairs, we now use the contrastive divergence method (Algorithm 5.3) to estimate the weights used for generating the configurations. We plot the energy for positive samples ($\mathbf{y}^+$) and negative samples ($\mathbf{y}^-$) through the iterations of the training procedure (*epochs*) in Figure 6.1 (d) and the inferred weights in (e). We see that the noise prior – that is irrelevant to the Point Process energy mixture — remains with an almost zero weight (even through multiple iterations of the inference procedure, see Figure 6.1 (f)). Although the resulting weights are not exactly the ones used to generate the training data, both models (the data generator model, and the model with inferred weights) produce similar results as shown in Figure 6.1 (c); both models seemingly have similar distributions at low temperature.

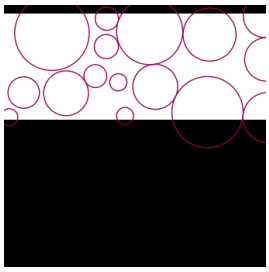
Remark 6.1.1 – For usual gradient descent methods, the loss should decrease monotonically through epochs. In this case, the loss is computed against a set of contrastive samples that is constantly updated; at each epoch new contrastive samples are generated, hopefully closer to the ground truth configurations. Here, as the model converges, we expect the positive ($U(\mathbf{y}^+)$)



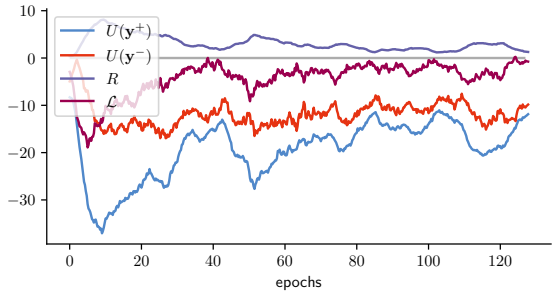
(a) train sample 1



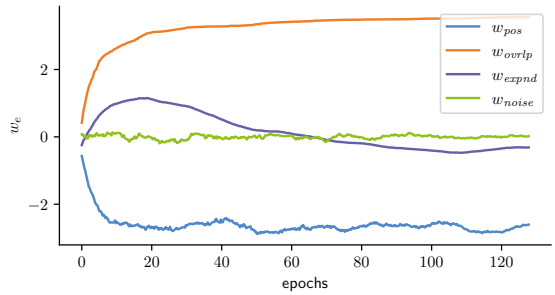
(b) train sample 2



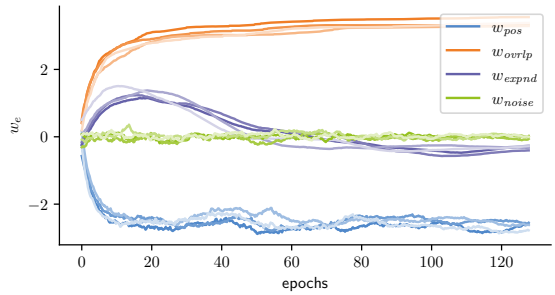
(c) inference



(d) energy and loss during training



(e) weights during training



(f) weights during training (multiple exp.)

Figure 6.1: Example of contrastive divergence for training. (a) and (b) show two training samples, a synthetic image with configuration of circles in lilac. (c) shows a configuration inferred from the model using the estimated parameters. In (d) we plot the loss and its components (see (5.29)) through the estimation procedure steps. (e) shows the weights value through the estimation procedure, as in (f) where we compare over multiple random initialization.

and negative ($U(\mathbf{y}^-)$) samples energies to progressively get closer. However, in Figure 6.1 (d) we observe a consistent offset between $U(\mathbf{y}^+)$ and $U(\mathbf{y}^-)$ after epoch 30; this is due to some high temperatures β in the mix used to generated $\mathbf{y}^-$ (see Section 5.2.2.3). At high temperatures the samples $\mathbf{y}^-$ are drawn closer to a uniform distribution in $\mathcal{Y}$, thus get rated with a higher energy.

6.1.2 Stable perturbation kernel

Within the contrastive divergence training procedure in Algorithm 5.3 the sampling of negative sample $\mathbf{y}^-$ is very costly. One can wonder whether we could instead generate negative samples $\mathbf{y}^-$ with a *static* kernel Q^- instead; reducing computational time and complexity while maintaining the advantages of the gradient descent approach to estimate parameters θ.

In this ablation study, we place ourselves in a simple setting similar to the energy model and training data described in previous Section 6.1.1. We propose three different experimental settings:

- **Contrastive**: using Algorithm 5.3 as such.
- **Static uniform**: replacing the costly line 7 in Algorithm 5.3 by $\mathbf{y}^- \sim Q^-(y^+, \cdot)$, where perturbation kernel Q^- shifts location and marks of points with a random Gaussian, removes points at random, and proposes new points uniformly in $\mathcal{S} \times \mathcal{M}$.
- **Static density**: same as previously, but new points are proposed with birth density d defined in (5.12).

We show results in Figure 6.2: from this simple text we see that Algorithm 5.3 fails to learn an energy that can generate proper point configurations when the negative samples are not generated with $\mathbf{y}^- \sim \exp(-U(\cdot, \mathbf{X}, \theta))$.

6.2 Results on optical data

Our application goal is the detection of small objects in images from satellites such as Pléiades-HR (*Pleiades-HR*, 2012) (0.7m resolution), Pléiades Neo (*Pléiades Neo*, n.d.) (0.3m resolution) or CO3D (*CO3D Constellation*, 2023) (0.5m resolution). Within the LiChIE project and our collaboration with Airbus Defense and Space, we set the resolution for our application to 0.5m.

6.2.1 COWC with non-marked Point Process

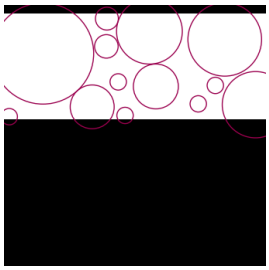
6.2.1.1 Data

Our first model published in (Mabon et al., 2021) is trained and tested on Cars Overhead With Context (COWC)[1] (Mundhenk et al., 2016).

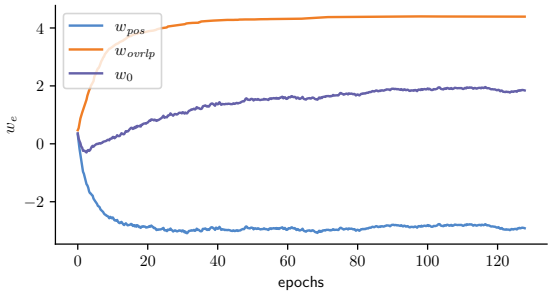
The dataset contains images of several urban areas, for which the centers of vehicles are labeled in their center. This dataset is quite diversified as it shows dense urban areas as well as forested areas or fields.

As we aim to develop detection model for satellite imagery, we sub-sample images to as resolution of 0.50m and refer to the resulting dataset as $COWC_{0.5}$. We use an anti-aliasing filter when

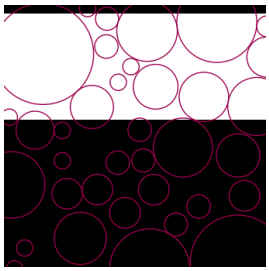
[1] Available at: https://gdo152.llnl.gov/cowc/



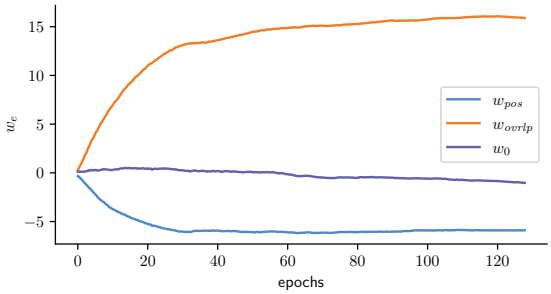
(a) result: contrastive



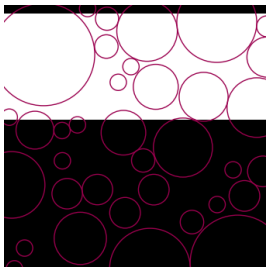
(d) weights at training: contrastive



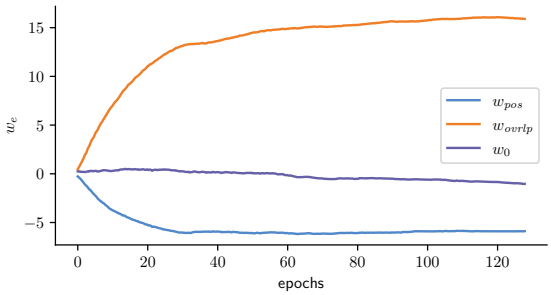
(b) result: static uniform



(e) weights at training: static uniform



(c) result: static density



(f) weights at training: static density

Figure 6.2: Inference results on a test image (a to c) and weights during training (d to f) for each experimental setup.

sub-sampling in order to avoid artifacts on object edges and better approximate the acquisition of a lower resolution image. We only use the RGB images of the dataset (no grayscale). For each urban area we split it into two thirds for training and one third for testing.

6.2.1.2 Point Process model

As the dataset only contains object location and no geometric data, we consider a Point Process without marks $y \in \mathcal{S}$.

The energy model contains the following energies, with corresponding weights:

- Position potential V_{pos} as defined in (4.10).
- Overlap prior V_{ovrlp} from (4.37), where each point is considered as a disk of constant diameter a set as the typical car width in pixels at resolution 0.5m.
- Triplet alignment potential V_{triAl} defined in (4.45).

The total energy is then given as:

$$U(\mathbf{y}, \mathbf{X}) = \sum_{y \in \mathbf{y}} w_{pos} V_{pos}(y, \mathbf{X}) + w_{ovrlp} V_{ovrlp}(y, \mathcal{N}^{\mathbf{y}}_{\{y\}}) + w_{triAl} V_{triAl}(y, \mathcal{N}^{\mathbf{y}}_{\{y\}}). \tag{6.5}$$

6.2.1.3 Parameter estimation and inference

To estimate the weights $w_{pos}, w_{ovrlp}, w_{triAl}$, we use the linear SVM methods as described in Section 5.2.1.2. We generate positive samples $\mathbf{y}^+$ from e ground truth configuration $\mathbf{y}^{GT}$ for each element of the training data with random Gaussian perturbation of low amplitude. Negative samples $\mathbf{y}^-$ are generated from uniform sampling of points or high amplitude Gaussian perturbation or random point removal from $\mathbf{y}^{GT}$.

The estimate $\hat{\mathbf{y}}$ of optimal configuration $\mathbf{y}^* = \arg\min_{\mathbf{y} \in \mathcal{Y}} U(\mathbf{y}, \mathbf{X})$ is obtained with the RJMCMC and simulated annealing method using the following kernels:

- Birth and Death kernels as introduced in 3.3.2.3 with non-uniform proposal density based on V_{pos}.
- A transform kernel based on data as described in Section 5.1.1, using the energy map from V_{pos}.

6.2.1.4 Results

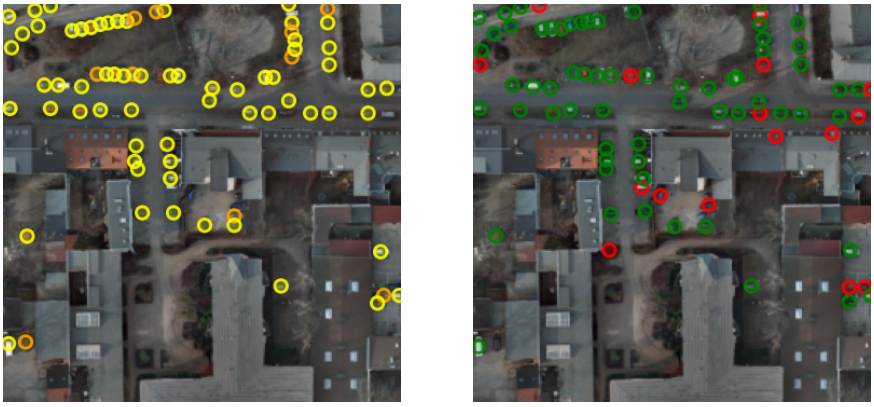
Metrics. To measure the validity of an estimated point configuration $\hat{\mathbf{y}}$ against the ground truth $\mathbf{y}^{GT}$ we try to match every ground truth center with estimated points within a radius $r = a/2$ as in (Mundhenk et al., 2016). Thus, an object is considered a *detection* if it matches to at least one point from $\hat{\mathbf{y}}$. An object of $\mathbf{y}^{GT}$ is a *False Negative* (FN) if it corresponds to no point in $\hat{\mathbf{y}}$. A point of $\hat{\mathbf{y}}$ is a *False Positive* (FP) if there is no matching object in $\mathbf{y}^{GT}$. If multiple point in $\hat{\mathbf{y}}$ match the same element of $\mathbf{y}^{GT}$, we count one *True Positive* (TP) and every extra matching point as FP. Lastly if one detected element in $\hat{\mathbf{y}}$ matches to multiple ground truth elements, we set the match to the closest ground truth object, if the remaining ground truth objects have no other matching detections they are counted as FN. From the TP, FP and FN we can compute the precision, recall and F1 score for an estimated configuration $\hat{\mathbf{y}}$ as shown in Table 6.1.

Precision (Pr), Recall (Rc) and F1 scores for a given score threshold t are given as:

$$Pr(t) = \frac{TP(t)}{TP(t) + FP(t)} \tag{6.6}$$

$$Rc(t) = \frac{TP(t)}{|\mathbf{y}^{GT}|} \tag{6.7}$$

$$F1(t) = 2\frac{Pr(t)Rc(t)}{Pr(t) + Rc(t)} \tag{6.8}$$



(a) ground truth (b) estimated configuration

nb. objects	TP	FP	FN	Precision	Recall	F1
74	48	17	12	0.73	0.64	0.68

Figure 6.3: Detection on a sample of $COWC_{0.5}$: (a) ground truth, (yellow : detected vehicle, orange: false negative) (b) estimated configuration $\hat{\mathbf{y}}$ (green: true positive, red: false positive). Table shows metrics on this patch.

Location	**No. obj.**	**Prec.**	**Rec.**	**F1**	**File**
Potsdam	425	0.74	0.66	0.70	`top_potsdam_6_9_RGB`
Potsdam	262	0.56	0.48	0.52	`top_potsdam_6_8_RGB`
Potsdam	110	0.76	0.68	0.72	`top_potsdam_6_7_RGB`
Selwyn	121	0.87	0.50	0.75	`Selwyn_BX22_Tile_RIGHT_15cm_0003`
Toronto	3969	0.87	0.63	0.74	`03747`

Table 6.1: Precision, recall and F1 on each test image of $COWC_{0.5}$.

6.2.2 Point Process of rectangles on DOTA and ADS data

6.2.2.1 Data

DOTA. We compile $DOTA_{0.5}$, a 0.5 meter per pixel vehicles in remote sensing dataset, from the DOTA dataset (Xia et al., 2018), and use it to train the CNN backbone, and infer the model parameters $\widehat{\theta}$. This dataset is built by sub-sampling images from DOTA to the desired spatial resolution.

To test the resilience of our models against noise, we test the methods on the same data adding Gaussian noise (with $\sigma = 0.3$ on input image $\mathbf{X}$ at inference, see Figure 6.4). We denote this dataset $DOTA_{0.5}^{\mathcal{N}}$.

ADS data. Moreover, we evaluate models on data provided by Airbus Defense and Space (ADS) of two areas around Lyon, this aerial data is sub-sampled to the desired resolution (0.5m), emulating the CO3D satellite characteristics. This dataset is unlabeled, thus will only serve to evaluate the models qualitatively, and was not used for training/parameters estimation.

6.2.2.2 Models

In this section we show results both on CNN based models, and PP models on the $DOTA_{0.5}$ dataset. The PP models use the following energies from Table 6.2:

	Energy term	Notation	Instances	Parameters	Eq. No.
DT[a]	position	V_{pos}	–	t_{pos}	(4.26)
	mark	V_{κ}	$\kappa \in \{a, b, \alpha\}$	–	(4.30)
Prior Terms	joint area & ratio	V_{jntAR}	$k \in \{car, truck\}$	$\mu_{k,ratio}$, $\mu_{k,area}$, $\sigma_{k,ratio}$, $\sigma_{k,area}$	(4.34)
	overlap	V_{ovrlp}	–	t_{ovrlp}	(4.37)
	alignment	V_{align}	$t_{align} \in \{0, \frac{\pi}{2}\}$	–	(4.38)
	repulsive	V_{repls}	–	t_{repls}	(4.39)
	attractive	V_{attrc}	–	t_{attrc}	(4.40)
	zero neighbor	V_{zrNbr}	–	–	(4.41)

[a] Data Terms

Table 6.2: Energy terms of the model.

To each energy term e we assign a respective weight w_e as in (4.3). For some above-mentioned energies we use multiple instances; e.g. we have one mark energy for each mark $\kappa \in \{a, b, \alpha\}$. Similarly, we have one alignment ratio for parallel objects ($t_{align} = 0$) and one for perpendicular objects $t_{align} = \pi/2$.

The models tested on the test dataset are as follows:

- *CNN-LocalMax.*: naive detection from the CNN backbone (used in the PP models); we find objects through local maxima in the output probability maps (or local minima in the energy maps).
- *CNN-PP*$^{\blacklozenge}$: PP model with minimal inferred parameters: inferred parameters θ are the energy weights $\{w_e,\ e \in \xi\}$. The energy term parameters in the table above are set manually.
- *CNN-PP**: PP model with parametrized priors: inferred parameters θ include both the energy weights $\{w_e,\ e \in \xi\}$ and the energy term parameters in the table above.
- *BBA-Vec.* and *YOLOV5-OBB*: lastly, we compare all of our models above with *BBA-Vec.* from (Yi et al., 2021) and *YOLOV5-OBB* from (Yang & Yan, 2022; Jocher et al., 2022). These models are trained on the same data as the above-mentioned models.

6.2.2.3 Parameter estimation and inference

Estimation of energy parameters. The parameters θ for both PP methods *CNN-PP*$^{\blacklozenge}$ and *CNN-PP** are estimated with the Algorithm 5.3. The procedure is applied over 512 epochs (iterations on the train data which consists of 2048 ground truth patches of size $(128, 128)$). Temperatures for sampling negative/contrastive configurations are picked as $\beta_1 = 1.0$, $\beta_2 = 10^{-2}$ and $\beta_3 = 10^{-4}$. We set the buffer size $n_{\mathcal{B}} = 1$ and its probability $p_{\mathcal{B}} = 0.99$. The regularization weight is set to $\gamma = 1$ for the loss in (5.29).

Inferring configurations from images. At inference, the configuration $\hat{\mathbf{y}}$ is obtained through Algorithm 5.1. We set a number of iterations per pixel $n_{ipp} = 2.0$, with the expected number of cells being process in parallel $n_{/\!/} = \mathbb{E}[|\tilde{s}|]$, we get the number of iterations for an image $\mathbf{X}$ of size (H, W):

$$n_s = \frac{n_{ipp} HW}{n_{/\!/}}. \tag{6.9}$$

The initial temperature is set to $T_0 = 0.1\varsigma$; i.e. the temperature is set to 0.1 for a unit variance model, and then scaled with ς computed from Algorithm 5.2. The annealing rate α_T is set so that $T_{n_s} = 10^{-10}$ as:

$$\alpha_T = \left(\frac{T_{n_s}}{T_0}\right)^{\frac{1}{n_s}} \tag{6.10}$$

6.2.2.4 Results

Method	AP	
	$DOTA_{0.5}$	$DOTA^{\mathcal{N}}_{0.5}$
BBA-Vec.	0.82	0.19
YOLOV5-OBB	0.86	0.10
CNN-LocalMax.	0.86	0.55
CNN-PP$^{\blacklozenge}$	**0.91**	**0.58**
CNN-PP*	**0.92**	**0.62**

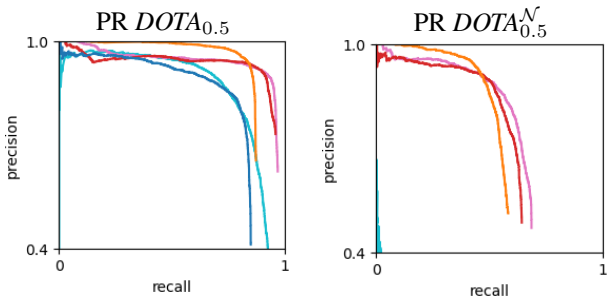


Table 6.3: Average Precision (AP) values, respectively for default $DOTA_{0.5}$ and noisy $DOTA^{\mathcal{N}}_{0.5}$ data. Right : Precision Recall (PR) curves, color correspondence in left table.

The above models are evaluated on a test split of the $DOTA_{0.5}$ dataset. Metrics are computed by matching the object proposals and Ground Truth using the IOU (Intersection Over Union), with a 0.25 threshold. We show detection results in Figure 6.4, with a score threshold set for each model to the one maximizing the F1 score. The Average Precision (AP) metrics (which are threshold independent), are shown in Table 6.3.

We notice in Figure 6.4, that although Ground Truth labels are noisy — which could have caused over-constraining problem with linear programming (see Section 5.2.1) — our models infer more regular configurations, that better fit the objects and their physical constraints.

Testing the models on noisy data in Figure 6.5 and on the *ADS* data in Figure 6.6 (we show highlights in Figure 6.7), shows that the added priors allow the PP based models to be more

Figure 6.4: Samples of detection on the test dataset for various models. The score threshold (only used for display purposes to prune low score objects) is set to maximize the $F1$ score for each model.

