Learning Point Processes and Convolutional Neural Networks for Object Detection in Satellite Images

Jules Mabon, Mathias Ortner, Josiane Zerubia

Article

Learning Point Processes and Convolutional Neural Networks for Object Detection in Satellite Images

Jules Mabon [1], Mathias Ortner [2] and Josiane Zerubia [1,*]

[1] Inria, Université Côte d'Azur, 06902 Sophia Antipolis, France; jules.mabon@inria.fr
[2] Airbus Defence and Space, 31400 Toulouse, France; mathias.ortner@airbus.com
* Correspondence: josiane.zerubia@inria.fr

Abstract: Convolutional neural networks (CNN) have shown great results for object-detection tasks by learning texture and pattern-extraction filters. However, object-level interactions are harder to grasp without increasing the complexity of the architectures. On the other hand, Point Process models propose to solve the detection of the configuration of objects as a whole, allowing the factoring in of the image data and the objects' prior interactions. In this paper, we propose combining the information extracted by a CNN with priors on objects within a Markov Marked Point Process framework. We also propose a method to learn the parameters of this Energy-Based Model. We apply this model to the detection of small vehicles in optical satellite imagery, where the image information needs to be complemented with object interaction priors because of noise and small object sizes.

Keywords: convolutional neural networks; point process; energy-based models; remote sensing

1. Introduction

While object detection has been thoroughly studied for the last 20 years [1], the detection of small objects in optical satellite images remains a challenging task: With limited spatial resolution (around 0.5 m/pixel), objects such as cars can be only a few pixels wide. Moreover, this tiny-object scattering density varies greatly, and when closely packed, instances might be difficult to differentiate. However, as the geometrical configuration of one object is often dependent on its neighbors, we can leverage these priors to improve the detection results.

Methods based on convolutional neural networks (CNN) such as Faster R-CNN [2], YOLO [3], or RetinaNet [4] propose to detect objects in "natural" images. Some of these approaches first extract features through convolutions, then propose a series of boxes (anchors) that are refined by regression afterward [5,6]. The specificities of remote-sensing data (high number of small objects) motivate the use of anchor-free methods, relying on heatmap (probability of object center) inference for oriented vehicle detection [7–9], or ship detection [10] (here approximating the heatmap through a vector field).

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. Given the sensor noise or atmospheric perturbations, remote-sensing detection methods need to be innovative to compensate for the limited signal. Approaches such as [11,12] use the temporal information from image time series to improve the detection of small objects. However, these are not suitable for detecting small static objects. To extract object orientation, Ref. [13] focuses on extracting fine-grained features while ignoring any inter-instance interactions. The interaction between nearby objects is key to analyzing a scene for human operators. However, those are often ignored in CNN-based approaches (at least not beyond a post-processing step such as non-maximum suppression) since CNN models are—by construction—most efficient at extracting texture and local information [14]. Meanwhile, long-range relationships are only caught when drastically increasing the depth of the network or by introducing dense connections. For instance, Ref. [15] models interactions through a cascade of Transformers at a great complexity cost, while [16] uses

Citation: Mabon, J.; Ortner, M.; Zerubia, J. Learning Point Processes and Convolutional Neural Networks for Object Detection in Satellite Images. *Preprints* **2024**, *1*, 0. https://doi.org/10.3390/rs16061019

Academic Editor: Xinghua Li

text-modal descriptors to introduce prior knowledge about the objects and their relations into the model.

Meanwhile, Point Processes [17] have been used to perform object detection while relying on a stochastic geometry model to jointly solve the detection and selection of objects with priors. Point Process approaches model the probability on the whole set of points in the image, thus taking into account the interactions between objects. These approaches rely on decomposing the Point Process density into a data term that measures the correspondence of the points against the image and some prior terms that measure the coherence of the point configuration itself. These approaches have shown good results for detection tasks in images with many small objects, such as biological imagery [18] or remote sensing for road [19,20] and building [21] extraction, or tree detection [22].

Previous methods make use of data terms built upon image contrast measures between the interior and exterior of the object [18,20,23] or image gradient [24], which perform great in images with clear contrast between objects and their background. However, in the case of optical satellite images, background diversity, variable visual aspects of objects, and inconsistent illumination make these contrast measures unreliable.

In this paper, 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.

Instead of increasing the complexity of the model by adding, for example, Transformers to a deep CNN architecture, we propose in this paper 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 (Section 2.3). From the latter, we derive more efficient sampling methods for the Point Process that do not rely on application-specific heuristics (Section 2.4). Then, we propose to bridge the gap in terms of parameter estimation using modern learning techniques inspired by Energy-Based Models (Section 2.5). Ultimately, we introduce a novel scoring function that takes into account object interaction in measuring the confidence in the detections and enables explainable results (Section 2.6). The final part presents quantitative and qualitative results on optical satellite data (Section 3).