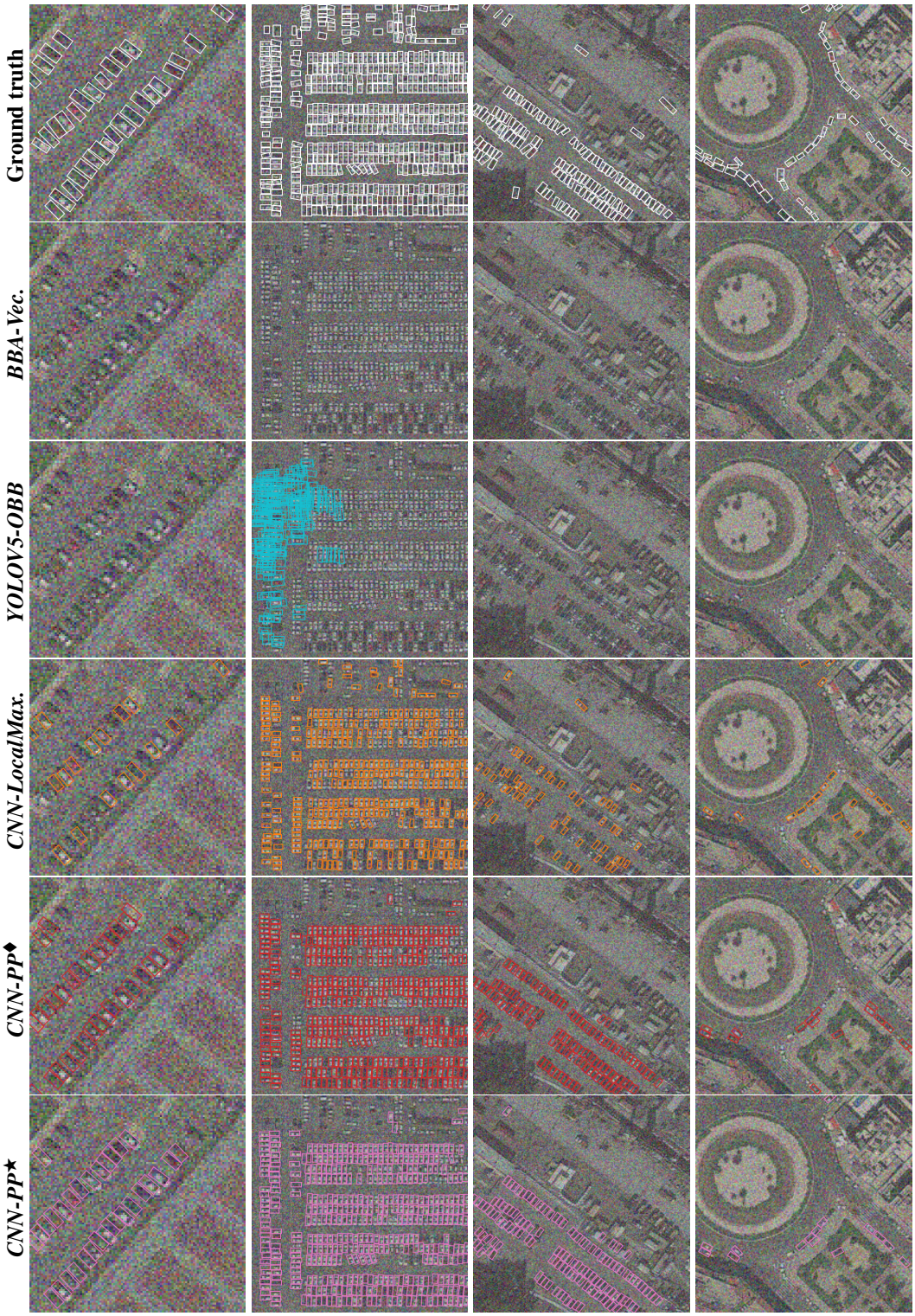


Figure 6.5: Samples of detection on the test dataset for various models with added noise in the input image. The score threshold (only used for display purposes to prune low score objects) is set to maximize the $F1$ score for each model.

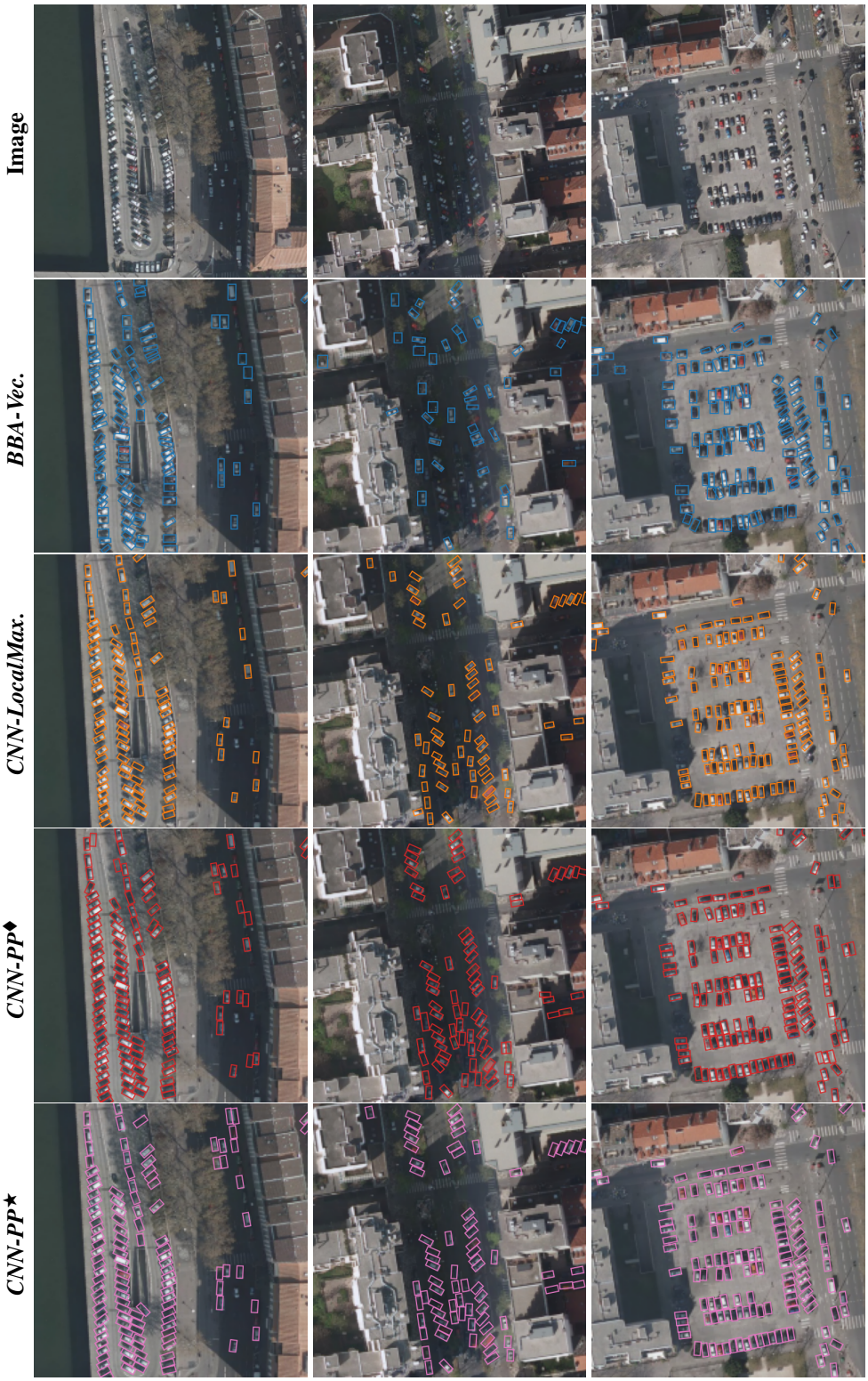


Figure 6.6: Models applied to *ADS* data. [© Airbus Defense and Space]

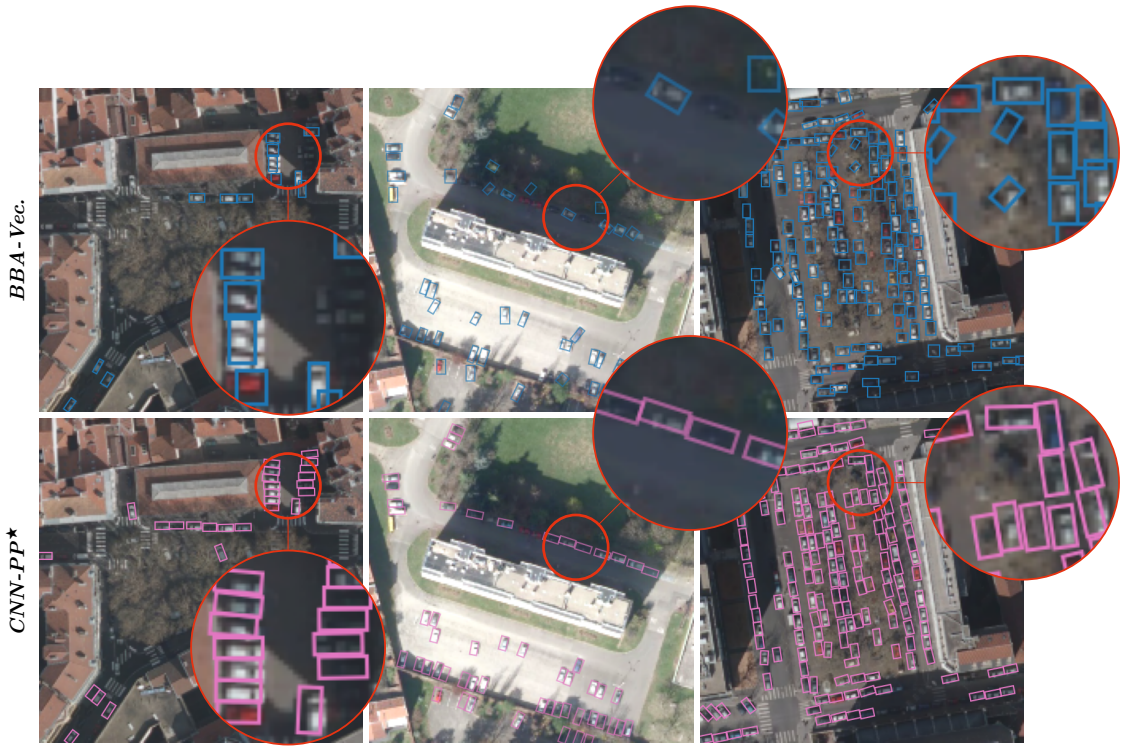


Figure 6.7: Model applied to *ADS* data. The second image (second column) is brightened for better display legibility. [© Airbus Defense and Space]

resilient to noise or areas of the image with limited information; e.g. low resolution, partial occlusions, shadows, as in Figure 6.6, line 2. The *ADS* data inference results are obtained with the models trained on the $DOTA_{0.5}$ training set which in top shows the capacity of the model to generalize to unseen (although similar) data.

Lastly, *CNN-PP*♦ and *CNN-PP*★ show comparable performances (qualitatively and quantitatively), demonstrating the capability of the contrastive divergence method to infer the prior/interaction model (via the prior parameters), thus limiting the manual setting of parameters via trial and error.

6.2.2.5 Computational complexity

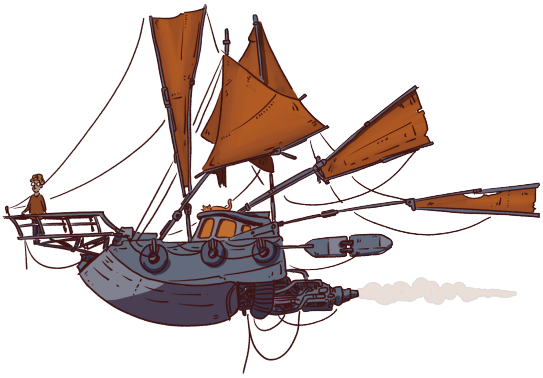
Our two MPP based methods — running on an *Nvidia Quadro RTX 8000* Graphics Processing Unit — execute in 300s on average, for 16k parallel iterations (equivalent to 77k sequential iterations) for one image. With an efficient implementation, considering a density of 1.9×10^{-3} (95th percentile of the observed object densities[2]), we estimate the cost of one iteration to be around 5 operations per pixel per iteration, thus around 4×10^5 operations per pixel in total. We show the derivation of those values in Appendix H. As a point of comparison transformer models for image classification range from 5×10^6 to 1.8×10^7 operations (Y. Zhao et al., 2021). The

[2]i.e. 95% of observed object densities are below this value.

complexity could be greatly reduced by reducing the number of iterations of the Point Process sampler, at the cost of straying away from the proper convergence properties.

6.3 Conclusion

In this chapter we applied our methods on remote sensing data sets; both on COWC (Mundhenk et al., 2016) and DOTA (Xia et al., 2018) that are publically available, and on some data provided by Airbus Defense and Space. Even with the limited complexity of our models, we achieved good qualitative results by ensuring regularity in the inferred geometry. Quantitatively our models show better results than other CNN based methods. The latter is more pronounced with noisy input data: our models maintain good qualitative and quantitative results even in adverse noise conditions. We also demonstrate the proposed parameter estimation method consistently matches or outperforms slightly the manual setting of model parameters, qualitatively and quantitatively, with the intrinsic advantage of limiting tedious manual trial and error. Lastly we show our model can generalize to new data by looking at qualitative results on ADS data after training on DOTA.



CHAPTER 7

Conclusion and Perspectives

7.1 Conclusion

In this PhD thesis we address the incorporation of interaction models in object detection methods, while leveraging the capabilities of deep convolutional neural networks. Satellite images are inherently flawed due to the atmospheric perturbations, partial occlusions and limited spatial resolution. To compensate for this lack of visual information, it becomes essential to incorporate prior knowledge about the arrangement of objects of interest.

On one hand CNN based methods are great at extracting patterns from images, but struggle to learn object to object interaction models without introducing attention mechanisms such as Transformers that drastically increase the complexity of the model: for instance, (Lu et al., 2023) proposed to include priors as text-model descriptors, while (Zeng et al., 2023) made use of cascades of attention modules.

On the other hand, Point Processes propose to jointly solve for both the object likelihoods relative to the image (data term), and the coherence of the arrangement of objects itself (prior term). Firstly, Point Process approaches model the configurations as vectorial geometry, constraining the state space by construction. Secondly, these models allow formulating explicit priors on configurations as energy functions. However, in most of the literature (Verdié & Lafarge, 2014; Schmidt et al., 2017), the data terms relied on contrast measures between the foreground and background. We showed the limitations of those on satellite data in this manuscript. Although we should note (T. Li et al., 2019) proposed transforming the original image with a CNN trained for classification, to then compute the contrast measures on that simplified result.

Instead of going the Transformers/complexity route, we proposed in this thesis to combine the CNN pattern extraction with the Point Process framework. The basic idea of this approach is to use the CNN output as the data term for a Point Process model. At first, we set up an original contrast-like method (pattern matching using cross correlation) on the output of CNN trained to map blobs at object centers. We then proposed a novel method using a CNN trained to infer a field of vectors that point towards the nearest object center. To obtain a heatmap we simply need to compute the divergence of this field. This allows to easily separate close objects at a low computational cost. Finally, we showed how we can reinterpret any heatmap-based CNN output as an energy that can be used for our model, both for the position and marks of each object. With our formulation of the data energy as the interpolation of a potential map, we transformed the contrast measure computation, into a simple lookup and interpolation of values in a 2D or 3D tensor.

With these pre-computed potential maps, we proceeded in adapting the Point Process sampling methods to make use of that easily available information. Indeed, these tensors define densities over a discrete space that is easy to sample from, and that approximates a truncated version of the energy model (which is defined in continuous space). First, we applied this data-guided sampling with the local transform perturbation kernel, and then built a birth map. These spatial densities also allow us to focus the parallel sampling of the Point Process to areas where the number of point is likely to be higher and thus requires more proposals. Finally, we leveraged the easily available automatic differentiation engines, to compute the energy gradient and perform diffusion for a more efficient exploration of the state space at a fixed number of points.

Previous methods would use Linear Programming to estimate the relative weights of the energies of the model (Q. Yu & Medioni, 2009; Craciun et al., 2015) from a set of constraints. However, this is prone to over-constraining and only takes into account a limited amount of data points. Our first parameter estimation resulted in an SVM based method, that looks for the best possible separation between positive and negative samples. Thanks to its permeable boundary, this method allows to consider many more data points, even noisy ones that would contradict with hard constraints. Ultimately, we proposed adapting a method used for Energy Based Models: the contrastive divergence (Hinton, 2002; Du & Mordatch, 2019). Within a gradient descent scheme, it alternates between generating contrastive/negative samples from the current energy, and maximizing the energy difference between positive samples (similar to the ground truth) and the contrastive samples. The procedure we presented allows estimating not only energy weights, but also the various internal parameters of the energy terms. Otherwise, those would have to be set manually (by trial and error) or with a separate procedure for each parameter of each energy term.

To be able to compare our model with other CNN based object detection models we introduced a scoring function for the objects inferred by the Point Process model. This score is computed from the Papangelou conditional intensity, which measures the likelihood of a point given the rest of the configuration. This is a key difference from classical scoring functions that only consider individual points, regardless of their neighborhood.

We apply our methods on remote sensing data sets; both on COWC (Mundhenk et al., 2016) and DOTA (Xia et al., 2018) that are publically available, and some data provided by Airbus Defense and Space. Even with the limited complexity of our models, we achieved good qualitative results by ensuring regularity in the inferred geometry. Quantitatively our models are better than some other recent CNN based methods. The improvement of results is more pronounced with noisy input data: our models maintained good qualitative and quantitative results even in adverse noise conditions. We also demonstrated the proposed parameter estimation method consistently matches or outperforms slightly the manual setting of model parameters, qualitatively and quantitatively, with the intrinsic advantage of limiting tedious manual trial and error procedures. Lastly we show our model can generalize to new data by looking at qualitative results on ADS data after training on DOTA.

7.2 Perspectives

Due to the iterative nature of MCMC sampling methods, and the limited optimization of our model (mainly to allow to modularity while experimenting), the inference time is significantly longer that state-of-the-art CNN based object detection methods. Speeding up the inference could be achieved by using fewer iterations, initializing the chain with an approximate inference (e.g.

the *CNN-LocalMax.* output shown in Section 6.2.2) along with a faster simulated annealing. This would yield faster results at the cost of straying further from the optimal convergence conditions. Other solutions exist such as discretizing the state space and solving a binary variable optimization problem (T. T. Pham et al., 2016).

While in this thesis we focused on small vehicle detection, the framework we proposed can be easily adapted to any other modality of objects where interactions between objects of interest are a key element. For instance road networks can be modeled with Point Processes of segments as in (Lacoste et al., 2005), and have important priors on their interactions (e.g. limited bend radius). Point Processes can also be applied to temporal data; (Craciun et al., 2015) performed tracking in remote sensing data. In this case, priors on dynamics are strong and could be able to complement the lack of visual information and even complete occlusions (e.g. car going under a bridge). Also, as our model is quite resilient to noise, we expect it would perform well on SAR data, as the input noise is more important with this type of data.

Future work could focus on the composition capabilities of such energy models: in short, it would be feasible to estimate the data energy functions on one set of data, and the prior energy model on another one, to then compose the two and form a complete model. In practice this would allow to learn a data model on imperfect annotations, and learn a prior model on *perfect* synthetic data that reproduces the expected structures; here we would only need configurations of objects, no difficult to produce images.

Finally, the proposed parameter estimation method is not bound to object detection: we noted earlier the generative nature of our Point Process model, thus we could apply it to learning patterns and structures of points in order to produce a generative model that will be capable of imitating patterns as done in (Hurtut et al., 2009). The latter could benefit from non-application-specific prior model that would be learned on the data, such as we propose in Appendix G: using a universal approximator for the per-object prior, and learning aggregation operators over the interactions using attention mechanisms.

Bibliography

Antonelli, S., Avola, D., Cinque, L., Crisostomi, D., Foresti, G. L., Galasso, F., ... Pannone, D. (2022, September). Few-Shot Object Detection: A Survey. *ACM Computing Surveys*, *54*(11s), 242:1–242:37. doi: 10.1145/3519022

Baddeley, A. J., & Lieshout, M. N. M. V. (1993, January). Stochastic geometry models in high-level vision. *Journal of Applied Statistics*, *20*(5-6), 231–256. doi: 10.1080/02664769300000065

Bahdanau, D., Cho, K., & Bengio, Y. (2015, May). Neural Machine Translation by Jointly Learning to Align and Translate. In *International Conference on Learning Representations (ICLR).* San Diego, USA. doi: 10.48550/arXiv.1409.0473

Bai, J., Ren, J., Xiao, Z., Chen, Z., Gao, C., Ali, T. A. A., & Jiao, L. (2023, September). Localizing From Classification: Self-Directed Weakly Supervised Object Localization for Remote Sensing Images. *IEEE Transactions on Neural Networks and Learning Systems*, 1–15. doi: 10.1109/TNNLS.2023.3309889

Bai, M., & Urtasun, R. (2017, July). Deep Watershed Transform for Instance Segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 2858–2866). doi: 10.1109/CVPR.2017.305

Belhadji, A., Bardenet, R., & Chainais, P. (2019, December). Kernel quadrature with DPPs. In *Advances in Neural Information Processing Systems (NeurIPS)* (Vol. 32). Vancouver, Canada: Curran Associates, Inc. Retrieved 2023-09-15, from https://proceedings.neurips.cc/paper_files/paper/2019/hash/7012ef0335aa2adbab58bd6d0702ba41-Abstract.html

Belhadji, A., Bardenet, R., & Chainais, P. (2020, January). A determinantal point process for column subset selection. *The Journal of Machine Learning Research*, *21*(1), 197:8083–197:8144. Retrieved from https://www.jmlr.org/papers/volume21/19-080/19-080.pdf

Belmouhcine, A., Burnel, J.-C., Courtrai, L., Pham, M.-T., & Lefèvre, S. (2023, July). Multimodal Object Detection in Remote Sensing. In *IEEE International Geoscience and Remote Sensing Symposium (IGARSS).* Pasadena, USA. doi: 10.48550/arXiv.2307.06724

Ben Hadj, S., Chatelain, F., Descombes, X., & Zerubia, J. (2010, August). *Estimation des paramètres de modèles de processus ponctuels marqués pour l'extraction d'objets en imagerie spatiale et aérienne haute résolution* (Research report). INRIA. Retrieved from https://inria.hal.science/inria-00508431

Beucher, S., & Lantuejoul, C. (1979, September). Use of watersheds in contour detection. In *International Workshop on image processing* (pp. 17–21). Rennes, France. Retrieved from http://cmm.ensmp.fr/~beucher/publi/watershed.pdf

Bochkovskiy, A., Wang, C.-Y., & Liao, H.-Y. M. (2020, April). *YOLOv4: Optimal Speed and Accuracy of Object Detection* (No. arXiv:2004.10934). arXiv. doi: 10.48550/arXiv.2004.10934

Bondi, E., Jain, R., Aggrawal, P., Anand, S., Hannaford, R., Kapoor, A., ... Tambe, M. (2020, March). BIRDSAI: A Dataset for Detection and Tracking in Aerial Thermal Infrared Videos. In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV)* (pp. 1747–1756). Snowmass Village, USA. Retrieved from https://openaccess.thecvf.com/content_WACV_2020/html/Bondi_BIRDSAI_A_Dataset_for_Detection_and_Tracking_in_Aerial_Thermal_WACV_2020_paper.html

Bordenave, C., Gousseau, Y., & Roueff, F. (2006, March). The dead leaves model: A general tessellation modeling occlusion. *Advances in Applied Probability*, *38*(1), 31–46. doi: 10.1239/aap/1143936138

Bottou, L. (2012). Stochastic gradient descent tricks. In G. Montavon, G. B. Orr, & K.-R. Müller (Eds.), *Neural networks: Tricks of the trade: Second edition* (pp. 421–436). Berlin, Heidelberg: Springer. doi: 10.1007/978-3-642-35289-8_25

Bozcan, I., & Kayacan, E. (2020, May). AU-AIR: A Multi-modal Unmanned Aerial Vehicle Dataset for Low Altitude Traffic Surveillance. In *IEEE International Conference on Robotics and Automation (ICRA)* (pp. 8504–8510). doi: 10.1109/ICRA40945.2020.9196845

Buslaev, A., Iglovikov, V. I., Khvedchenya, E., Parinov, A., Druzhinin, M., & Kalinin, A. A. (2020, February). Albumentations: Fast and Flexible Image Augmentations. *Information*, *11*(2), 125. doi: 10.3390/info11020125

Campbell, J. B., & Wynne, R. H. (2011). *Introduction to Remote Sensing, Fifth Edition*. Guilford Press. Retrieved from https://books.google.fr/books?id=NkLmDjSS8TsC

Cao, Y., Bai, Y., Pang, R., Liu, B., & Zhang, K. (2023, February). Vehicle Detection Algorithm Based on Background Features Assistance in Remote Sensing Image. *Sensors and Materials*, *35*(2), 607. doi: 10.18494/SAM4204

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., & Zagoruyko, S. (2020). End-to-End Object Detection with Transformers. In A. Vedaldi, H. Bischof, T. Brox, & J.-M. Frahm (Eds.), *Computer Vision – ECCV* (pp. 213–229). Cham: Springer International Publishing. doi: 10.1007/978-3-030-58452-8_13

Castillo-Navarro, J., Le Saux, B., Boulch, A., & Lefèvre, S. (2021, November). Energy-Based Models in Earth Observation: From Generation to Semisupervised Learning. *IEEE Transactions on Geoscience and Remote Sensing*, *60*, 1–11. doi: 10.1109/TGRS.2021.3126428

Celeux, G., Chauveau, D., & Diebolt, J. (1995, March). *On Stochastic Versions of the EM Algorithm* (Research Report). INRIA. Retrieved 2023-09-13, from https://inria.hal.science/inria-00074164

Chainais, P., Kœnig, É., Delouille, V., & Hochedez, J.-F. (2011, January). Virtual Super Resolution of Scale Invariant Textured Images Using Multifractal Stochastic Processes. *Journal of Mathematical Imaging and Vision*, *39*(1), 28–44. doi: 10.1007/s10851-010-0222-6

Chatelain, F., Descombes, X., & Zerubia, J. (2009). Parameter Estimation for Marked Point Processes. Application to Object Extraction from Remote Sensing Images. In D. Cremers, Y. Boykov, A. Blake, & F. R. Schmidt (Eds.), *Energy Minimization Methods in Computer Vision and Pattern Recognition* (pp. 221–234). Berlin, Heidelberg: Springer. doi: 10.1007/978-3-642-03641-5_17

Cherel, N., Almansa, A., Gousseau, Y., & Newson, A. (2022, September). Attention stochastique basée patchs pour l'édition d'images. In *GRETSI 2022 - XXVIIIème Colloque Francophone de Traitement du Signal et des Images.* Retrieved from https://gretsi.fr/data/colloque/pdf/2022_cherel952.pdf

Corsel, C. W., van Lier, M., Kampmeijer, L., Boehrer, N., & Bakker, E. M. (2023, January). Exploiting Temporal Context for Tiny Object Detection. In *IEEE/CVF Winter Conference on Applications of Computer Vision Workshops (WACVW)* (pp. 1–11). Waikoloa, USA: IEEE. doi: 10.1109/WACVW58289.2023.00013

Cortes, C., & Vapnik, V. (1995, September). Support-vector networks. *Machine Learning*, *20*(3), 273–297. doi: 10.1007/BF00994018

Cox, D. R., & Isham, V. (1980). *Point Processes*. CRC Press. Retrieved from https://books.google.fr/books?id=KWF2xY6s3PoC

Crăciun, P. (2015). *Stochastic geometry for automatic multiple object detection and tracking in remotely sensed high resolution image sequences* (These de Doctorat, Université Nice Sophia Antipolis). Retrieved 2020-11-13, from https://tel.archives-ouvertes.fr/tel-01235255

Craciun, P., Ortner, M., & Zerubia, J. (2015, January). Joint Detection and Tracking of Moving Objects Using Spatio-temporal Marked Point Processes. In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV)* (pp. 177–184). Waikoloa, USA. doi: 10.1109/WACV.2015.31

Daley, D., & Vere-Jones, D. (1988). *An introduction to the theory of point processes*. Springer-Verlag. doi: 10.1007/b97277

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum Likelihood from Incomplete Data Via the EM Algorithm. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, *39*(1), 1–22. doi: 10.1111/j.2517-6161.1977.tb01600.x

Descamps, S., Descombes, X., Bechet, A., & Zerubia, J. (2008, March). Automatic Flamingo detection using a multiple birth and death process. In *IEEE International Conference on Acoustics, Speech and Signal Processing* (pp. 1113–1116). doi: 10/fb794f

Descombes, X. (2013). *Stochastic geometry for image analysis*. Wiley. Retrieved from https://books.google.fr/books?id=gzIwNet5aNQC

Descombes, X. (2017, February). Multiple objects detection in biological images using a marked point process framework. *Methods*, *115*, 2–8. doi: 10.1016/j.ymeth.2016.09.009

Descombes, X., Minlos, R., & Zhizhina, E. (2009, March). Object Extraction Using a Stochastic Birth-and-Death Dynamics in Continuum. *Journal of Mathematical Imaging and Vision*, *33*(3), 347–359. doi: 10/bt68mj

Descombes, X., & Zerubia, J. (2002, September). Marked point process in image analysis. *IEEE Signal Processing Magazine*, *19*(5), 77–84. doi: 10.1109/MSP.2002.1028354

Du, Y., Li, S., Tenenbaum, J., & Mordatch, I. (2021, July). Improved Contrastive Divergence Training of Energy-Based Models. In *38th International Conference on Machine Learning* (pp. 2837–2848). PMLR. Retrieved 2023-09-11, from https://proceedings.mlr.press/v139/du21b.html

Du, Y., & Mordatch, I. (2019, December). Implicit generation and modeling with energy based models. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems (NeurIPS)* (Vol. 32). Vancouver, Canada: Curran Associates, Inc. Retrieved from https://proceedings.neurips.cc/paper_files/paper/2019/file/378a063b8fdb1db941e34f4bde584c7d-Paper.pdf

Duda, R. O., & Hart, P. E. (1972, January). Use of the Hough transformation to detect lines and curves in pictures. *Communications of the ACM*, *15*(1), 11–15. doi: 10.1145/361237.361242

Eldin, A. G., Descombes, X., Charpiat, G., & Zerubia, J. (2012). Multiple Birth and Cut Algorithm for Multiple Object Detection. *Journal of Multimedia Processing and Technologies*. Retrieved 2021-02-04, from https://hal.archives-ouvertes.fr/hal-00616371

Geman, S., & Hwang, C.-R. (1986, July). Diffusions for Global Optimization. *SIAM Journal on Control and Optimization*, *24*(5), 1031–1043. doi: 10.1137/0324060

Geyer, C. J., & Møller, J. (1994, December). Simulation Procedures and Likelihood Inference for Spatial Point Processes. *Scandinavian Journal of Statistics*, *21*(4), 359–373. Retrieved 2023-10-09, from https://www.jstor.org/stable/4616323

Girshick, R. (2015, December). Fast R-CNN. In *Proceedings of the IEEE/CVF international conference on computer vision (ICCV)* (pp. 1440–1448). Santiago, Chile. Retrieved 2023-10-09, from https://openaccess.thecvf.com/content_iccv_2015/html/Girshick_Fast_R-CNN_ICCV_2015_paper.html

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014, June). 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). doi: 10.1109/CVPR.2014.81

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press. Retrieved from http://www.deeplearningbook.org

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... Bengio, Y. (2014, December). Generative adversarial nets. In Z. Ghahramani, M. Welling,

C. Cortes, N. Lawrence, & K. Weinberger (Eds.), *Advances in neural information processing systems (NIPS)* (Vol. 27). Montréal, Canada: Curran Associates, Inc. Retrieved from https://proceedings.neurips.cc/paper_files/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf

Goudail, F., Réfrégier, P., & Delyon, G. (2004, July). Bhattacharyya distance as a contrast parameter for statistical processing of noisy optical images. *Journal of the Optical Society of America A*, *21*(7), 1231–1240. doi: 10.1364/JOSAA.21.001231

Grathwohl, W., Wang, K.-C., Jacobsen, J.-H., Duvenaud, D., Norouzi, M., & Swersky, K. (2019, September). Your classifier is secretly an energy based model and you should treat it like one. In *International Conference on Learning Representations (ICLR)*. virtual. Retrieved 2023-10-09, from https://openreview.net/forum?id=Hkxzx0NtDB

Green, P. J. (1995, December). Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, *82*(4), 711–732. doi: 10/bt2s2t

Green, P. J., & Hastie, D. I. (2009, March). Reversible jump MCMC. *Genetics*, *155*(3), 1391–1403. Retrieved from https://people.maths.bris.ac.uk/~mapjg/papers/RJMCMC_30_Mar.pdf

Grenander, U., & Miller, M. I. (1994). Representations of Knowledge in Complex Systems. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, *56*(4), 549–581. doi: 10.1111/j.2517-6161.1994.tb02000.x

Guilmeau, T., Chouzenoux, E., & Elvira, V. (2021, July). Simulated Annealing: A Review and a New Scheme. In *IEEE Statistical Signal Processing Workshop (SSP)* (pp. 101–105). Rio de Janeiro, Brazil. doi: 10.1109/SSP49050.2021.9513782

Guo, C., Pleiss, G., Sun, Y., & Weinberger, K. Q. (2017, July). On Calibration of Modern Neural Networks. In *Proceedings of the 34th International Conference on Machine Learning* (pp. 1321–1330). Sydney, Autralia: PMLR. Retrieved 2023-10-09, from https://proceedings.mlr.press/v70/guo17a.html

Guo, Y., Wu, C., Du, B., & Zhang, L. (2022, July). Density Map-based vehicle counting in remote sensing images with limited resolution. *ISPRS Journal of Photogrammetry and Remote Sensing*, *189*, 201–217. doi: 10.1016/j.isprsjprs.2022.05.004

Hallou, A., Yevick, H. G., Dumitrascu, B., & Uhlmann, V. (2021, September). Deep learning for bioimage analysis in developmental biology. *Development*, *148*(18), dev199616. doi: 10.1242/dev.199616

Han, F., Tu, Z., & Zhu, S.-C. (2004, September). Range image segmentation by an effective jump-diffusion method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *26*(9), 1138–1153. doi: 10.1109/TPAMI.2004.70

He, H., Ding, J., & Xia, G.-S. (2023, August). *On the Robustness of Object Detection Models in Aerial Images*. arXiv. Retrieved from http://arxiv.org/abs/2308.15378

He, K., Zhang, X., Ren, S., & Sun, J. (2015, September). Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *37*(9), 1904–1916. doi: 10.1109/TPAMI.2015.2389824

Hershey, S., Chaudhuri, S., Ellis, D. P. W., Gemmeke, J. F., Jansen, A., Moore, R. C., ... Wilson, K. (2017, March). CNN architectures for large-scale audio classification. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 131–135). New Orleans, USA. doi: 10.1109/ICASSP.2017.7952132

Hinton, G. (2002, August). Training Products of Experts by Minimizing Contrastive Divergence. *Neural Computation*, *14*(8), 1771–1800. doi: 10.1162/089976602760128018

Hinton, G., Vinyals, O., & Dean, J. (2014, December). Distilling the Knowledge in a Neural Network. In *Advances in Neural Information Processing Systems Workshop (NIPS)*. arXiv. (Comment: NIPS 2014 Deep Learning Workshop) doi: 10.48550/arXiv.1503.02531

Ho, J., Jain, A., & Abbeel, P. (2020, December). Denoising Diffusion Probabilistic Models. In *Advances in Neural Information Processing Systems (NeurIPS)* (Vol. 33, pp. 6840–6851). virtual: Curran Associates, Inc. Retrieved from https://proceedings.neurips.cc/paper/2020/hash/4c5bcfec8584af0d967f1ab10179ca4b-Abstract.html

Hornik, K., Stinchcombe, M., & White, H. (1989, January). Multilayer feedforward networks are universal approximators. *Neural Networks*, *2*(5), 359–366. doi: 10.1016/0893-6080(89)90020-8

Hu, M., Li, Z., Yu, J., Wan, X., Tan, H., & Lin, Z. (2023, January). Efficient-Lightweight YOLO: Improving Small Object Detection in YOLO for Aerial Images. *Sensors*, *23*(14), 6423. doi: 10.3390/s23146423

Huang, Z., Li, W., Xia, X.-G., & Tao, R. (2022, February). A General Gaussian Heatmap Label Assignment for Arbitrary-Oriented Object Detection. *IEEE Transactions on Image Processing*, *31*, 1895–1910. doi: 10.1109/TIP.2022.3148874

Hurtut, T., Landes, P.-E., Thollot, J., Gousseau, Y., Drouillhet, R., & Coeurjolly, J.-F. (2009, August). Appearance-guided synthesis of element arrangements by example. In *Proceedings of the 7th International Symposium on Non-Photorealistic Animation and Rendering* (pp. 51–60). New York, USA: Association for Computing Machinery. doi: 10.1145/1572614.1572623

Isola, P., Zhu, J.-Y., Zhou, T., & Efros, A. A. (2017, July). Image-to-Image Translation with Conditional Adversarial Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 5967–5976). doi: 10.1109/CVPR.2017.632

Jeune, P. L., & Mokraoui, A. (2023, July). *Rethinking Intersection Over Union for Small Object Detection in Few-Shot Regime.* arXiv. doi: 10.48550/arXiv.2307.09562

Jocher, G., Chaurasia, A., Stoken, A., Borovec, J., NanoCode012, Kwon, Y., ... Jain, M. (2022, November). *Ultralytics/yolov5: V7.0 - YOLOv5 SOTA realtime instance segmentation.* Zenodo. (software) doi: 10.5281/zenodo.7347926

Jones, C., & Kammen, D. M. (2014, January). Spatial Distribution of U.S. Household Carbon Footprints Reveals Suburbanization Undermines Greenhouse Gas Benefits of Urban Population Density. *Environmental Science & Technology*, *48*(2), 895–902. doi: 10.1021/es4034364

Kingma, D. P., & Welling, M. (2014, April). Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations (ICLR)*. Banff, Canada. doi: 10.48550/arXiv.1312.6114

Kline, D. M., & Berardi, V. L. (2005, December). Revisiting squared-error and cross-entropy functions for training neural network classifiers. *Neural Computing & Applications*, *14*(4), 310–318. doi: 10.1007/s00521-005-0467-y

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012, December). ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in neural information processing systems (NIPS)* (Vol. 25). Lake Tahoe, USA: Curran Associates, Inc. Retrieved from https://proceedings.neurips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html

Kulikova, M., Jermyn, I., Descombes, X., Zhizhina, E., & Zerubia, J. (2011, April). A Marked Point Process Model Including Strong Prior Shape Information Applied to Multiple Object Extraction From Images. *International Journal of Computer Vision and Image Processing*, *1*(2), 1–12. doi: 10.4018/ijcvip.2011040101

Lacoste, C. (2004). *Extraction de réseaux linéiques à partir d'images satellitaires et aériennes par processus ponctuels marqués* (These de Doctorat, Nice). Retrieved from https://www.theses.fr/2004NICE4046

Lacoste, C., Descombes, X., & Zerubia, J. (2005, October). Point processes for unsupervised line network extraction in remote sensing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *27*(10), 1568–1579. doi: 10.1109/TPAMI.2005.206

Lafarge, F., Gimel'farb, G., & Descombes, X. (2010, September). Geometric Feature Extraction by a Multimarked Point Process. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *32*(9), 1597–1609. doi: 10.1109/TPAMI.2009.152

LaLonde, R., Zhang, D., & Shah, M. (2018, June). ClusterNet: Detecting Small Objects in Large Scenes by Exploiting Spatio-Temporal Information. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)* (pp. 4003–4012). Salt Lake City, USA. doi: 10.1109/CVPR.2018.00421

Law, H., & Deng, J. (2018, September). CornerNet: Detecting Objects as Paired Keypoints. In *Proceedings of the European Conference on Computer Vision (ECCV)* (pp. 734–750). Munich, Germany. Retrieved from https://openaccess.thecvf.com/content_ECCV_2018/html/Hei_Law_CornerNet_Detecting_Objects_ECCV_2018_paper.html

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989, December). Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation*, *1*(4), 541–551. doi: 10.1162/neco.1989.1.4.541

LeCun, Y., Chopra, S., Hadsell, R., Ranzato, M., & Huang, F. J. (2006). A Tutorial on Energy-Based Learning. *Predicting structured data*, 59. Retrieved from http://yann.lecun.com/exdb/publis/orig/lecun-06.pdf

Lee, J., Lee, Y., Kim, J., Kosiorek, A. R., Choi, S., & Teh, Y. W. (2019, May). Set Transformer: A Framework for Attention-based Permutation-Invariant Neural Networks. *arXiv:1810.00825 [cs, stat]*. Retrieved 2020-11-13, from http://arxiv.org/abs/1810.00825 (Comment: ICML 2019)

Li, T., Comer, M., & Zerubia, J. (2019, September). Feature Extraction and Tracking of CNN Segmentations for Improved Road Detection from Satellite Imagery. In *IEEE International Conference on Image Processing (ICIP)* (pp. 2641–2645). Taipei, Taiwan. doi: 10.1109/ICIP.2019.8803355

Li, T., Comer, M., & Zerubia, J. (2020, May). An Unsupervised Retinal Vessel Extraction and Segmentation Method Based On a Tube Marked Point Process Model. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 1394–1398). Barcelona, Spain. doi: 10.1109/ICASSP40776.2020.9054023

Li, Z., Hou, B., Wu, Z., Ren, B., Ren, Z., & Jiao, L. (2023, August). Gaussian synthesis for high-precision location in oriented object detection. *IEEE Transactions on Geoscience and Remote Sensing*, 1–1. doi: 10.1109/TGRS.2023.3310619

Lieshout, M.-C. V. (2000). *Markov Point Processes and Their Applications*. London: Imperial College Press. Retrieved from https://books.google.fr/books?id=e_tpDQAAQBAJ

Lin, T.-Y., Dollar, P., Girshick, R., He, K., Hariharan, B., & Belongie, S. (2017, July). Feature Pyramid Networks for Object Detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 936–944). Honolulu, USA: IEEE. doi: 10.1109/CVPR.2017.106

Lin, T.-Y., Goyal, P., Girshick, R., He, K., & Dollar, P. (2017, October). Focal loss for dense object detection. In *Proceedings of the IEEE/CVF international conference on computer vision (ICCV)*. doi: 10.1109/ICCV.2017.324

Llerena, J. M., Zeni, L. F., Kristen, L. N., & Jung, C. (2021, June). *Gaussian Bounding Boxes and Probabilistic Intersection-over-Union for Object Detection.* arXiv. doi: 10.48550/arXiv.2106.06072

Long, J., Shelhamer, E., & Darrell, T. (2015, June). Fully Convolutional Networks for Semantic Segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 3431–3440). Boston, USA. doi: 10.1109/CVPR.2015.7298965

Lu, X., Sun, X., Diao, W., Mao, Y., Li, J., Zhang, Y., … Fu, K. (2023, February). Few-Shot Object Detection in Aerial Imagery Guided by Text-Modal Knowledge. *IEEE Transactions on Geoscience and Remote Sensing*, *61*, 1–19. doi: 10.1109/TGRS.2023.3250448

Macchi, O. (1975, March). The Coincidence Approach to Stochastic Point Processes. *Advances in Applied Probability*, *7*(1), 83–122. doi: 10.2307/1425855