2. Materials and Methods

In this paper, we propose an object-detection method leveraging CNN-extracted information within a Point Process framework that models object interactions. First, we introduce the foundations for the Point Process on which our model is built, then detail the energy model, along with its sampling and parameter estimation method. We also propose a novel scoring method to evaluate the confidence in the output while considering the interactions and providing explainable results.

2.1. Materials: Datasets

Our application goal with Airbus Defense and Space (ADS) is the detection of small objects in images from satellites such as Pléiades, (Constellation Pléiades [https://pleiades.cnes.fr/en/PLEIADES/GP_systeme.htm, pleiades.cnes.fr, (accessed on 20 October 2023)]) or CO3D, (Constellation Optique 3D [https://www.eoportal.org/satellite-missions/co3d-constellation, eoportal.org, (accessed on 20 October 2023)]), which have a typical spatial resolution (or ground sampling distance) of 0.5 meters per pixel.

To train the CNN and infer the model parameters $\boldsymbol{\theta}$, we compile $DOTA_{0.5}$, a dataset of vehicles in remote-sensing images with 0.5 meters per pixel resolution, from the DOTA dataset [25]. This dataset $DOTA_{0.5}$ is built by sub-sampling images from DOTA to the desired spatial resolution and keeping only `small-vehicle` and `large-vehicle` classes. The data consists of images from satellite or aerial sources of urban or countryside scenes with variable densities of objects (from isolated cars on country roads to dense parking

lots). The images are labeled with sets of oriented rectangles for the objects of interest. We also compile a noisy version of $DOTA_{0.5}$:$DOTA_{0.5}$+*noise*, by adding a Gaussian noise on each image ($\sigma = 0.3$) to test the resilience of the methods against noise. Moreover, we evaluate models on data provided by ADS. These aerial data are sub-sampled to the desired resolution, emulating the above-mentioned satellite characteristics. This dataset is unlabeled; thus, it will only serve to evaluate the models qualitatively and was not used for training.

2.2. Point Process for Object Detection

Point Processes model configurations of points (a finite non-ordered set $\mathbf{y} = \{y^1, \dots, y^n\}$ of elements of $\mathcal{S}$) as realizations of a random variable Φ in the set of all possible configurations $\mathcal{Y} = \bigcup_{n=0}^{\infty} (\mathcal{S} \times \mathcal{M})^n$ (with an arbitrary amount of points). Space $\mathcal{S}$ corresponds to the image space, and $\mathcal{M}$ to the mark space. The marks can be any random variable that describes the point beyond its location, from the radius of a circle to a discrete categorization of the object. Additionally, we denote $n(\mathbf{y})$ the number of points in the configuration $\mathbf{y}$ and $\mathcal{Y}^n$ the set of all n points configurations.

As we look to detect vehicles, we model points as rectangles described by the following marks: width y_a, length y_b, and angle y_α as shown in Figure 1 (with the mark space defined as $\mathcal{M} = [a_{\min}, a_{\max}] \times [b_{\min}, b_{\max}] \times [0, \pi]$).

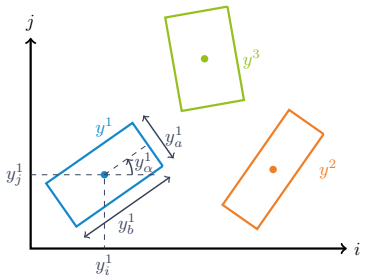


Figure 1. Example configuration with three points; $\mathbf{y} = \{y^1, y^2, y^3\}$.

A Point Process Φ is described by its density h relative to the uniform Point Process [17]. The model of selection and interaction of points derives from an energy U, through a Gibbs density:

$$h(\mathbf{y}|\mathbf{X}) = \frac{1}{Z} \exp(-U(\mathbf{y}, \mathbf{X}, \boldsymbol{\theta})), \tag{1}$$

where $\mathbf{X}$ is the image, $\boldsymbol{\theta}$ the parameters of the model, and Z is a normalizing constant. This constant is intractable due to the non-fixed dimension of $\mathcal{Y}$. However, it can be shown that Z exists (The existence and finiteness of Z are sufficient as the following computations will consider ratios of h; thus Z will cancel out.) and thus h is properly defined as the energy per point is bounded (see Appendix A.2). In short, the energy function U measures the compatibility of the configuration $\mathbf{y}$ with the image $\mathbf{X}$; the lower the energy, the higher the compatibility (see Energy-Based Models [26]). It follows that the most compatible output $\mathbf{y}^*$ minimizes the energy U (i.e. maximizes the density h).

2.2.1. Markovian Point Process

For most applications [18,20,21,23], the physical constraints of the system that is being modeled imply that the marginal density of a point $y \in \mathbf{y}$ should only depend on a neighborhood around it (e.g. vehicles only align with others within a limited distance). This property is formalized through the Markovianity of the Point Process.