Mayo, P., Anantrasirichai, N., Chalidabhongse, T. H., Palasuwan, D., & Achim, A. (2022, March). *Detection of Parasitic Eggs from Microscopy Images and the emergence of a new dataset* (No. arXiv:2203.02940). arXiv. (Comment: 7 pages, 3 figures, 1 table) doi: 10.48550/arXiv.2203.02940

McKinney, M. L. (2002, October). Urbanization, Biodiversity, and Conservation: The impacts of urbanization on native species are poorly studied, but educating a highly urbanized human population about these impacts can greatly improve species conservation in all ecosystems. *BioScience*, *52*(10), 883–890. doi: 10.1641/0006-3568(2002)052[0883:UBAC]2.0.CO;2

Mei, S., Lian, J., Wang, X., Su, Y., Ma, M., & Chau, L.-P. (2023, June). *A Comprehensive Study on the Robustness of Image Classification and Object Detection in Remote Sensing: Surveying and Benchmarking*. arXiv. doi: 10.48550/arXiv.2306.12111

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., & Teller, E. (1953, June). Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics*, *21*(6), 1087–1092. doi: 10.1063/1.1699114

Moser, G., Serpico, S. B., & Benediktsson, J. A. (2013, March). Land-Cover Mapping by Markov Modeling of Spatial–Contextual Information in Very-High-Resolution Remote Sensing Images. *Proceedings of the IEEE*, *101*(3), 631–651. doi: 10.1109/JPROC.2012.2211551

Mundhenk, T. N., Konjevod, G., Sakla, W. A., & Boakye, K. (2016, October). A Large Contextual Dataset for Classification, Detection and Counting of Cars with Deep Learning. In B. Leibe, J. Matas, N. Sebe, & M. Welling (Eds.), *Computer Vision – ECCV* (pp. 785–800). Amsterdam, The Netherlands: Springer International Publishing. doi: 10.1007/978-3-319-46487-9_48

Neven, D., Brabandere, B. D., Proesmans, M., & Van Gool, L. (2019, June). Instance Segmentation by Jointly Optimizing Spatial Embeddings and Clustering Bandwidth. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)* (pp. 8829–8837). Long Beach, USA: IEEE. doi: 10.1109/CVPR.2019.00904

Ortner, M. (2004). *Processus ponctuels marqués pour l'extraction automatique de caricatures de bâtiments à partir de modèles numériques d'élévation* (These de doctorat, Université Nice Sophia Antipolis). Retrieved 2020-11-13, from https://tel.archives-ouvertes.fr/tel-00189803

Ortner, M., Descombes, X., & Zerubia, J. (2007, October). *An adaptive simulated annealing cooling schedule for object detection in images* (Research Report). INRIA. Retrieved from https://inria.hal.science/inria-00181764

Ortner, M., Descombes, X., & Zerubia, J. (2008, January). A Marked Point Process of Rectangles and Segments for Automatic Analysis of Digital Elevation Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *30*(1), 105–119. doi: 10.1109/TPAMI.2007.1159

Pastorino, M., Moser, G., Serpico, S. B., & Zerubia, J. (2022, January). Semantic Segmentation of Remote-Sensing Images Through Fully Convolutional Neural Networks and Hierarchical

Probabilistic Graphical Models. *IEEE Transactions on Geoscience and Remote Sensing*, *60*, 1–16. doi: 10.1109/TGRS.2022.3141996

Perrin, G., Descombes, X., & Zerubia, J. (2004, September). Tree crown extraction using marked point processes. In *12th European Signal Processing Conference (EUSIPCO)* (pp. 2127–2130). Vienna, Austria. Retrieved from https://ieeexplore.ieee.org/abstract/document/7079695

Pham, M.-T., Gangloff, H., & Lefèvre, S. (2023, July). Weakly supervised marine animal detection from remote sensing images using vector-quantized variational autoencoder. In *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*. Pasadena, USA. doi: 10.48550/arXiv.2307.06720

Pham, T. T., Hamid Rezatofighi, S., Reid, I., & Chin, T.-J. (2016, June). Efficient Point Process Inference for Large-Scale Object Detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 2837–2845). Las Vegas, USA. doi: 10.1109/CVPR.2016.310

Prewitt, J. M. (1970). Object enhancement and extraction. In *Picture processing and psychopictorics* (pp. 75–138). Elsevier Science. Retrieved from https://books.google.fr/books?id=vp-w_pC9JBAC

Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., & Chen, M. (2022, April). *Hierarchical Text-Conditional Image Generation with CLIP Latents*. arXiv. doi: 10.48550/arXiv.2204.06125

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016, June). 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, USA: IEEE. doi: 10.1109/CVPR.2016.91

Ren, S., He, K., Girshick, R., & Sun, J. (2015, December). Faster R-CNN: Towards real-time object detection with region proposal networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems (NIPS)* (Vol. 28). Montréal, Canada: Curran Associates, Inc. Retrieved from https://proceedings.neurips.cc/paper/2015/file/14bfa6bb14875e45bba028a21ed38046-Paper.pdf

Renner, I. W., Elith, J., Baddeley, A., Fithian, W., Hastie, T., Phillips, S. J., … Warton, D. I. (2015, February). Point process models for presence-only analysis. *Methods in Ecology and Evolution*, *6*(4), 366–379. doi: 10.1111/2041-210X.12352

Robert, C. P., & Casella, G. (2004). Diagnosing Convergence. In C. P. Robert & G. Casella (Eds.), *Monte Carlo Statistical Methods* (pp. 459–509). Springer. doi: 10.1007/978-1-4757-4145-2_12

Ronneberger, O., Fischer, P., & Brox, T. (2015, October). U-Net: Convolutional Networks for Biomedical Image Segmentation. In N. Navab, J. Hornegger, W. M. Wells, & A. F. Frangi (Eds.), *Medical Image Computing and Computer-Assisted Intervention (MICCAI)* (pp. 234–241). Munich, Germany. doi: 10/gcgk7j

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, *65*(6), 386–408. doi: 10.1037/h0042519

Rue, H., & Hurn, MA. (1999, September). Bayesian object identification. *Biometrika*, *86*(3), 649–660. doi: 10.1093/biomet/86.3.649

Ruelle, D. (1970, June). Superstable interactions in classical statistical mechanics. *Communications in Mathematical Physics*, *18*(2), 127–159. doi: 10.1007/BF01646091

Schmidt, A., Lafarge, F., Brenner, C., Rottensteiner, F., & Heipke, C. (2017, April). Forest point processes for the automatic extraction of networks in raster data. *ISPRS Journal of Photogrammetry and Remote Sensing*, *126*, 38–55. doi: 10.1016/j.isprsjprs.2017.01.012

Schmitt, M., Ahmadi, S. A., & Hänsch, R. (2021, July). There is No Data Like More Data - Current Status of Machine Learning Datasets in Remote Sensing. In *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)* (pp. 1206–1209). Brussels, Belgium. doi: 10.1109/IGARSS47720.2021.9555129

Shelhamer, E., Long, J., & Darrell, T. (2017, April). Fully Convolutional Networks for Semantic Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *39*(4), 640–651. doi: 10.1109/TPAMI.2016.2572683

Shermeyer, J., Hossler, T., Van Etten, A., Hogan, D., Lewis, R., & Kim, D. (2021, January). RarePlanes: Synthetic Data Takes Flight. In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV)* (pp. 207–217). virtual. doi: 10.1109/WACV48630.2021.00025

Song, Y., & Ermon, S. (2019, December). Generative Modeling by Estimating Gradients of the Data Distribution. In *Advances in Neural Information Processing Systems (NeurIPS)* (Vol. 32). Vancouver, Canada: Curran Associates, Inc. Retrieved from https://proceedings.neurips.cc/paper/2019/hash/3001ef257407d5a371a96dcd947c7d93-Abstract.html

Srivastava, A., Grenander, U., Jensen, G. R., & Miller, M. I. (2002, April). Jump–diffusion Markov processes on orthogonal groups for object pose estimation. *Journal of Statistical Planning and Inference*, *103*(1), 15–37. doi: 10.1016/S0378-3758(01)00195-1

Stoica, R. S., Gregori, P., & Mateu, J. (2005, November). Simulated annealing and object point processes: Tools for analysis of spatial patterns. *Stochastic Processes and their Applications*, *115*(11), 1860–1882. doi: 10.1016/j.spa.2005.06.007

Stoyan, D., Kendall, W., & Mecke, J. (1995). *Stochastic geometry and its applications*. Wiley. Retrieved from https://books.google.fr/books?id=oFDvAAAAMAAJ

Student. (1908). The Probable Error of a Mean. *Biometrika*, *6*(1), 1–25. doi: 10.2307/2331554

Sun, Z., Dai, M., Leng, X., Lei, Y., Xiong, B., Ji, K., & Kuang, G. (2021, July). An Anchor-Free Detection Method for Ship Targets in High-Resolution SAR Images. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, *14*, 7799–7816. doi: 10.1109/JSTARS.2021.3099483

Teh, Y. W., Welling, M., Osindero, S., & Hinton, G. E. (2003, December). Energy-based models for sparse overcomplete representations. *The Journal of Machine Learning Research*, *4*, 1235–1260. Retrieved from https://dl.acm.org/doi/abs/10.5555/945365.964304

Tian, Z., Shen, C., Chen, H., & He, T. (2019, October). FCOS: Fully Convolutional One-Stage Object Detection. In *Proceedings of the IEEE/CVF international conference on computer vision (ICCV)* (pp. 9627–9636). Seoul, Korea. Retrieved from https://openaccess.thecvf.com/content_ICCV_2019/html/Tian_FCOS_Fully_Convolutional_One-Stage_Object_Detection_ICCV_2019_paper.html

Tieleman, T. (2008, July). Training restricted Boltzmann machines using approximations to the likelihood gradient. In *Proceedings of the 25th international conference on Machine learning (ICML)* (pp. 1064–1071). New York, USA: Association for Computing Machinery. doi: 10.1145/1390156.1390290

Tu, Z., & Zhu, S.-C. (2002, May). Image segmentation by data-driven Markov chain Monte Carlo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *24*(5), 657–673. doi: 10.1109/34.1000239

Tupin, F., Inglada, J., & Nicolas, J. (2014). *Remote sensing imagery*. Wiley. Retrieved from https://books.google.fr/books?id=GfcXAwAAQBAJ

Vandame, P., Karam, C., Argentier, L., & Chanussot, J. (2023, August). Réduction de modèles neuronaux profonds par adaptation de l'architecture. In *GRETSI 2023 - XXIXème Colloque Francophone de Traitement du Signal et des Images*. Grenoble, France. Retrieved from https://gretsi.fr/data/colloque/pdf/2023_vandame1120.pdf

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017, December). Attention is all you need. In I. Guyon et al. (Eds.), *Advances in Neural Information Processing Systems (NIPS)* (Vol. 30). Long Beach, USA: Curran Associates, Inc. Retrieved from https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., & Bengio, Y. (2018, February). Graph Attention Networks. In *International Conference on Learning Representations (ICLR)*. Vancouver, Canada. Retrieved from https://openreview.net/forum?id=rJXMpikCZ

Verdié, Y., & Lafarge, F. (2012, October). Efficient Monte Carlo sampler for detecting parametric objects in large scenes. In A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, & C. Schmid (Eds.), *Computer Vision – ECCV* (pp. 539–552). Florence, Italy: Springer Berlin Heidelberg. doi: 10.1007/978-3-642-33712-3_39

Verdié, Y., & Lafarge, F. (2014, January). Detecting parametric objects in large scenes by Monte Carlo sampling. *International Journal of Computer Vision*, *106*(1), 57–75. doi: 10.1007/s11263-013-0641-0

Vincent, L., & Soille, P. (1991, June). Watersheds in digital spaces: An efficient algorithm based on immersion simulations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *13*(6), 583–598. doi: 10.1109/34.87344

Viola, P., & Jones, M. (2001, December). Rapid object detection using a boosted cascade of simple features. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)* (Vol. 1, p. I-I). Kauai, USA. doi: 10.1109/CVPR.2001.990517

Wang, C.-Y., Bochkovskiy, A., & Liao, H.-Y. M. (2023, June). YOLOv7: Trainable Bag-of-Freebies Sets New State-of-the-Art for Real-Time Object Detectors. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)* (pp. 7464–7475). Vancouver, Canada. doi: 10.1109/CVPR52729.2023.00721

Wei, S., Zeng, X., Qu, Q., Wang, M., Su, H., & Shi, J. (2020, June). HRSID: A High-Resolution SAR Images Dataset for Ship Detection and Instance Segmentation. *IEEE Access*, *8*, 120234–120254. doi: 10.1109/ACCESS.2020.3005861

Weir, N., Lindenbaum, D., Bastidas, A., Etten, A. V., McPherson, S., Shermeyer, J., ... Tang, H. (2019, October). SpaceNet MVOI: A Multi-View Overhead Imagery Dataset. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)* (pp. 992–1001). Seoul, Korea. Retrieved from https://openaccess.thecvf.com/content_ICCV_2019/html/Weir_SpaceNet_MVOI_A_Multi-View_Overhead_Imagery_Dataset_ICCV_2019_paper.html

Welling, M., & Teh, Y. W. (2011, June). Bayesian learning via stochastic gradient Langevin dynamics. In *Proceedings of the 28th International Conference on International Conference on Machine Learning (ICML)* (pp. 681–688). Madison, WI, USA: Omnipress. Retrieved from http://people.ee.duke.edu/~lcarin/398_icmlpaper.pdf

Xia, G.-S., Bai, X., Ding, J., Zhu, Z., Belongie, S., Luo, J., ... Zhang, L. (2018, June). DOTA: A Large-Scale Dataset for Object Detection in Aerial Images. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)* (pp. 3974–3983). Salt Lake City, USA. doi: 10.1109/CVPR.2018.00418

Xu, H., Liu, X., Ma, Y., Zhu, Z., Wang, S., Yan, C., & Dai, F. (2023, January). Rotated Object Detection with Circular Gaussian Distribution. *Electronics*, *12*(15), 3265. doi: 10.3390/electronics12153265

Yang, X., & Yan, J. (2022, May). On the Arbitrary-Oriented Object Detection: Classification Based Approaches Revisited. *International Journal of Computer Vision*, *130*(5), 1340–1365. doi: 10.1007/s11263-022-01593-w

Yao, Y., Cheng, G., Wang, G., Li, S., Zhou, P., Xie, X., & Han, J. (2022, December). On Improving Bounding Box Representations for Oriented Object Detection. *IEEE Transactions on Geoscience and Remote Sensing*, *61*, 1–11. doi: 10.1109/TGRS.2022.3231340

Yi, J., Wu, P., Liu, B., Huang, Q., Qu, H., & Metaxas, D. (2021, January). Oriented Object Detection in Aerial Images with Box Boundary-Aware Vectors. In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV)* (pp. 2149–2158). virtual. doi: 10.1109/WACV48630.2021.00220

Yu, D., Guo, H., Zhao, C., Liu, X., Xu, Q., Lin, Y., & Ding, L. (2023, August). An Anchor-Free and Angle-Free Detector for Oriented Object Detection Using Bounding Box Projection.

IEEE Transactions on Geoscience and Remote Sensing, *61*, 1–17. doi: 10.1109/TGRS.2023.3305729

Yu, Q., & Medioni, G. (2009, December). Multiple-Target Tracking by Spatiotemporal Monte Carlo Markov Chain Data Association. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *31*(12), 2196–2210. doi: 10.1109/TPAMI.2008.253

Zeng, Q., Ran, X., Zhu, H., Gao, Y., Qiu, X., & Chen, L. (2023, August). Dynamic Cascade Query Selection for Oriented Object Detection. *IEEE Geoscience and Remote Sensing Letters*, *20*, 1–5. doi: 10.1109/LGRS.2023.3304023

Zhang, J., Lei, J., Xie, W., Fang, Z., Li, Y., & Du, Q. (2023, March). SuperYOLO: Super Resolution Assisted Object Detection in Multimodal Remote Sensing Imagery. *IEEE Transactions on Geoscience and Remote Sensing*, *61*, 1–15. doi: 10.1109/TGRS.2023.3258666

Zhang, J., Lei, J., Xie, W., Li, Y., Yang, G., & Jia, X. (2023, July). Guided Hybrid Quantization for Object Detection in Remote Sensing Imagery via One-to-One Self-Teaching. *IEEE Transactions on Geoscience and Remote Sensing*, *61*, 1–15. doi: 10.1109/TGRS.2023.3293147

Zhang, R., Che, T., Ivanovic, B., Wang, R., Pavone, M., Bengio, Y., & Paull, L. (2022, April). Robust and Controllable Object-Centric Learning through Energy-based Models. In *International Conference on Learning Representations (ICLR)*. virtual. Retrieved 2023-10-09, from https://openreview.net/forum?id=wcNtbEtcGIC

Zhao, T., Liu, N., Celik, T., & Li, H.-C. (2021, June). An Arbitrary-Oriented Object Detector Based on Variant Gaussian Label in Remote Sensing Images. *IEEE Geoscience and Remote Sensing Letters*, 1–5. doi: 10.1109/LGRS.2021.3087492

Zhao, Y., Wang, G., Tang, C., Luo, C., Zeng, W., & Zha, Z.-J. (2021, November). *A Battle of Network Structures: An Empirical Study of CNN, Transformer, and MLP*. arXiv. doi: 10.48550/arXiv.2108.13002

Zheng, K., Dong, Y., Xu, W., Tan, W., & Huang, P. (2023, August). Auto Learner of Objects Co-Occurrence Knowledge for Object Detection in Remote Sensing Images. *IEEE Geoscience and Remote Sensing Letters*, *20*, 1–5. doi: 10.1109/LGRS.2023.3305617

Zheng, Z., Zhong, Y., Wang, J., Ma, A., & Zhang, L. (2023, July). FarSeg++: Foreground-Aware Relation Network for Geospatial Object Segmentation in High Spatial Resolution Remote Sensing Imagery. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1–16. doi: 10.1109/TPAMI.2023.3296757

Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., & Torralba, A. (2016, June). Learning Deep Features for Discriminative Localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 2921–2929). Las Vegas, USA. doi: 10.1109/CVPR.2016.319

Zhou, X., Wang, D., & Krähenbühl, P. (2019, April). *Objects as Points* (No. arXiv:1904.07850). arXiv. doi: 10.48550/arXiv.1904.07850

Zhu, P., Wen, L., Du, D., Bian, X., Fan, H., Hu, Q., & Ling, H. (2022, November). Detection and Tracking Meet Drones Challenge. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *44*(11), 7380–7399. doi: 10.1109/TPAMI.2021.3119563

Zou, Z., Chen, K., Shi, Z., Guo, Y., & Ye, J. (2023, March). Object Detection in 20 Years: A Survey. *Proceedings of the IEEE*, *111*(3), 257–276. doi: 10.1109/JPROC.2023.3238524

Press articles

Bronner, E. (2023, September). Séisme au Maroc : les satellites peuvent aider les secours à réagir au plus vite. *The Conversation*. Retrieved from http://theconversation.com/seisme-au-maroc-les-satellites-peuvent-aider-les-secours-a-reagir-au-plus-vite-183675

Cartes des incendies dans le monde : suivez en temps réel les feux de forêt observés sur Terre. (2023, July). *Franceinfo*. Retrieved from https://www.francetvinfo.fr/faits-divers/incendie/cartes-grece-canada-france-suivez-en-temps-reel-l-etat-des-incendies-dans-le-monde_5944736.html

Gellman, B., & Poitras, L. (2013, August). U.S., British intelligence mining data from nine U.S. Internet companies in broad secret program. *Washington Post*. Retrieved from https://web.archive.org/web/20130824083615/http://articles.washingtonpost.com/2013-06-06/news/39784046_1_prism-nsa-u-s-servers

La détection par intelligence artificielle de piscines non déclarées va être généralisée en France. (2022, August). *Le Monde.fr*. Retrieved from https://www.lemonde.fr/pixels/article/2022/08/29/experimentee-dans-neuf-departements-la-detection-de-piscines-non-declarees-par-intelligence-artificielle-va-etre-generalisee_6139439_4408996.html

OSINT : aux sources d'un nouveau journalisme ? (2023, April). Radio France. Retrieved from https://www.radiofrance.fr/franceculture/podcasts/le-meilleur-des-mondes/osint-aux-sources-d-un-nouveau-journalisme-4663390

Verdon, E., & Nabat, Y. (2023, June). Proposition de loi sur la reconnaissance faciale : Un pas de plus vers la surveillance généralisée ? *The Conversation*. Retrieved from http://theconversation.com/proposition-de-loi-sur-la-reconnaissance-faciale-un-pas-de-plus-vers-la-surveillance-generalisee-207677

Xiao, M., Mozur, P., Qian, I., & Cardia, A. (2022, June). Video: China's Surveillance State Is Growing. These Documents Reveal How. *The New York Times*. Retrieved 2023-10-10, from https://www.nytimes.com/video/world/asia/100000008314175/china-government-surveillance-data.html

Web pages

60th Anniversary of the First GAMBIT-1 Photoreconnaissance Satellite Flight. (2023, July). Retrieved from https://airandspace.si.edu/stories/editorial/60th-anniversary-first-gambit-1

CO3D Constellation. (2023, August). Retrieved from https://www.eoportal.org/satellite-missions/co3d-constellation#performance-specifications

CUDA. (2017, July). Retrieved from https://developer.nvidia.com/cuda-zone

Earth Science Data Systems, NASA. (2015, April). *Fire Information for Resource Management System (FIRMS).* Earth Science Data Systems, NASA. Retrieved from https://www.earthdata.nasa.gov/learn/find-data/near-real-time/firms

European Data Relay Satellite System (EDRS) Overview. (n.d.). Retrieved from https://connectivity.esa.int/european-data-relay-satellite-system-edrs-overview

European sea surface temperature. (2023, June). Retrieved from https://www.eea.europa.eu/ims/european-sea-surface-temperature

IGN. (2020, July). *Les activités de l'IGN en appui des décideurs publics.* Retrieved from https://ign.fr/institut/nos-activites

OrbTrack.org. (2023). Retrieved from https://www.orbtrack.org

Pleiades-HR. (2012, May). Retrieved from https://www.eoportal.org/satellite-missions/pleiades#eop-quick-facts-section

Pléiades Neo. (n.d.). Retrieved from https://earth.esa.int/eogateway/missions/pleiades-neo#instruments-section

Polygon Collision - GPWiki. (2012). Retrieved from http://web.archive.org/web/20141127210836/http://content.gpwiki.org/index.php/Polygon_Collision

PyTorch. (n.d.). Retrieved from https://www.pytorch.org

Torch.nn.functional.grid_sample — PyTorch 2.0 documentation. (n.d.). Retrieved from https://pytorch.org/docs/stable/generated/torch.nn.functional.grid_sample.html

Une carte d'occupation du sol pour faciliter le suivi de l'artificialisation - Institut - IGN. (2023, July). Retrieved from https://ign.fr/institut/une-carte-doccupation-du-sol-pour-faciliter-le-suivi-de-lartificialisation

Other sources

Honoré Daumier. (1862). *Nadar Élevant la Photographie à la Hauteur de l'Art.* Retrieved from https://commons.wikimedia.org/wiki/File:Brooklyn_Museum_-_Nadar_%C3%89levant_la_Photographie_%C3%A0_la_Hauteur_de_l%27Art_-_Honor%C3%A9_Daumier.jpg

Larhmam. (2018, October). *Maximum-margin hyperplane and margin for an SVM trained on two classes. Samples on margins are called support vectors.* Retrieved from https://commons.wikimedia.org/wiki/File:SVM_margin.png

List of Figures

1.1 Nadar Élevant la Photographie à la Hauteur de l'Art . . . 15
1.2 Satellite track . . . 17
1.3 Optical pushbroom . . . 18
1.4 Limitation of the Bayesian approach . . . 22

2.1 Bibliography graph . . . 27
2.2 Viola Jones features . . . 28
2.3 Region with CNN features . . . 29
2.4 Heatmap for center detection . . . 30
2.5 Evolution of Remote Sensing datasets . . . 33
2.6 Diffusion for image generation . . . 37

3.1 Poisson Point Process: homogeneous and non-homogeneous . . . 44
3.2 Strauss Point Process with different λ values . . . 47
3.3 Pairwise marked Point Process . . . 48
3.4 Maximum number of neighbors edge case . . . 51
3.5 Local perturbation examples . . . 56
3.6 Cell independence . . . 57
3.7 Unet architecture . . . 61

4.1 Marked Point Process of rectangles parametrization . . . 69
4.2 Composite Point Process example . . . 71
4.3 Contrast for parametric shapes . . . 72
4.4 Performance of contrast measure on synthetic and real data . . . 73
4.5 Energy map example . . . 74
4.6 Relaxation and discretization of the target map . . . 75
4.7 Filter and response for energy potential . . . 76
4.8 Vector field and divergence . . . 77
4.9 Inferred energy map example . . . 79
4.10 Quality function inflection point . . . 81
4.11 Mark discretization . . . 81
4.12 Energy over marks . . . 82
4.13 Priors illustration . . . 84
4.14 Area and ratio distributions . . . 85
4.15 Combining attractive and repulsive priors . . . 87
4.16 Triplet alignment . . . 88
4.17 Energy model pipeline . . . 90
4.18 Continuity of the energy model . . . 91

5.1 Data-driven translation kernel . . . 96
5.2 Mic sets . . . 100

5.3 Separability of configurations 104
5.4 Contrastive energy learning 107
5.5 Buffer size 109

6.1 Training example with noise prior 120
6.2 Training with non-contrastive samples 122
6.3 Sample detection on COWC 124
6.4 Results on DOTA 127
6.5 Results on DOTA with noise 128
6.6 Results on ADS data 129
6.7 Results on ADS with highlights 130

B.1 Image augmentations 165
B.2 Cutting a large image for inference 166
D.3 Fast overlap computation 168
E.4 Determining cell size 169
E.5 Space partitioning around cells 170
G.6 Attention on graphs 175

List of Tables

6.1 Metrics on COWC 124
6.2 Model energies 125
6.3 Metrics on DOTA 126

1 Cost per operation 176
2 Cost per energy term 177
3 Number of operations 182

List of Definitions

3.1.1 Configuration 43
3.1.2 Point Process 43
3.1.3 Simple Point Process 43
3.1.4 Poisson Point Process 43
3.1.5 Neighborhood 44
3.1.6 Markov Point Process 45
3.1.7 Papangelou intensity 45
3.1.8 Markov marked Point Process 46
3.3.1 Markov chain 51
3.3.2 Homogeneous Markov chain 51
3.3.3 Transition kernel 51
3.3.4 Lipschitz-continuity 58
3.4.1 Convolutional Neural Network 60
3.4.2 Densely connected layer 61
3.4.3 Fully Convolutional Network 61

4.2.1 Heatmap 73
4.2.2 Watershed 74

5.2.1 Energy Based Model 105

List of Algorithms

5.1 Configuration sampling method 102
5.2 Computation of the energy scaling factor 110
5.3 Parameters estimation 111
I.1 Configuration energy computation (Python) 184
I.2 Ratio energy computation (Python) 184
I.3 Position energy computation (Python) 185
I.4 Alignent energy computation (Python) 186
I.5 Point distances computation (Python) 187
I.6 GenericEnergyModel (Python) 188
I.7 Point distances computation (Python) 189

Appendices

A Jacobian for the local transform kernel

In Section 3.3.2.3 we consider a Marked Point Process with $\dim(\mathcal{M})$ marks and denote $d = \dim(\mathcal{M}) + \dim(\mathcal{S})$ the dimension of each point.

For a move from $\mathbf{y}$ to $\mathbf{x}$, with perturbation δ We denote $\mathbf{y} = \{y^1, \dots, y^n\}$ and $\mathbf{x} = \{x^1, \dots, x^n\}$, with $n = n(\mathbf{y}) = n(\mathbf{x})$. We have $x^i = y^i$, $i \neq k$ and $x^k = y^k + \delta$, also $\delta' = \delta$.

Here we illustrate a case where $n \neq k$ for sake of readability, but the reader will easily picture how the result is similar if $n = k$. I_d and O_d respectively denote the identity matrix and zero matrix of size $d \times d$.

First we have the following partial derivatives:

	$i = j \neq k$	$i = j = k$	$i \neq j$
$\frac{\partial x^i}{\partial y^j}$	I_d	I_d	O_d
$\frac{\partial x^i}{\partial \delta}$	O_d	I_d	-
$\frac{\partial \delta'}{\partial y^j}$	O_d	O_d	-
$\frac{\partial \delta'}{\partial \delta}$	I_d		

The Jacobian is then computed as:

$$\left|\frac{\partial(\mathbf{x}, \delta')}{\partial(\mathbf{y}, \delta)}\right| = \begin{vmatrix} \frac{\partial x^1}{\partial y^1} & \dots & \frac{\partial x^1}{\partial y^k} & \dots & \frac{\partial x^1}{\partial y^n} & \frac{\partial x^1}{\partial \delta} \\ \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\ \frac{\partial x^k}{\partial y^1} & \dots & \frac{\partial x^k}{\partial y^k} & \dots & \frac{\partial x^k}{\partial y^n} & \frac{\partial x^k}{\partial \delta} \\ \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\ \frac{\partial x^n}{\partial y^1} & \dots & \frac{\partial x^n}{\partial y^k} & \dots & \frac{\partial x^n}{\partial y^n} & \frac{\partial x^n}{\partial \delta} \\ \frac{\partial \delta'}{\partial y^1} & \dots & \frac{\partial \delta'}{\partial y^k} & \dots & \frac{\partial \delta'}{\partial y^n} & \frac{\partial \delta'}{\partial \delta} \end{vmatrix} \tag{A.1}$$

$$= \begin{vmatrix} I_d & \dots & O_d & \dots & O_d & O_d \\ \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\ O_d & \dots & I_d & \dots & O_d & I_d \\ \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\ O_d & \dots & O_d & \dots & I_d & O_d \\ O_d & \dots & O_d & \dots & O_d & I_d \end{vmatrix} \tag{A.2}$$

$$= \det(I_{nd}) \det(I_d) \tag{A.3}$$

$$= 1 \tag{A.4}$$

B Training a CNN for small object detection in remote sensing data

In this section we present a few methods used to train the CNN models to build the data term regardless of the loss function.

Sampling on objects. When training on large satellite images such as in $DOTA_{0.5}$, the density of objects can vary greatly. As we train the CNN with constant size patches $(128, 128, 3)$, we sample those from the large images of the training dataset. However, sampling those patches uniformly produces many empty patches (no object of interest). Thus, we sample patches $1/3$ of patches uniformly in space, and $2/3$ centered on objects (with a random Gaussian perturbation to avoid consistently centered objects). This allows maintaining a good number of examples of objects, while still avoiding over-detection in empty areas. These patches are resampled in the training dataset every 8 epochs to avoid overfitting.

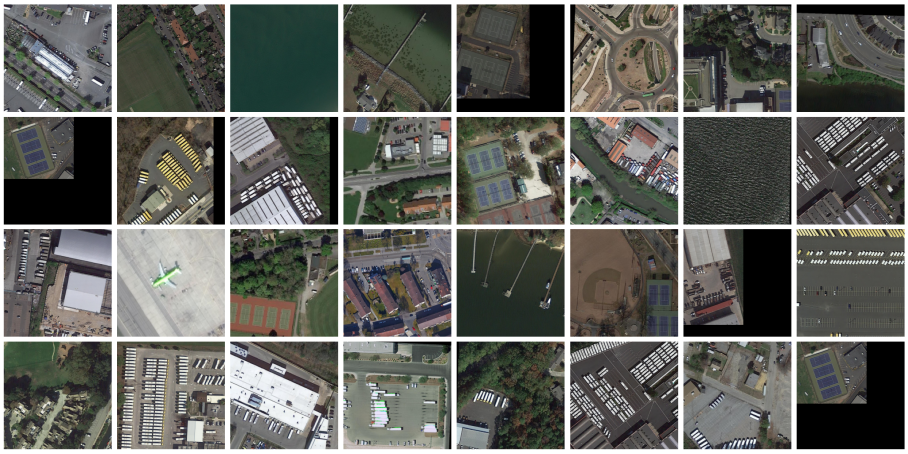
Augmentations. We perform image augmentations on the training patches to add synthetic diversity to the data. We implement two sets of augmentations: a *realistic* one, that produces images that perceptually fit the aspect of images we get from satellite imagery; and a *strong* one that produce images that could not realistically be produced by the sensor (if not for a technical issue). The strong augmented image remains legible though, for the vast majority (except some edge cases), the objects can be still be seen in the image. The augmentations are illustrated in Figure B.1, and composed as follows:

Realistic	**Strong**
rotation (90 degree increments)	*same*
flip	
histogram matching	
RGB shift or CLAHE [1]	
blur	
Gaussian noise	
	random shadow patches or fog
	channel shuffle or dropout of channels
	random brightness and contrast

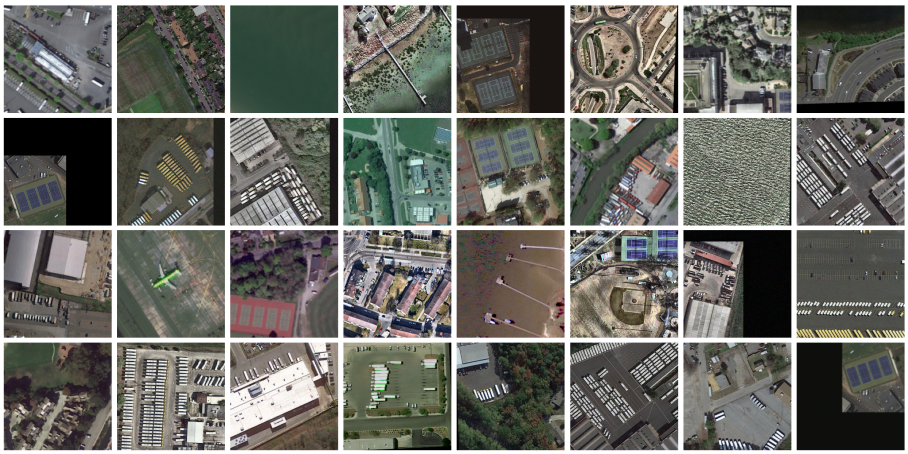
The augmentations are implemented using Albumentations (Buslaev et al., 2020). We find that training the CNN model on the strong augmentation gives better results on the validation dataset. We believe this makes the model more robust thanks to the greater (artificial) variety of data seen at training.

Sampling hard patches. As training progresses, we focus the training on some hard patches. After some number of epochs, we compute the loss (or a part of it) on all the training images; for instance we compute $\mathrm{BCE}(\widehat{\mathbf{H}}_M, \mathbf{H}_M)$ from (4.17) on every image (not patches). We then define a per pixel density from the error by normalizing it. *Difficult* patches are sampled using this density. We find that blurring the density map with a Gaussian blur allows sampling patches with a higher average error. These difficult patches are mixed in the previously described sampling procedure, with $1/3$ difficult patches, and $2/3$ from previous method.

[1] Contrast Limited Adaptive Histogram Equalization



(a) Input patches



(b) *Realistic* augmentations

(c) *Strong* augmentations

Figure B.1: Image augmentation examples

Inferring on large images. The Fully Convolutional aspect of the Unet allows it to be translation invariant. When processing a very large image, we cannot always load the full image into the model at once. We can cut the image into patches, infer on each patch, and stitch those back together. Given the translation invariant property we expect the same results as if we inferred on the whole image at once.

However, one has to consider the padding used within the model. When performing a convolution with a 3×3 kernel, the output will be 2 pixels smaller on both width and height if not padding is applied, as the convolution kernel reaches the edge. To maintain the same output size, we apply mirror padding to add 1 pixel on each side on each dimension to ensure the convolution output is of the same size. This, in turns, introduces some edge artifacts, as the mirror padding tries to mimic data that is not there, thus losing the translation invariance of the result. This padding is however useful when processing image, in order to be able to get results on the borders of the image.

As we can compute the size of this edge effect, we can take it into account when cutting the image into patches, so that we only keep the non-affected center part. We illustrate the cutting scheme in Figure B.2. The size of the edge effect s is approximated by:

$$2s = 2^d + 2^{d+1}l + 8(2^{d-1} - 1) + 4, \tag{B.5}$$

with d the depth of the Unet (number of pooling operations), and l the number of convolution in a bloc ($l = 2$ in the classical implementation of Unet we use). Knowing this edge size, we pad the image with this amount on each side, then split it into patches so that the inner part of the patches that have no edge effect connect together.

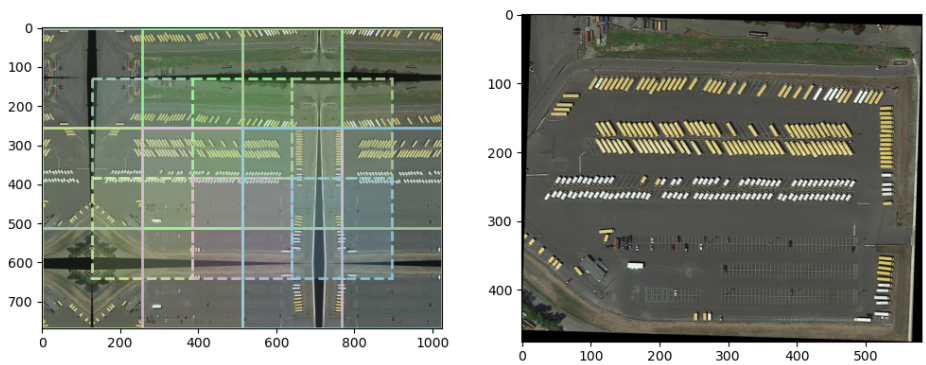


Figure B.2: Cutting a large image for inference. Right, input image; left, patches overlay, plain lines are patches boundaries while dotted lines correspond to the center parch with no edge effect.

C Marks data energy derivation

This Section justifies the formulation of (4.30) in Section 4.2.3. First we consider a simple energy model with no interactions such that:

$$U(\mathbf{y}, \mathbf{X}) = \sum_{y \in \mathbf{y}} V_{pos}(y, \mathbf{X}) + V_\kappa(y, \mathbf{X}) \tag{C.6}$$

For any configuration of points $\mathbf{y}$ we have:

$$\begin{aligned} p(y|\mathbf{y}, \mathbf{X}) &= \frac{p(\{y\} \cup \mathbf{y}|\mathbf{X})}{p(\mathbf{y}|\mathbf{X})} \\ &= \frac{\exp\left(-\sum_{y' \in \{y\} \cup \mathbf{y}} V_{pos}(y', \mathbf{X}) + V_\kappa(y', \mathbf{X})\right)}{\exp\left(-\sum_{y' \in \mathbf{y}} V_{pos}(y', \mathbf{X}) + V_\kappa(y', \mathbf{X})\right)} \\ &= \exp\left(-V_{pos}(y, \mathbf{X}) - V_\kappa(y, \mathbf{X})\right). \end{aligned} \tag{C.7}$$

Here $p(y|\mathbf{y}, \mathbf{X})$ gives the probability of y being a point, given points in the configuration $\mathbf{y}$. Without interactions between points in the model, we have $p(y|\mathbf{y}, \mathbf{X}) = p(y|\mathbf{X})$:

$$\begin{aligned} p(y|\mathbf{X}) &= \frac{1}{Z} \exp\left(-V_{pos}(y\mathbf{X}) - V_\kappa(y, \mathbf{X})\right) \\ p(y_i, y_j|\mathbf{X}) p(y_\kappa|y_i, y_j, \mathbf{X}) &= \frac{1}{Z} \exp\left(-V_{pos}(y, \mathbf{X})\right) \exp\left(-V_\kappa(y, \mathbf{X})\right) \end{aligned} \tag{C.8}$$

From the position energy definition, we identify $p(y_i, y_j|\mathbf{X}) \propto \exp(-V_{pos}(y, \mathbf{X}))$, thus:

$$p(y_\kappa|y_i, y_j, \mathbf{X}) = \frac{1}{Z'} \exp\left(-V_\kappa(y, \mathbf{X})\right). \tag{C.9}$$

Meanwhile, the CNN classifier gives us:

$$\hat{p}(y_\kappa \in c_\kappa|y_i, y_j, \mathbf{X}) = \mathrm{Softmax}(\widehat{\mathbf{Z}}_\kappa^{i,j})_c = \frac{\exp\left(\widehat{\mathbf{Z}}_\kappa^{i,j}[c_\kappa]\right)}{\sum_{c'=1}^{n_\kappa} \exp\left(\widehat{\mathbf{Z}}_\kappa^{i,j}[c'_\kappa]\right)}. \tag{C.10}$$

We approximate probability $p(y_\kappa|y_i, y_j, \mathbf{X})$ over continuous values of y_κ, as its discrete counterpart $\hat{p}(y_\kappa \in \lceil c_\kappa(y_\kappa) \rceil | y_i, y_j, \mathbf{X})$ of y_κ belonging to the $\lceil c_\kappa(y_\kappa) \rceil^{\text{th}}$ bin of values (see (4.28)). Thus we have:

$$V_\kappa(y, \mathbf{X}) = -\ln\left(\hat{p}(y_\kappa \in \lceil c_\kappa(y_\kappa) \rceil | y_i, y_j, \mathbf{X})\right) \tag{C.11}$$

$$V_\kappa(y, \mathbf{X}) = -\widehat{\mathbf{Z}}_\kappa^{i,j}\left[\lceil c_\kappa(y_\kappa) \rceil\right] + \ln\left(\sum_{c=1}^{n_\kappa} \exp\left(\widehat{\mathbf{Z}}_\kappa^{i,j}[c]\right)\right) \tag{C.12}$$

D Fast computation of overlap

We find there is no simple closed-form formula for the area of intersection of two oriented rectangles: $\text{Intersec}(u, y) = \frac{\text{Area}(u \cap y)}{\min\{\text{Area}(u), \text{Area}(y)\}}$. In order to be able to compute overlap values over numerous rectangles without needing to branch the program for each pair of objects, we use a faster approximation to replace the precise overlap value. This makes use of the hyperplane separation theorem applied to convex polygons (*Polygon Collision - GPWiki*, 2012): if two convex polygons are not intersecting there exists a line that passes between them; such a line exists only if one of the sides of one of the polygons forms such a line.

For a given pair of rectangles, we project both into one dimension four times; one for each long and short axis of each of the two rectangles. The 1D overlap can then be easily computed in each projection. The smallest intersection value approximates the intersection of the two objects. While is this not exactly the intersection value $\text{Intersec}(u, y)$ defined in (4.37), this approximation returns 0 only when no overlap is present, and increases (continuously) to 1 when a full overlap happens.

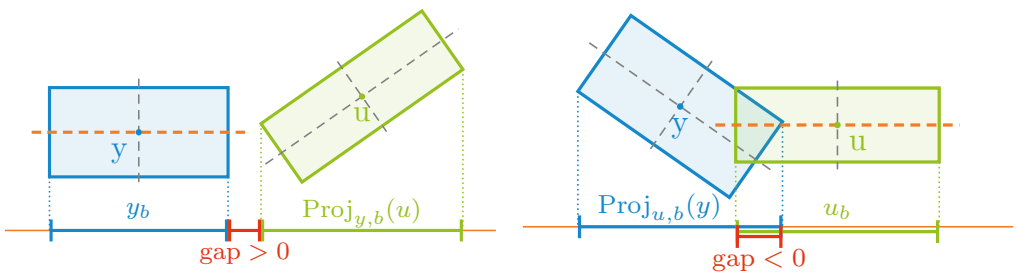
$$\text{Intersec}(u, y) \simeq \max\left\{0, -\frac{1}{c} \max_{\substack{v \in \{u,y\} \\ k \in \{a,b\}}} \left\{\text{Gap}_{v,k}(u, y)\right\}\right\} \tag{D.13}$$

$$c = \min_{\substack{v \in \{u,y\} \\ k \in \{a,b\}}} \left\{\min\left(\text{Proj}_{v,k}(u), \text{Proj}_{v,k}(y)\right)\right\} \tag{D.14}$$

$$\tag{D.15}$$

Where $\text{Proj}_{y,b}(u)$ is the length of the projection of rectangle u on axis b of rectangle y. The computation of $\text{Gap}_{v,k}(u, y)$ and $\text{Proj}_{v,k}$ is illustrated in Figure D.3. On top of approximating the intersection easily, we also get the gap between the two shapes as shown in Figure D.3. For later use we define:

$$\text{Gap}(u, y) = \max_{\substack{v \in \{u,y\} \\ k \in \{a,b\}}} \left\{\text{Gap}_{v,k}(u, y)\right\}. \tag{D.16}$$



(a) no intersection

(b) intersection

Figure D.3: Fast overlap computation illustrated: left has no overlap (gap>0); right has overlap (gap<0). In dashed orange the current projection axis, in gray the other three axes; the final result is the maximum over the four projection axes.

E Parallelism

E.1 Minimum cell size

From Section 3.2.1.1 we know the relation that ensures Markovianity for our model is $\underset{m}{\sim}$ (i.e. a $2d_{max}$ radius). Also, we built all perturbations to only translate objects up to δ_{max}. Moreover, any complex perturbation can be simplified to a sequence of births and deaths (e.g. moving a point is one death and one birth). Thus, any perturbation is a combination of the following:

- A death in cell c, given the Markovianity relation the acceptance rate α depends only on objects less than $2d_{max}$ away from c.
- A birth in cell c or within distance δ_{max} of cell c, thus the acceptance ratio if only function of points within $2d_{max} + \delta_{max}$. Note that other cells of the same mic-set as the current cell might add objects in their δ_{max}-neighborhood.

We illustrate the above in Figure E.4.

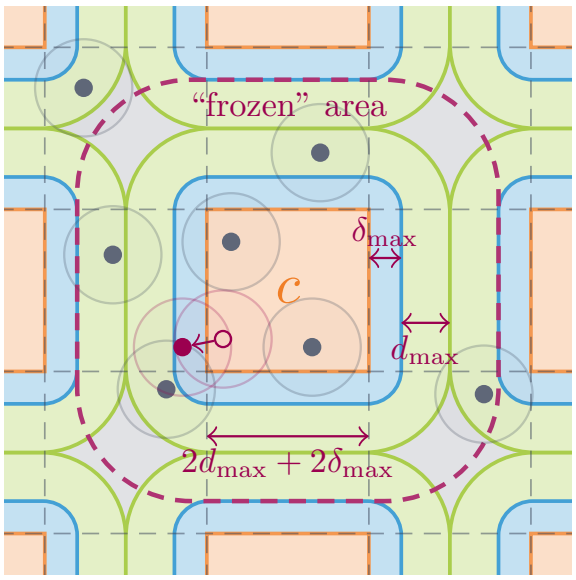


Figure E.4: Determining cell size from basic moves. A point is moved starting from a cell in the current mic-set. The point can be moved in the δ_{max}-neighborhood. This move might change the energy of points in the $(\delta_{max} + d_{max})$-neighborhood. In short, the Markovianity ensures this move depends at most on the dashed area.

E.2 Acceptance ratio for a move in a cell

Here we aim to show that for a move from $\mathbf{y}$ to $\mathbf{y}'$ in cell c we have:

$$\alpha(\mathbf{y}, \mathbf{y}') = \alpha(C_{\bar{c}}(\mathbf{y}), C_{\bar{c}}(\mathbf{y}')), \tag{E.17}$$

This derives from:

$$\Delta U(\mathbf{y} \to \mathbf{y}') = \Delta U(C_{\bar{c}}(\mathbf{y}) \to C_{\bar{c}}(\mathbf{y}')), \tag{E.18}$$

with $\Delta U(\mathbf{y} \to \mathbf{y}') = U(\mathbf{y}', \mathbf{X}, \theta) - U(\mathbf{y}, \mathbf{X}, \theta)$. Any move in cell c can be decomposed into the removal of points from $\mathbf{y}$ in cell c, and the addition of points in the δ_{max}-neighborhood of cell c. Considering the series of configurations induced by this decomposition into N simple additions or removals we have:

$$\Delta U(\mathbf{y} \to \mathbf{y}') = \Delta U(\mathbf{y} \to \mathbf{y}^1 \dots \mathbf{y}^{N-1} \to \mathbf{y}') \tag{E.19}$$

$$= \sum_{k=0}^{N-1} \Delta U(\mathbf{y}^k \to \mathbf{y}^{k+1}), \tag{E.20}$$

with $\mathbf{y} = \mathbf{y}^0$ and $\mathbf{y}' = \mathbf{y}^N$.

From here we only have to show (E.18) for a birth of a single point, as the energy change of a death is minus the energy of the opposed birth move. So we now need to show (E.18) for $\mathbf{y}' = \mathbf{y} \cup \{u\}$.

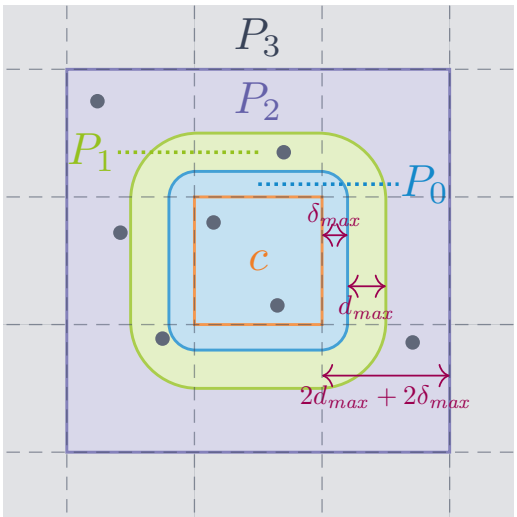


Figure E.5: Space partitioning around cells

We consider the following (non overlapping) partition of $\mathbf{y}$ illustrated in Figure E.5:

$$P_0 = \{y \in Y \,|\, \exists \tilde{y} \in c \times \mathcal{M},\ d(y, \tilde{y}) \leq \delta_{max}\} \tag{E.21}$$

$$P_1 = \{y \in Y \,|\, \exists \tilde{y} \in c \times \mathcal{M},\ d(y, \tilde{y}) \leq d_{max} + \delta_{max}\} \setminus P_0 \tag{E.22}$$

$$P_2 = C_{\bar{c}}(\mathbf{y}) \setminus P_1 \setminus P_0 \tag{E.23}$$

$$P_3 = \mathbf{y} \setminus C_{\bar{c}}(\mathbf{y}) \tag{E.24}$$

Then we have $\mathbf{y} = P_0 \cup P_1 \cup P_2 \cup P_3$. We defined similarly $P'_0, \dots, P'_3$ for $\mathbf{y}'$. Here we suppose $\mathbf{y}' = \mathbf{y} \cup \{u\}$. Since u is placed in cell c or δ_{max} away from c we have $P_1 = P'_1$, $P_2 = P'_2$, $P_3 = P'_3$, but $P_0 \neq P'_0$.

We split the energy difference into our partitions:

$$\Delta U(\mathbf{y} \to \mathbf{y}') = \sum_{y \in Y} V(y|\mathcal{N}_{\{y\}}^{\mathbf{y}'}) - \sum_{y \in \mathbf{y}'} V(y|\mathcal{N}_{\{y\}}^{\mathbf{y}}) \tag{E.25}$$

$$= \sum_{k=0}^{3} \left(\sum_{y \in P_k} V(y|\mathcal{N}_{\{y\}}^{\mathbf{y}'}) - \sum_{y \in P_k'} V(y|\mathcal{N}_{\{y\}}^{\mathbf{y}}) \right) \tag{E.26}$$

- The difference cancels out between P_3 and P_3' as $P_3 = P_3'$ and $\forall y \in P_3, \mathcal{N}_{\{y\}}^{\mathbf{y}} = \mathcal{N}_{\{y\}}^{\mathbf{y}'} \subseteq P_3 \cup P_2$ by construction (i.e. the d_{max}-neighborhood of points in P_3 in unchanged too).
- The same holds for P_2 and P_2', as $P_2 = P_2'$ and $\forall y \in P_2, \mathcal{N}_{\{y\}}^{\mathbf{y}} = \mathcal{N}_{\{y\}}^{\mathbf{y}'} \subseteq P_3 \cup P_2 \cup P_1$.
- For P_1, P_1' however we have $P_1 = P_1'$ but $\forall y \in P_1, \mathcal{N}_{\{y\}}^{\mathbf{y}} \subseteq P_2 \cup P_1 \cup P_0$ and $P_0 \neq P_0'$. Still we can write the following (also valid for P_1' and $C_{\bar{c}}(\mathbf{y}')$):

$$\begin{aligned} \forall y \in P_1, \mathcal{N}_{\{y\}}^{\mathbf{y}} &= \mathcal{N}_{\{y\}}^{P_0 \cup P_1 \cup P_2} \\ &= \mathcal{N}_{\{y\}}^{C_{\bar{c}}(\mathbf{y})} \end{aligned} \tag{E.27}$$

- The above equality is also valid for $y \in P_0$ and $y \in P_0'$

We are now left with what we needed to show (we highlight the changes from (E.26) in red):

$$\Delta U(\mathbf{y} \to \mathbf{y}') = \sum_{k=0}^{1} \left(\sum_{y \in P_k} V(y|\mathcal{N}_{\{y\}}^{C_{\bar{c}}(\mathbf{y}')}) - \sum_{y \in P_k'} V(y|\mathcal{N}_{\{y\}}^{C_{\bar{c}}(\mathbf{y})}) \right) \tag{E.28}$$

$$= U(C_{\bar{c}}(\mathbf{y}'), \mathbf{X}, \theta) - U(C_{\bar{c}}(\mathbf{y}), \mathbf{X}, \theta) \tag{E.29}$$

Remark E.1 – The partition P_2 could be reduced further. Indeed, if we set $P_2 = \{y \in Y | \exists \tilde{y} \in c \times \mathcal{M}, d(y, \tilde{y}) \leq 2d_{max} + \delta_{max}\} \setminus P_1$, the above demonstration still hold, and we can compute the ΔU on a smaller partition of $\mathbf{y}$. In practice, we use the larger P_2, as points are stored in memory by chunks corresponding to their cell, making it easier to simply pull 9 cells from memory than checking for distances of each point against point u.

F Buffer: size and chain length

Notations. One step of the contrastive divergence performs n_s Markov chain iterations (with transition kernel $K_\theta(\cdot,\cdot)$, resulting of parametrized energy model $U(\cdot,\theta)$) starting from a configuration $\mathbf{x}$ resulting in $\mathbf{x}'$. We denote the latter by:

$$\mathbf{x} \xrightarrow{n_s \times K_\theta} \mathbf{x}' \tag{F.30}$$

We also denote $\mathbf{y}_n^-$ the negative sample at step n.

Basic contrastive divergence. Without buffer, at epoch n, the negative samples $\mathbf{y}^-$ is generated as :

$$\mathbf{x}_0 \xrightarrow{n_s \times K_{\theta_n}} \mathbf{y}_n^-, \tag{F.31}$$

with $\mathbf{x}_0 = \mathbf{y}^{GT}$ (Hinton, 2002) or $\mathbf{x}_0 \sim \mathcal{U}(\mathcal{Y})$ (Du & Mordatch, 2019). To increase chain length (and have negative samples of better quality), one has to increase n_s, at a great computational cost. As we alternate between sampling negative configurations and upgrading the parameters, this greatly increases the inference time.

Introducing persistent chains. Here we consider a one-sized buffer (or persistent contrastive divergence (Tieleman, 2008)) as described in Section 5.2.2.2. The resulting procedure for step n is as such:

$$\begin{cases} \mathbf{y}_{n-1} \xrightarrow{n_s \times K_{\theta_n}} \mathbf{y}_n^- & \text{with probability } p_{\mathcal{B}} \\ \mathbf{x}_0 \xrightarrow{n_s \times K_{\theta_n}} \mathbf{y}_n^- & \text{otherwise,} \end{cases} \tag{F.32}$$

with $\mathbf{y}_{n-1}$ pulled from the single element buffer. With this use of the buffer, we virtually run a Markov chain *across* the epochs/steps, for which the transition kernel is updated every n_s steps:

$$\mathbf{x}_0 \xrightarrow{n_s \times K_{\theta_{n-l}}} \mathbf{y}_{n-l}^- \xrightarrow{n_s \times K_{\theta_{n-l+1}}} \dots \xrightarrow{n_s \times K_{\theta_{n-1}}} \mathbf{y}_{n-1}^- \xrightarrow{n_s \times K_{\theta_n}} \mathbf{y}_n^- \tag{F.33}$$

For this cross-epoch chain to converge, the updates from each θ_n to θ_{n+1} has to be small enough so that the stationary density of the Markov chain at $n+1$ is close enough to the density at n. Otherwise, the burn-in period for the chain to stabilize to its new stationary density might be longer than the number of Markov chain iterations n_s. In practice θ_n is updated via small gradient descent steps, and as the energy function U is Lipschitz-continuous w.r.t. to its parameters θ, it guarantees limited variations in the stationary density.

The size of the chain from an initial state $\mathbf{x}_0$ to $\mathbf{y}_n^-$ depends on the random variable L representing the consecutive times the memory was picked in a row after a random initialization. The low of L is simply:

$$p(L = l) = (1 - p_{\mathcal{B}})p_{\mathcal{B}}^l. \tag{F.34}$$

In turn, the chain presented in (F.33) from $\mathbf{x}_0$ to $\mathbf{y}_n^-$, has an expected length given by:

$$\mathbb{E}[Ln_s] = \frac{n_s p_{\mathcal{B}}}{1 - p_{\mathcal{B}}}. \tag{F.35}$$

In practice using $p_{\mathcal{B}} = 0.95$ yields an expected length of $19n_s$. This allows using a lower n_s thus reducing the computational cost, while maintaining qualitative samples from the density derived from $U(\cdot, \mathbf{X}, \theta_n)$.

Increasing the buffer size, inevitably leads to a lower cross-epoch chain length as it introduces the possibility at each step to resume from a shorter cross-epoch length.

G Towards generic (learnable) energy terms

In Chapter 4 we proposed a series of priors with parameters to either set manually of estimate from the data. The selection and design of the energy prior functions to put in the total energy U is by itself a prior on the model, as it restricts the space of energies $U(\ldots,\theta)$ generated by parameters θ. In this section, as a perspective, we propose to push further the parametrization of the energy priors in order to go beyond the function-design-induced constraints, while allowing to build energy function that better fit the data. We first go through a design of per-point priors with MLP, and then show that attention mechanisms can be used to learn a generic aggregation function for interaction priors.

Generic single-point priors. The above-defined priors on single points ($e \in \xi_{prior-point}$), are of the form $V_e(y)$. As the MLP is a universal approximator, a sufficiently generic prior function would be:

$$V_{sgl}(y) = \mathrm{MLP}_{\theta_{sgl}}(y), \tag{G.36}$$

this single point prior could replace all single point priors defined previously (e.g. (4.33) or (4.34)) as the MLP can learn an approximation of those functions from the data.

Generic interaction priors. Similarly, we can define a generic form of prior for interaction energies:

$$V_{int}(y, \mathcal{N}^{\mathbf{y}}_{\{y\}}) = \mathcal{F}_{int}\left\{\mathrm{MLP}_{\theta_{int}}(y,u), u \in \mathcal{N}^{\mathbf{y}}_{\{y\}}\right\}, \tag{G.37}$$

where $\mathrm{MLP}_{\theta_{int}}(y,u)$ gives a potential per interaction, which are aggregated with the operator $\mathcal{F}_{int}$. Here, the choice of the aggregation operator $\mathcal{F}_{int}$ remains. One could set if to a preselected operator $\min$, $\max$ or average. But in the aim to generalize, we instead would need a family of operators parametrized by θ that can approximate these two operators, preferably within a continuous parameter space so that we can estimate the operators by gradient descent.

Attention as aggregation operator. In broad strokes, the operator $\mathcal{F}_{int}$ needs to aggregate a set of point to point interactions into a single value by considering only the relevant elements. This reminds us of attention mechanisms in (Bahdanau et al., 2015) (Vaswani et al., 2017), where the goal is to aggregate the query the relevant relations of a word in a sentence with its neighbors. In that paragraph we will see that the attention mechanism fit our criterion for the design of an interaction energy prior.

Attention is usually defined as:

$$\mathrm{Attention}(\mathbf{Q},\mathbf{K},\mathbf{v}) = \mathrm{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{E}}\right)\mathbf{v}, \tag{G.38}$$

with $\mathbf{v}$ a vector of N values $\mathbf{K}$ the corresponding keys of shape (N,E) and $\mathbf{Q}$ a query matrix of shape (L,E). The result is of size L, the number of query elements. In short, the keys $\mathbf{K}$ represent each element of $\mathbf{v}$ in an embedding space of size E. Each query vector $\mathbf{Q}$ get *compared* to each key in $\mathbf{K}$ in the matrix product $\mathbf{Q}\mathbf{K}^\top$. After normalizing with a Softmax, this result serves to weight each element on $\mathbf{v}$ according to its dot product in the embedding space with the query elements. Since keys $\mathbf{K}$, are supposed to represent each value in $\mathbf{v}$, we can have them generated from the

same input matrix $\mathbf{V}$ of shape (N, F). Matrix $\mathbf{V}$ represents each of the N elements in a feature space of size F. Equation G.38 then becomes:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{v}) = \text{Attention}(\mathbf{q}\mathbf{Q}_w, \mathbf{V}\mathbf{K}_w, \mathbf{V}\mathbf{v}_w), \tag{G.39}$$

with $\mathbf{Q}_w, \mathbf{K}_w, \mathbf{v}_w$ matrices of weights of respective sizes (F, E), (F, E) and $(F, 1)$, and $\mathbf{q}$ a vector of L queries in feature space (shape (L, F)).

In our case, the elements to apply attention on are the neighbors $\mathcal{N}^{\mathbf{y}}_{\{y\}}$ of y ($N = |\mathcal{N}^{\mathbf{y}}_{\{y\}}|$), queried by the current point y ($L = 1$). To transform each point into feature space if size F we propose using an MLP. Eventually we have:

$$V_{att}(y, \mathcal{N}^{\mathbf{y}}_{\{y\}}) = \text{Attention}(\mathbf{q}_y\mathbf{Q}_w, \mathbf{V}_{y,\mathbf{y}}\mathbf{K}_w, \mathbf{V}_{y,\mathbf{y}}\mathbf{v}_w), \tag{G.40}$$

$$\mathbf{V}_{y,\mathbf{y}}[k] = \text{MLP}_{\theta_{att}}(u_k),\ k = 1, \dots, N,\ \{u_1, \dots, u_N\} = \mathcal{N}^{\mathbf{y}}_{\{y\}} \tag{G.41}$$

$$\mathbf{q}_y = \text{MLP}_{\theta_{att}}(y) \tag{G.42}$$

were $\mathbf{Q}_w, \mathbf{K}_w, \mathbf{v}_w$ are parameters of the model (i.e. part of θ).

The above definition fits within the generic formulation defined in (4.35) for the interaction potential v_{att}:

$$v_{att}(y, u_k) = \text{MLP}_{\theta_{att}}(u_k)\mathbf{v}_w. \tag{G.43}$$

Similarly, for the aggregation operator we have:

$$\mathcal{F}_{att}(\mathbf{v}) = \sum_{v \in \mathbf{v}} \omega_v v, \tag{G.44}$$

$$\omega_v = \text{Softmax}\left(\frac{\mathbf{q}_y\mathbf{Q}_w(\mathbf{V}_{y,\mathbf{y}}\mathbf{K}_w)^\top}{\sqrt{E}}\right), \tag{G.45}$$

where the construction of the Softmax operator ensure that have $\sum_{v \in \mathbf{v}} \omega_v = 1$, thus ensuring condition (4.36) is met.

Remark G.1 – A careful reader might have noticed the shift from an unordered set representation to a vector representation in (G.41); it is crucial that the aggregation of interactions be permutation invariant as there is no inherent ordering of elements in the set of neighbors $\mathcal{N}^{\mathbf{y}}_{\{y\}}$. Hopefully the attention as defined above (i.e. without positional encodings) in permutation invariant (Lee et al., 2019).

The greater number of parameters in this proposed energy model combining learned priors and attention makes the parameter estimation procedure harder to tune. We are still tuning the details of that approach, and hope to produce meaningful results soon.

Remark G.2 – From going through the proposed attention on neighbors approach, we notice striking similarities of the resulting model with the graph attention networks proposed in (Veličković et al., 2018). Indeed, the energy computation over a configuration can be seen as a node level inference on a graph; each node of the graph is an object y in configuration $\mathbf{y}$ and the links are the neighborhood relationship. Within a message passing framework, (Veličković et al., 2018) update node $\overrightarrow{h}_i$ into $\overrightarrow{h}'_i$ as such:

$$\overrightarrow{h}'_i = \sigma\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{j \in \mathcal{N}_i} \alpha^k_{i,j}\mathbf{W}^k\overrightarrow{h}'_j\right), \tag{G.46}$$

where $\overrightarrow{h}_i$ is the feature vector of node i, K the number of attention heads [2] *and $\alpha^k_{i,j}$ the weights resulting from the attention head k as illustrated in Figure G.6.*

Considering our current proposed energy model and the similarity it bears with graph models, future works could look into approaching the energy inference through the lens of Graph Neural Networks.

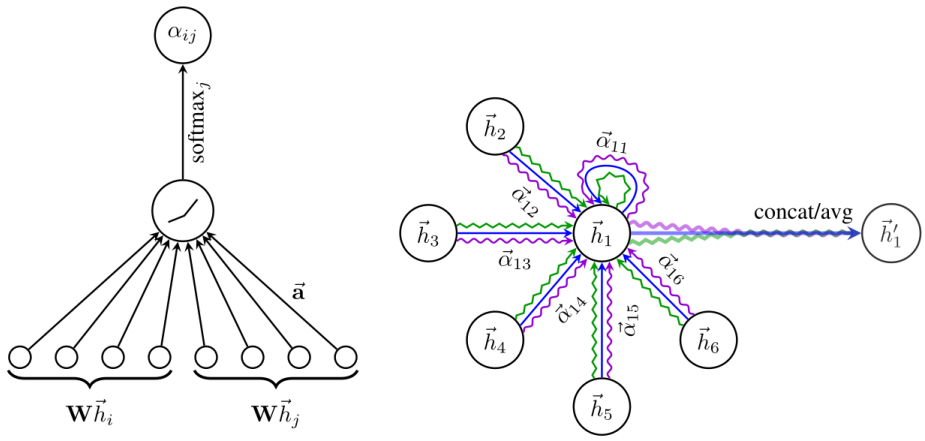


Figure G.6: **Left**: the attention mechanism $a(\mathbf{W}\overrightarrow{h}_i, \mathbf{W}\overrightarrow{h}_j)$ employed by our model, parametrized by a weight vector $\overrightarrow{a} \in \mathbb{R}^{2F'}$, applying a LeakyReLU activation. **Right**: an illustration of multi-head attention (with $K = 3$ heads) by node 1 on its neighborhood. Different arrow styles and colors denote independent attention computations. The aggregated features from each head are concatenated or averaged to obtain $\overrightarrow{h}'_1$. *From (Veličković et al., 2018).*

[2] Instead of one attention mechanism they use multiple *attention heads* at once and average the result.

H Algorithmic complexity

In this appendix we estimate the number of CPU cycles to perform an inference. Here we do not consider the parallelization of operations which can save a lot of computation time.

Energy model.

$$U\left(Y|X,\theta\right) = \sum_{y\in Y}\theta_{w,0} + \sum_{e\in\xi} w_e V_k\left(y|X,\mathcal{N}_\theta^{;}\right) \tag{H.47}$$

Operations costs. We use the values for operations cost in Table 1

Operation	Cost
$a \pm b$	3
ab	5
a/b	25
$\log(a)$	100
a^b	100
$\cos(a)$, $\sin(a)$	100
$a < b$, $a > b$, ...	3
lookup value $a[b]$	3
Bilinear Interpolation (BI)	181
Trilinear Interpolation (TI)	393
Gap (D.16)	73

Table 1: Cycles cost per operation, in number of cycles (approximate).

For $\mathbf{F}(\text{Gap})$ we consider the procedure given in Appendix D: One has to compute the maximum over 4 $\text{Gap}_{v,k}(u,y)$ ($v =\in \{u,y\}$, $k =\in \{a,b\}$). Computing one $\text{Gap}_{v,k}(u,y)$ amounts to projecting the four corners of a rectangle into the axis of the other; i.e. performing a change of reference axis. Once the corners projected, one has to find the minimum and maximum of those four values. With the projection bounds found, it is only a matter to subtract values to find the gap. Considering the heading and orthogonal-heading vectors (long and short axis) of each shape precomputed, the cost amounts to:

$$\mathbf{F}\left(\text{Gap}_{v,k}(u,y)\right) \simeq \underbrace{4\left(2\mathbf{F}(ab) + \mathbf{F}(a+b)\right)}_{\text{projection}} + \underbrace{6\mathbf{F}(a<b)}_{\text{projection min/max}} + F(a+b) \tag{H.48}$$

The pre-computation of the heading ($\overrightarrow{y_t}$) and orthogonal-heading ($\overrightarrow{y_n}$) vectors (for projection) necessitates computing $\cos(y_\alpha)$ and $\sin(y_\alpha)$, thus we approximate a cost of $\mathbf{F}(\overrightarrow{y_t},\overrightarrow{y_n}) \simeq 200$ for one shape. In turn the total cost of Gap becomes:

$$\mathbf{F}\left(\text{Gap}\right) \simeq 4\mathbf{F}\left(\text{Gap}_{v,k}(u,y)\right) + \underbrace{3\mathbf{F}(a,b)}_{\text{max Gap}_{v,k}} = 73 \tag{H.49}$$

The vectors $\overrightarrow{y_t}$ and $\overrightarrow{y_n}$ precomputed (to avoid computing those fore each pair of shapes), thus accounted for in a separate line in Table 2

Notations.

- n_κ: number of classes for mark κ
- $N_y = |\mathcal{N}^{\mathbf{y}}_{\{y\}}|$: number of neighbors for y in $\mathbf{y}$
- d_c: size of cell (see Section 5.1.2)

We assume the distance matrix is precomputed, we only have to update parts of it when new points are added, or when points are updated.

H.1 Cost per energy term

We give the costs of each energy term (and the precomputing of $\overrightarrow{y_t}, \overrightarrow{y_n}$) in Table 2. For interaction energies, we only specify the cost of computing on one interaction (not computing the reduction operator yet). With symbol $\uparrow$ we signify we use the results from the previous computation thus only counting it once. Interpolation (trilinear or bilinear), requires several lookups to interpolate (4 for bilinear, 8 for trilinear).

Term	Eq.	$\pm$	ab	a/b	$\log(a)$	a^b	$\cos(a)$	$a < b$	$a[b]$	BI	TI	Gap	$\mathbf{F}(\cdot)$
Pre-computation													
$\overrightarrow{y_t}, \overrightarrow{y_n}$	—						2						200
Local energies													
V_{pos}	4.26								4	1			193
V_κ	4.30								8		1		417
V_{jntAR}	4.34	3	6	2		1							189
V_{zrNbr}	4.41							1					3
Interaction energies													
v_{ovrlp}	4.37											1	73
v_{align}	4.38	2				1		1					109
v_{repls}	4.39	2		1				1				$\uparrow$	34
v_{attrc}	4.40	2		1				1				$\uparrow$	34

Table 2: cycles costs for energy terms $V_k(y|X, \mathcal{N}^{\cdot}_\theta)$

We decompose the cost of computing $V(y)$ into the cost of computing the local energies $V_e(y)$, the cost of computing the $N_y = |\mathcal{N}^{\mathbf{y}}_{\{y\}}|$ interactions $v_e(y, u), u \in \mathcal{N}^{\mathbf{y}}_{\{y\}}$, and lastly the cost of aggregating those interactions with $\mathcal{F}_e$:

$$\mathbf{F}(V(y)) = \mathbf{F}(V_e) + N_y\mathbf{F}(v_e) + \mathbf{F}(\mathcal{F}_e) + |\xi|(\mathbf{F}(ab) + \mathbf{F}(a+b)) \tag{H.50}$$

$$\mathbf{F}(V_e) = \sum_{e \in \{pos, \kappa, jntAR, zrNbr\}} m_e\mathbf{F}(V_e) \simeq 2025 \tag{H.51}$$

$$\mathbf{F}(v_e) = \sum_{e \in \{ovrlp, align, repls, attrc\}} m_e\mathbf{F}(v_e) \simeq 359 \tag{H.52}$$

$$\mathbf{F}(\mathcal{F}_e) = \sum_{e \in \{ovrlp, align, repls, attrc\}} m_e\mathbf{F}(\mathcal{F}_e) \simeq 12(N_y - 1) \tag{H.53}$$

with m_e the number of instances of each energy, for all terms $m_e = 1$ except for $m_{align} = 2$ and $m_\kappa = 3$. Last equation results from all current aggregation functions being a $\min$ or $\max$ thus $\mathbf{F}(\mathcal{F}_e) = (N_y - 1)\mathbf{F}(a < b)$. Cost $|\xi|(\mathbf{F}(ab) + \mathbf{F}(a+b)) = 96$ corresponds to weighting the energy terms and summing them.

H.2 Cost per kernel

H.2.1 Birth kernel

The birth kernel application is split into tree phases : sampling a new point (K_{sample}), adding the point to the configuration (K_{add}), computing the Green ratio (r) to accept or reject the addition:

$$\mathbf{F}(K_B) = \mathbf{F}(Q_{B,sample}) + \mathbf{F}(Q_{B,add}) + \mathbf{F}(r). \tag{H.54}$$

Sampling. To sample a new point, we pick one new point in the cell. The new point is picked in cell c (of size d_c) given a density map of size d_c^2, the marks are sampled with densities of size n_κ each (3 marks). We approximate the cost of sampling an index in an array of size n to be around $\ln n\mathbf{F}(a,b)$: we draw a random number with $\mathcal{U}([0,1])$ (complexity $\mathcal{O}(1)$) then look in the array representing the cumulative distribution (that can be seen as a sorted array) where the random number fits ($\mathcal{O}(\ln n)$ comparisons).

Considering the cost to sample an array of size n to be n comparisons (3 cycles):

$$\begin{aligned}\mathbf{F}(Q_{B,sample}) &\simeq \mathbf{F}(a<b)\,(2\ln(d_c) + 3\ln(n_\kappa)) \\ &\simeq 6\ln(d_c) + 9\ln(n_\kappa).\end{aligned} \tag{H.55}$$

Adding the point to the configuration. Adding a new point requires computing its distance with the possible neighbors. Computing the distance itself is $\mathbf{F}(dist(y,u)) = \mathbf{F}((y_i - u_i)^2 + (y_j - u_j)^2) \simeq 16$. We also compute the difference of each mark $u_\kappa - y_\kappa$; the total cost of computing the distance and marks difference is $\mathbf{F}(dist, diff) \simeq 25$ When inserting a point we need to compute the distance with all points in $\bar{c}$, i.e. $|C_{\bar{c}}(\mathbf{y})|$ elements:

$$\mathbf{F}(Q_{B,add}) \simeq 25|C_{\bar{c}}(\mathbf{y})|. \tag{H.56}$$

Computing the Green ratio. Computing the Green ratio for the addition of one point y requires computing the energy change induced by the addition of that point.

$$\mathbf{F}(\Delta U(\mathbf{y} \to \mathbf{y} \cup \{y\})) = \underbrace{\mathbf{F}(V_e) + N_y\mathbf{F}(v_e) + \mathbf{F}(\mathcal{F}_e) + 96}_{\text{local and interaction energies of the added point}} + \underbrace{\sum_{u \in \mathcal{N}^{\mathbf{y}}_{\{y\}}} \mathbf{F}(Update(u))}_{\text{update interaction energy of existing points}} \tag{H.57}$$

$$\mathbf{F}(Update(u)) = 96 + \underbrace{\sum_{e \in \{ovrlp, align, repls, attrc\}} m_e\mathbf{F}(a<b)}_{\text{check if new interaction updates max or min}} \simeq 111 \tag{H.58}$$

Thus:

$$\mathbf{F}(\Delta U(\mathbf{y} \to \mathbf{y} \cup \{y\})) \simeq 482N_y + 2109. \tag{H.59}$$

Birth kernel overall cost. It results the following overall cost:

$$\boxed{\mathbf{F}(K_B) \simeq 6\ln(d_c) + 9\ln(n_\kappa) + 25|C_{\bar{c}}(\mathbf{y})| + 482N_y + 2109.} \tag{H.60}$$

with $N_y = |\mathcal{N}^{\mathbf{y}}_{\{y\}}|$.

H.2.2 Death kernel

$$\mathbf{F}(K_D) = \mathbf{F}(Q_{d,sample}) + \mathbf{F}(Q_{D,remove}) + \mathbf{F}(r). \tag{H.61}$$

Sampling. The point to delete is sampled uniformly, we thus approximate:

$$\mathbf{F}(D_{sample}) \simeq 1, \tag{H.62}$$

as we simply sample of an integer uniformly in range $[\![1, |C_c(\mathbf{y})|]\!]$

Removing the point to the configuration. Removing the point does not require any new computation, we simply set the cost to:

$$\mathbf{F}(D_{remove}) \simeq 1. \tag{H.63}$$

Computing the Green ratio. The energy change of the removal $\mathbf{y} \to \mathbf{y} \setminus \{y\}$ is the opposite of the energy of the addition $\mathbf{y} \setminus \{y\} \to \mathbf{y}$. Thus:

$$\mathbf{F}(\Delta U(\mathbf{y} \to \mathbf{y} \setminus \{y\})) = \mathbf{F}(-\Delta U(\mathbf{y} \setminus \{y\} \to \mathbf{y})) \tag{H.64}$$

$$\simeq 482(N_y - 1) + 2109 \tag{H.65}$$

Death kernel overall cost. It results the following overall cost:

$$\boxed{\mathbf{F}(K_B) \simeq 482N_y + 1629,} \tag{H.66}$$

with $N_y = |\mathcal{N}^{\mathbf{y}}_{\{y\}}|$.

H.2.3 Diffusion

To perform Diffusion on the configuration we have to compute $\frac{\partial U(\mathbf{y})}{\partial u}$ for $u \in \mathbf{y}$. In our current code we compute directly $\frac{\partial U(\mathbf{y})}{\partial u}$ to facilitate implementation. However, we can simplify the whole computation as follows. Prior to that, we define two non-overlapping sets of energy terms such that $\xi_{loc.} \cup \xi_{inter.} = \xi$:

- *Local* energy terms $\xi_{loc.}$, such as V_e, $e \in \xi_{loc.}$ can be written as a function of y, $\mathbf{X}$ and θ only; such as $V_{pos}, V_\kappa, V_{zrNbr}$...
- *Interaction* energy terms $\xi_{inter.}$, such as V_e, $e \in \xi_{inter.}$ can be written as a function of y, $\mathcal{N}^{\mathbf{y}}_{\{y\}}$, $\mathbf{X}$ and θ only; such as V_{align}, V_{ovrlp}...

$$\frac{\partial U(\mathbf{y})}{\partial u} = \sum_{y \in \mathbf{y}} \frac{\partial V\left(y, \mathbf{X}, \mathcal{N}^{\mathbf{y}}_{\{y\}}\right)}{\partial u} \tag{H.67}$$

$$= \sum_{y \in \mathbf{y}} \sum_{e \in \xi} w_e \frac{\partial V_e\left(y, \mathbf{X}, \mathcal{N}^{\mathbf{y}}_{\{y\}}\right)}{\partial u} \tag{H.68}$$

$$= \sum_{e \in \xi} w_e \frac{\partial V_e\left(u, \mathbf{X}, \mathcal{N}^{\mathbf{y}}_{\{u\}}\right)}{\partial u} + \sum_{y \in \mathcal{N}^{\mathbf{y}}_{\{u\}}} \sum_{e \in \xi} w_e \frac{\partial V_e\left(y, \mathbf{X}, \mathcal{N}^{\mathbf{y}}_{\{y\}}\right)}{\partial u} \tag{H.69}$$

$$= \sum_{e \in \xi} w_e \frac{\partial V_e\left(u, \mathbf{X}, \mathcal{N}^{\mathbf{y}}_{\{u\}}\right)}{\partial u} + \sum_{y \in \mathcal{N}^{\mathbf{y}}_{\{u\}}} \sum_{e \in \xi_{inter.}} w_e \frac{\partial V_e\left(y, \mathbf{X}, \mathcal{N}^{\mathbf{y}}_{\{y\}}\right)}{\partial u} \tag{H.70}$$

$$= \sum_{e \in \xi_{loc.}} w_e \frac{\partial V_e\left(u, \mathbf{X}\right)}{\partial u} + \sum_{e \in \xi_{inter.}} \sum_{y \in \{u\} \cup \mathcal{N}^{\mathbf{y}}_{\{u\}}} w_e \frac{\partial V_e\left(y, \mathbf{X}, \mathcal{N}^{\mathbf{y}}_{\{y\}}\right)}{\partial u} \tag{H.71}$$

To pass from H.68 to H.69, we note that $V_e(y, \mathbf{X}, \mathcal{N}^{\mathbf{y}}_{\{y\}})$ does not depend on u if $y \neq u$ and $y \notin \mathcal{N}^{\mathbf{y}}_{\{u\}}$. From H.69 to H.70 we note that for $e \in \xi_{loc.}$, $V_e(y, \mathbf{X})$ does not depend on u if $y \neq u$.

To further simplify the computation we look into the interaction energies in the right-hand part of (H.71). First we specify the form of $\mathcal{F}_e$:

$$\mathcal{F}_e(v_1, \dots, v_N) = \sum_{k=1}^{N} \omega_k(v_1, \dots, v_N) v_k. \tag{H.72}$$

This form remains quite general as it encompasses the following aggregation operations (amongst many others):

- min: with $\omega_k(v_1, \dots, v_N) = \mathbb{1}_{k = \underset{k'=1,\dots,N}{\arg\min}\, v_k}(k)$
- max: with $\omega_k(v_1, \dots, v_N) = \mathbb{1}_{k = \underset{k'=1,\dots,N}{\arg\max}\, v_k}(k)$
- Average: with $\omega_k(v_1, \dots, v_N) = \frac{1}{N}$
- Attention mechanisms as described in Appendix I.2

Moreover we consider symmetrical interaction energies such that $v_e(y, u) = v_e(u, y)$.

The right-hand part of (H.71) can be then derived as:

$$\frac{\partial V_e\left(y, \mathbf{X}, \mathcal{N}^{\mathbf{y}}_{\{y\}}\right)}{\partial u} = \sum_{k=1}^{N_y} \frac{\partial \omega_k(v_e(y, y'_1), \dots, v_e(y, y'_{N_y})) v_e(y, y'_k)}{\partial u} \tag{H.73}$$

$$= \sum_{k=1}^{N_y} \frac{\partial \omega_k(\dots)}{\partial u} v_e(y, y'_k) + \frac{\partial v_e(y, y'_k)}{\partial u} \omega_k(\dots) \tag{H.74}$$

with $\mathcal{N}^{\mathbf{y}}_{\{y\}} = \{y'_1, \dots, y'_{N_y}\}$, denoting the number of neighbors $N_y = |\mathcal{N}^{\mathbf{y}}_{\{y\}}|$ for simplicity. If we consider $\mathcal{F}_e$ to be a $\min$ (or $\max$), as our current model is done, the computation simplifies to:

$$\frac{\partial V_e\left(y, \mathbf{X}, \mathcal{N}^{\mathbf{y}}_{\{y\}}\right)}{\partial u} = \sum_{k=1}^{N_y} \mathbb{1}_{k = \underset{l=1,\dots,N_y}{\arg\min}\, v_e(y, y'_l)}(k) \frac{\partial v_e(y, y'_k)}{\partial u} \tag{H.75}$$

$$= \frac{\partial v_e(y, y'_{k_y})}{\partial u}, \quad \text{with } k_y = \underset{l=1,\dots,N_y}{\arg\min}\, v_e(y, y'_l). \tag{H.76}$$

Equation (H.71) then becomes:

$$\frac{\partial U(\mathbf{y})}{\partial u} = \sum_{e \in \xi_{loc.}} w_e \frac{\partial V_e\left(u, \mathbf{X}\right)}{\partial u} + \sum_{e \in \xi_{inter.}} \sum_{y \in \{u\} \cup \mathcal{N}^{\mathbf{y}}_{\{u\}}} w_e \frac{\partial v_e(y, y'_{k_y})}{\partial u} \tag{H.77}$$

With this we can approximate the complexity of computing $\frac{\partial U(\mathbf{y})}{\partial u}$. We approximate $\mathbf{F}(\frac{\partial V_e(u,\mathbf{X})}{\partial u}) \simeq \mathbf{F}(V_e(u, \mathbf{X}))$ and $\mathbf{F}(\frac{\partial v_e(y,y')}{\partial u}) \simeq \mathbf{F}(v_e(y, y'))$:

$$\begin{aligned} \mathbf{F}\left(\frac{\partial U(\mathbf{y})}{\partial u}\right) \simeq & |\xi_{loc.}|(\mathbf{F}(a+b) + \mathbf{F}(ab)) + \sum_{e \in \xi_{loc.}} \mathbf{F}(V_e(u, \mathbf{X})) \\ & + (N_y + 1)\left(|\xi_{inter.}|(\mathbf{F}(a+b) + \mathbf{F}(ab)) + \sum_{e \in v} \mathbf{F}(v_e(y, y')\right). \end{aligned} \tag{H.78}$$

Using values in table 2 and (H.51), (H.52):

$$\mathbf{F}\left(\frac{\partial U(\mathbf{y})}{\partial u}\right) \simeq 399 N_y + 2480. \tag{H.79}$$

To compute diffusion over the whole cell:

$$\mathbf{F}\left(\frac{\partial U(\mathbf{y})}{\partial C_c\left(\mathbf{y}\right)}\right) \simeq |C_c\left(\mathbf{y}\right)|(399 N_y + 2480). \tag{H.80}$$

To compute the diffusion update, we sample $|C_c\left(\mathbf{y}\right)|$ Gaussian distributed values. We grossly approximate this sampling to 100 (this is negligible compared to the cost with a $N_y|C_c\left(\mathbf{y}\right)|$ factor):

$$\boxed{\mathbf{F}(\mathit{Diffusion}_c) \simeq 2580|C_c\left(\mathbf{y}\right)| + 399 N_y|C_c\left(\mathbf{y}\right)|. \qquad \text{(H.81)}}$$

Alternatively the diffusion applied to a single element $y \in \mathbf{y}$ costs:

$$\boxed{\mathbf{F}(\mathit{Diffusion}_y) \simeq 2580 + 399 N_y. \qquad \text{(H.82)}}$$

H.3 Total cost

We can now compute the cost of one Markov chain step K_c in cell c. In practice, we have $d_{max} = 16$ and $\delta_{max} = 8$ thus $d_c = 48$ and $n_\kappa = 32$. From these and the object density λ we can derive approximate values for the parameters in the equations above:

$$N_y \simeq \lambda \pi d_{max}^2 \tag{H.83}$$

$$|C_c(\mathbf{y})| \simeq \lambda d_c^2 \tag{H.84}$$

$$|C_{\bar{c}}(\mathbf{y})| \simeq 9\lambda d_c^2 \tag{H.85}$$

$$N_y |C_c(\mathbf{y})| \simeq \lambda^2 \pi d_{max}^2 d_c^2 \tag{H.86}$$

We get:

$$\boxed{\begin{aligned}\mathbf{F}(K_c) &= 0.8\mathbf{F}(\mathit{Diffusion}_c) + 0.1\mathbf{F}(K_B) + 0.1\mathbf{F}(K_D) \\ &\simeq 3.8 \times 10^2 + 4.9 \times 10^6 \lambda + 5.9 \times 10^8 \lambda^2.\end{aligned}} \tag{H.87}$$

To get the operations per pixel (ops/px/iter) we compute $\mathbf{F}(K_c)/d_c^2$. In practice, we use $n_s = 77000$ iterations of the Markov chain, so the total number of operations (ops/px) is $n_s\mathbf{F}(K_c)/d_c^2$. In the testing data we observe a variety of object densities, we report complexity values for the minimum and maximum, average and 95th percentile ($q95$) and report those values in Table 3 (a). For these density values, we compute the cost per pixel with diffusion on whole cells, on single object and without diffusion in Table 3 (b).

(a) Density values

λ	value
λ_{min}	1.4×10^-5
λ_{avg}	7.9×10^-4
λ_{q95}	1.9×10^-3
λ_{max}	5.2×10^-3

(b) Cost per pixel

λ	ops/px/iter	ops/px
With diffusion on c		
λ_{min}	0.2	1.5×10^4
λ_{avg}	2.0	1.5×10^5
λ_{q95}	5.2	4.0×10^5
λ_{max}	18.3	1.4×10^6
With diffusion on y		
λ_{min}	1.1	8.2×10^4
λ_{avg}	1.2	9.2×10^4
λ_{q95}	1.4	1.1×10^5
λ_{max}	1.9	1.5×10^5
Without diffusion		
λ_{min}	0.8	6.3×10^4
λ_{avg}	1.0	8.0×10^4
λ_{q95}	1.4	1.0×10^5
λ_{max}	2.3	1.8×10^5

Table 3: Number of operations depending on object density.

I Implementation: data structures and parallelism

We define all the energy terms in terms of tensor computation with an extra batch dimension so that we can compute energies over multiple cells at once, and leave the tensor computation library deal with the parallelization for us.

Example I.1 – Consider a simple sum operation: given a tensor $\mathbf{a}$ *of size* (N)*, we compute the sum to return a single scalar* $\sum_{i=1}^{N} a[i]$*. If we want to perform this operation over a batch of* B *tensors* $\mathbf{a}_1, \ldots, \mathbf{a}_B$*, we concatenate those into* $\mathbf{A}$ *of shape* (B, N) *such that* $\mathbf{A}[j] = \mathbf{a}_j$*, now the sum operation over the batched tensor is such that the resulting tensor* $\mathbf{R}$ *of size* (B) *is* $\mathbf{R}[j] = \sum_i A[j, i]$*. The sum operation is parallelized across the batch dimension transparently, and results for each input element* $\mathbf{a}_j$ *is stored in* $\mathbf{R}[j]$*.*

I.1 Batch computation of cells

When performing the RJMCMC in parallel, we have to compute the acceptance ratio for a set $\tilde{s}$ of B cells in parallel. We have the following property:

$$\alpha(\mathbf{y}, \mathbf{y}') = \alpha(C_{\bar{c}}(\mathbf{y}), C_{\bar{c}}(\mathbf{y}')).$$

Denoting the point dimension $D = \dim(\mathcal{S}) + \dim(\mathcal{M})$, we can represent the current cells $\tilde{s}$ content as a tensor $\mathbf{Y}$ of shape (B, N, D), where N is the maximum number of points in any cells and its neighbors:

$$N = \max_{c \in \tilde{s}} n(C_{\bar{c}}(\mathbf{y})).$$

Each element $\mathbf{Y}[k, l]$ is a D-dimensional vector, where each value represents (in that order) the two coordinates, width, length and angle of $y_{\bar{c}_k, l}$.

For instance, considering a set of cells $c_1, \ldots, c_4$ with $y_{\bar{c}_1,1}, y_{\bar{c}_1,2}, \ldots$ the content of $\bar{c}_1$ and similarly for the others. Denoting by $\varnothing$ the absence of point (since all $\bar{c}$ do not have the same number of points necessarily), the following shows an example of $\mathbf{Y}$ for $B = 4$, $N = 5$, $D = 5$ (if we consider rectangles, each y is a vector of size $D = 5$):

$$\mathbf{Y} = \begin{bmatrix} y_{\bar{c}_1,1} & y_{\bar{c}_1,2} & y_{\bar{c}_1,3} & \varnothing & \varnothing \\ y_{\bar{c}_2,1} & y_{\bar{c}_2,2} & y_{\bar{c}_2,3} & y_{\bar{c}_2,4} & \varnothing \\ y_{\bar{c}_3,1} & y_{\bar{c}_3,2} & \varnothing & \varnothing & \varnothing \\ y_{\bar{c}_4,1} & y_{\bar{c}_4,2} & y_{\bar{c}_4,3} & y_{\bar{c}_4,4} & y_{\bar{c}_4,5} \end{bmatrix}$$

We complement $\mathbf{Y}$ with a binary mask $\mathbf{M_Y}$ to encode actual values against undefined, as in practice the $\varnothing$ contain some filler values. The mask is of size (B, N). For our example:

$$\mathbf{M_Y} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Total energy. Denoting $\mathcal{V}(\cdot)$ the equivalent in batched tensor operations of per point energy V (see (4.3)) we get:

$$V\left(y_{\bar{c}_k,l}, \mathcal{N}^{C_{\bar{c}_k}(\mathbf{y})}_{\{y_{\bar{c}_k,l}\}}, \mathbf{X}\right) = \left(\mathbf{M_Y} \odot \mathcal{V}(\mathbf{Y})\right)[k,l]$$

$$U(C_{\bar{c}_k}(\mathbf{y}), \mathbf{X}) = \left(\sum_{l=1}^{N} \left(\mathbf{M_Y} \odot \mathcal{V}(\mathbf{Y})\right)[l]\right)[k],$$

with $A \odot B$ the element wise multiplication of matrices A and B. In Python, we get Algorithm I.1.

```
from torch import Tensor
def configuration_energy(Y:Tensor, M:Tensor) -> Tensor:
    """
    :param Y: batched configuration Tensor of shape (B,N,D)
    :param M: batched mask Tensor of shape (B,N)
    :return: configuration energy Tensor of shape (B)
    """
    U = torch.sum(M*V(Y), dim=1)
    return U
```

Algorithm I.1: Computation of the energy over batched cells.

Point priors. For point-wise priors such as V_{ratio} defined in (4.33), tensor operation $\mathcal{V}_{ratio}$ is defined as a per-point operation (only acts on last dimension D of shape (B, N, D)). We show the resulting simplified code in Algorithm I.2.

```
from torch import Tensor, nn
from torch.nn import Module

class RatioEnergy(Module):
    def __init__(self, mu:float, sigma:float):
        super(RatioEnergy,self).__init__()
        self.mu = nn.Parameter(torch.tensor(mu)) # initialize parameters
        self.sigma = nn.Parameter(torch.tensor(sigma))

    def q_fun(self,ratios:Tensor) -> Tensor:
        return - torch.exp(- torch.square(ratios - self.mu)/(2*torch.square(
            self.sigma)))

    def forward(self,Y:Tensor,**kwargs) -> Tensor:
        """
        :param Y: batched configuration Tensor of shape (B,N,D)
        :return: ratio energy Tensor of shape (B,N)
        """
        ratios = Y[:,:,2] / Y[:,:,3] # compute ratio for each point
        return self.q_fun(ratios) # apply quality function
```

Algorithm I.2: Computation of the ratio energy over batched cells.

Data terms. Data terms are akin to point priors as they are computed for each point independently. We show an example for the position energy term defined in (4.26). Importantly, the

`interpolate` function interpolates values over the map `Z` in a way that the gradient can be back propagated to the configuration `Y`. This allows to perform diffusion easily (see Section 3.3.3).

```
from torch import Tensor, nn
from torch.nn import Module
from torch.nn.functional import grid_sample

class PositionEnergy(Module):
    def __init__(self,threshold:float):
        super(PositionEnergy,self).__init__()
        self.threshold = nn.Parameter(torch.tensor(threshold))

    def q_fun(self, Z_values:Tensor) -> Tensor:
        return torch.log(1 + torch.exp( - Z_values + self.threshold))

    def forward(self,Y:Tensor,Z:Tensor,**kwargs) -> Tensor:
        """
        :param Y: batched configuration Tensor of shape (B,N,D)
        :param Z: energy map Tensor of shape (1,1,H,W)
        :return: position energy Tensor of shape (B,N)
        """
        Z_values = interpolate( # interpolate map values at points positions
            positions = Y[:,:,:2].view(1, n_cells, n_points, 2),
            image = Z
        )
        return self.q_fun(Z_values) # apply quality function

def interpolate(positions:Tensor,image:Tensor) -> Tensor:
    """
    :param positions: Tensor of coordinates (1,B,N,2)
    :param image: image Tensor (1,C,H,W)
    :return: tensor of values at positions (1,1,N,C)
    """
    assert len(image.shape) == 4
    assert len(positions.shape) == 4
    h, w = image.shape[2:]
    samples = torch.stack([ # remap sample coordinates to range [-1,1]
        positions[..., 1] / (w - 1),
        positions[..., 0] / (h - 1)
    ], dim=-1) * 2 - 1
    return grid_sample(image, samples, mode='bilinear', align_corners=True,
        padding_mode='border')
```

Algorithm I.3: Computation of the position energy over batched cells.

Interactions. For interaction priors such as the alignment prior (4.38), we compute energies over interactions as shown in Algorithm I.4. That means handling interaction energy tensors of shape (B, N, N). The quadratic computation cost is kept in check as N is bounded by the maximum number of objects in a cell and its neighbors ($9n_{c,max}$). The tensor `deltas` represents the element-wise differences $\texttt{deltas}[k, l, l', d] = \mathbf{Y}[k, l, d] - \mathbf{Y}[k, l', d]$. Parameter `inbound` is a binary tensor encoding if two points $\mathbf{Y}[k, l], \mathbf{Y}[k, l']$ are neighboring each other. Those are computed with the procedure shown in Algorithm I.5. Computing these interaction tensors outside each energy term module allows those to be re-used for each energy term.

Remark I.1 – Some optimizations remain to be implemented here: for instance the `deltas` *tensor could be computed only on its upper triangular part as each* `deltas`$[k]$ *is an anti-symmetric matrix ($A^\top = -A$).*

```
from torch import Tensor, nn
from torch.nn import Module
import numpy as np

class AlignEnergy(Module):
    def __init__(self):
        super(AlignEnergy,self).__init__()
        self.pi = torch.tensor(np.pi)

    def q_fun(self, angles_dist:Tensor)-> Tensor:
        return - torch.cos(2 * (angles_dist))

    def aggregate(self,align_enr:Tensor, inbound_points: Tensor)-> Tensor:
        w = torch.where( # set out-of-bounds points to torch.inf
            condition=inbound_points,
            input=align_enr, other=torch.tensor([torch.inf])
        )
        aggregated_enr, _ = torch.min(w, dim=2) # aggregate w/ minimum
        return torch.nan_to_num( # points w/o neighbors get 0 energy
            aggregated_enr, nan=0, posinf=0, neginf=0
        )

    def forward(self, Y:Tensor, deltas: Tensor,inbound: Tensor,**kwargs) ->
        Tensor:
        """
        :param Y: batched configuration Tensor of shape (B,N,D)
        :param deltas: delta on each component, Tensor of shape (B,N,N,D)
        :param inbound: Binary tensor of neighboring points, of shape (B,N,N)
        :return: alignment energy Tensor of shape (B,N)
        """
        angle_deltas = deltas[:,:,:,4] # angle deltas tensor of shape (B,N,N)
        angles_dist = torch.minimum( # remap angle differences to [0,pi]
            torch.remainder(angle_deltas, self.pi),
            self.pi - angles_dist
        )
        return self.aggregate( # aggregate interactions
            self.q_func(angles_dist),  # apply quality function
            inbound
        )
```

Algorithm I.4: Computation of the alignment energy over batched cells.

```python
from torch import Tensor, nn
from torch.nn import Module
import numpy as np

def compute_distances_and_deltas(Y:Tensor, M:Tensor, dmax:float)-> Tensor:
    """
    :param Y: batched configuration Tensor of shape (B,N,D)
    :param M: batched mask Tensor of shape (B,N)
    :param dmax: maximum interaction distance
    :return: deltas (B,N,N,D), distances (B,N,N), inbound (B,N,N)
    """
    b = Y.shape[0] # =B
    n = Y.shape[1] # =N
    d = Y.shape[2] # =D
    # use torch boradcasting rules to form a difference tensor of shape (B,N,N,
        D)
    deltas = Y.reshape((b, n, 1, d)) - Y.reshape((b, 1, n, d))
    distances = torch.sqrt( # compute distances from coordinates difference
        torch.sum(torch.square(deltas[..., :2]), dim=-1) + 1e-8
    )
    mask_1 = ~M.reshape((b, n, 1)) # mask out missing points in dim=1
    mask_2 = ~M.reshape((b, 1, n)) # mask out missing points in dim=2
    self_mask = ~torch.eye(n,dtype=bool) # mask out distance to self
    dist_mask = mask_1 | mask_2 | self_mask
    # set missing points (or self) to be out of bounds
    distances = distances + dist_mask * (dmax + 1)
    inbound = distances < dmax # check who is inbound
    return deltas, distances, inbound
```

Algorithm I.5: Computation of the distances and component differences for points in batched cells.

I.2 Generic energy models

The above defined modules all join into the `GenericEnergyModel` module, that computes the total energy of a configuration. Here we present it stripped of the many debugging and logging functions our code actually contains in order to convey the main ideas of the implementation:

```python
from torch import Tensor, nn
from torch.nn import Module,ModuleList
import numpy as np

class GenericEnergyModel(BaseEnergyModel, Module):
    def __init__(self,config):
        super(GenericEnergyModel, self).__init__()
        """
        [...]
        """
        self.energy_modules: ModuleList = load_modules(config)
        # for instance self.energy_modules is a list of Modules such as PositionEnergy, AlignEnergy, RatioEnergy etc defined in the configuration file.
        self.max_dist = self.config['maximum_distance']
        self.energy_names = [p.name for p in self.energy_modules]
        self.energy_combination_module = LinearCombinator(
            energy_names=self.energy_names)

    def forward(self, Y:Tensor, M:Tensor, Z:Tensor):
        deltas, distances, inbound = compute_distances_and_deltas(
            Y=Y, M=M, dmax=self.max_dist)
        energy_dict = {}
        for e in self.energy_modules:
            energy_dict[e.name] = e.forward(
                Y=Y, M=M, Z=Z, deltas=deltas, inbound=inbound)
        energy_vector = torch.stack([energy_dict[k] for k in self.energy_names], dim=-1)
        energy_per_point = self.energy_combination_module.forward(energy_vector) * M
        #mask out the energy of non existing points
        energy_per_cell = torch.sum(energy_per_point, dim=-1)
        total_energy = torch.sum(energy_per_cell)
        return {'energy_per_point': energy_per_point,
                'energy_per_subset': energy_per_cell,
                'total_energy': total_energy,
                **energy_dict}


class LinearCombinator(Module):
    def __init__(self,energy_names:List[str]):
        self.f = nn.Linear(in_features=len(energy_names),out_features=1,bias=True)

    def forward(x:Tensor):
        """
        :param x: tensor (B,N,V) with V the number of energy terms
        :return: tensor (B,N), energy per point
        """
        return self.f(x).squeeze(dim=-1) #squeeze to remove the last dim of size 1
```

Algorithm I.6: GenericEnergyModel, computes the energies of a batch of cells.

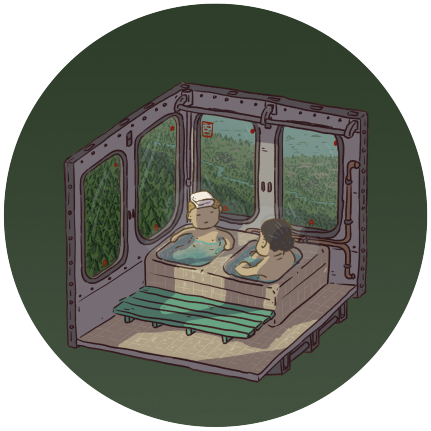
I.3 Miscellaneous methods

Divergence computation. The divergence (needed for computing position potential (4.22)) is computed over a field of 2D vectors defined in the image raster space. In Algorithm I.7 we show how it is computed using Pytorch: using such a library allows to leverage the automatric gradient computation.

```
from torch import Tensor

def divergence(f: Tensor, indexing='ij'):
    """
    :param f: vector field compunents of shape (B,C,H,W) where C is the number of dims
    :param indexing: indexing type, "xy" or "ij"
    :return : Divergence, Tensor of shape (B,1,H,W)
    """
    b = f.shape[0]
    n_dims = f.shape[1]
    spacing = 1.0
    if indexing == "xy":
        stack = [ # for each dim compute the gradient over the corresponding axis
            torch.gradient(f[:, n_dims - i - 1], dim=i + 1)[0] for i in range(n_dims)
            ]
    elif indexing == "ij":
        stack = [ # for each dim compute the gradient over the corresponding axis
            torch.gradient(f[:, i], dim=i + 1)[0] for i in range(n_dims)
            ]
    else:
        raise ValueError # sorry but you did not read the doc !
    stack = torch.stack(stack, dim=1)
    return torch.sum(stack, dim=1, keepdim=True) # sum each gradient over each dim
```

Algorithm I.7: Computation of the divergence of a vector field in PyTorch.



Apprentissage de modèles de géométrie stochastique et réseaux de neurones convolutifs. Application à la détection d'objets multiples dans des jeux de données aérospatiales.

Jules MABON

Résumé

Les drones et les satellites en orbite basse, dont les CubeSats, sont de plus en plus utilisés pour la surveillance, générant d'importantes masses de données à traiter. L'acquisition d'images satellitaires est sujette aux perturbations atmosphériques, aux occlusions et à une résolution limitée. Pour détecter de petits objets, l'information visuelle est limitée. Cependant, les objets d'intérêt (comme les petits véhicules) ne sont pas uniformément répartis dans l'image, présentant des configurations spécifiques. Ces dernières années, les Réseaux de Neurones Convolutifs (CNN) ont montré des compétences remarquables pour extraire des informations, en particulier les textures. Cependant, modéliser les interactions entre objets nécessite une complexité accrue. Les CNN considèrent généralement les interactions lors d'une étape de post-traitement. En revanche, les Processus Ponctuels permettent de modéliser la vraisemblance des points par rapport à l'image et leurs interactions simultanément. La plupart des modèles stochastiques utilisent des mesures de contraste pour la correspondance à l'image ; elles sont adaptées aux objets à contraste fort et faible complexité du fond. Cependant, les petits véhicules sur les images satellitaires présentent divers niveaux de contraste et une grande variété d'objets de fond et de fausses alarmes. Cette thèse de doctorat propose d'utiliser les CNN pour l'extraction d'informations, combinées aux Processus Ponctuels pour modéliser les interactions, en utilisant les sorties CNN comme données. De plus, nous introduisons une méthode unifiée pour estimer les paramètres du modèle de Processus Ponctuel. Nos résultats montrent l'efficacité de ce modèle sur plusieurs jeux de données de télédétection, avec régularisation géométrique et robustesse accrue pour un nombre limité de paramètres.

Mots-clés : Géométrie Stochastique, Modèles à Base d'Énergies, Détection, Objets Multiples, Images Satellitaires, Très Haute Résolution.

Abstract

Unmanned aerial vehicles and low-orbit satellites, including CubeSats, are increasingly used for wide-area surveillance, generating substantial data for processing. Satellite imagery acquisition is susceptible to atmospheric disruptions, occlusions, and limited resolution, resulting in limited visual data for small object detection. However, the objects of interest (e.g., small vehicles) are unevenly distributed in the image: there are some priors on the structure of the configurations. In recent years, convolutional neural network (CNN) models have excelled at extracting information from images, especially texture details. Yet, modeling object interactions requires a significant increase in model complexity and parameters. CNN models generally treat interaction as a post-processing step. In contrast, Point Processes aim to simultaneously model each point's likelihood in relation to the image (data term) and their interactions (prior term). Most Point Process models rely on contrast measures (foreground vs. background) for their data terms, which work well with clearly contrasted objects and minimal background clutter. However, small vehicles in satellite images exhibit varying contrast levels and a diverse range of background and false alarm objects. In this PhD thesis, we propose harnessing CNN models information extraction abilities in combination with Point Process interaction models, using CNN outputs as data terms. Additionally, we introduce a unified method for estimating Point Process model parameters. Our model demonstrates excellent performance on multiple remote sensing datasets, providing geometric regularization and enhanced noise robustness, all with a minimal parameter footprint.

Keywords: Stochastic Geometry, Energy Based Models, Detection, Multiple objects, Satellite Images, Very High Resolution.