Definition 1. *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 $h(\mathbf{y}) > 0$ [17]:*

- $\forall \mathbf{x} \subset \mathbf{y}$, $h(\mathbf{x}) > 0$

- *For every point $u \in \mathcal{S}$, $h(\{u\} \cup \mathbf{y})/h(\mathbf{y})$ only depends on u and its neighbors according to $\sim$ (i.e., $\{y \in \mathbf{y}, y \sim u\}$).*

Theorem 1. *For a Point Process derived from the energy:*

$$U(\mathbf{y}, \mathbf{X}, \boldsymbol{\theta}) = \sum_{y \in \mathbf{y}} V(y, \mathcal{N}_y^{\mathbf{y}}, \mathbf{X}, \boldsymbol{\theta}), \tag{2}$$

where $\mathcal{N}_y^{\mathbf{y}} = \{y' \in \mathbf{y}, y' \sim y, y \neq y'\}$ is the set of neighbors of y for relation $\sim$. Then, the Point Process is Markovian for relation $\sim^2$ with $y \sim^2 y' \equiv \exists u \in \mathcal{S} \times \mathcal{M}, y \sim u \sim y'$.

The proof is provided in Appendix A.1.

In short, if the energy $V(y, \mathcal{N}_y^{\mathbf{y}}, \mathbf{X}, \boldsymbol{\theta})$ of a point y depends on its neighbors within a distance of d_{max}, the Point Process is Markovian for a distance $2d_{max}$. The Markovianity will prove useful to simplify the sampling procedure and run it in parallel over the whole image.

2.2.2. Papangelou Conditional Intensity

The reference Poisson Point Process has an intensity λ that is either constant or depends on the location (For any compact $A \subseteq \mathcal{S}$, $\mathbb{E}[n(\Phi)] = \lambda|A|$ if λ is constant.). The density in (1) implies the intensity is now a function of the location and neighborhood of a point:

Definition 2. *The Papangelou conditional intensity $\lambda(\cdot;\cdot)$ [17] associated with a Point Process Φ, is defined as:*

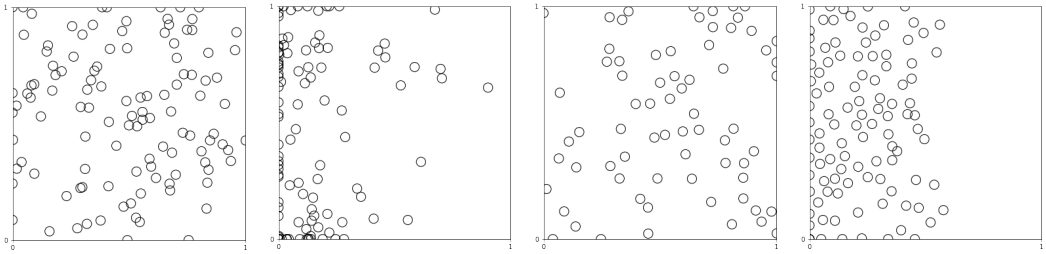
$$\lambda(y; \mathbf{y})dy = p(n_\Phi(dy) = 1 | \Phi \cap (dy)^c = \mathbf{y} \cap (dy)^c), \tag{3}$$

i.e., the infinitesimal probability of finding a point in region dy around $y \in \mathcal{S}$, given the configuration $\mathbf{y}$ outside dy (i.e. $(dy)^c$). When $y \in \mathbf{y}$, the Papangelou conditional intensity can be computed from the energy function U as:

$$\begin{aligned} \lambda(y; \mathbf{y} \setminus \{y\}) &= \frac{h(\mathbf{y}|\mathbf{X})}{h(\mathbf{y} \setminus \{y\}|\mathbf{X})} \\ &= \exp(U(\mathbf{y} \setminus \{y\}, \mathbf{X}, \boldsymbol{\theta}) - U(\mathbf{y}, \mathbf{X}, \boldsymbol{\theta})). \end{aligned} \tag{4}$$

2.3. Energy-Based Model

The energy formulation of the Point Process allows the compositing of several simple point behaviors into one model. We illustrate the composition of several simple point distributions through the addition of energy terms in Figure 2: There, we show that the point distribution in (4) can be obtained by simply adding the energies used to produce (2) and (3).



(**a**) uniform (**b**) gradient (**c**) repulsive (**d**) gradient+repulsive

Figure 2. Point Processes derived from simple energies; (**a**): $V_a(y) \propto 1$, (**b**): $V_b(y) \propto y_i$, (**c**): $V_c(y|\mathcal{N}_y^{\mathbf{y}}) \propto \min_{y' \in \mathcal{N}_y^{\mathbf{y}}} d(y, y')$, (**d**): $V_d(y|\mathcal{N}_y^{\mathbf{y}}) \propto V_b + V_c$. The horizontal axis corresponds to coordinate i and vertical to j.

When performing object detection, the image provides information about the location and properties of objects (e.g., location-based energy as in Figure 2(2)), while prior knowl-

edge of the objects of interest complements this information (e.g., a repulsion term as in Figure 2(3)); the composability of the energy model allows the factoring in of both pieces of information into a single model (as in the composition of Figure 2(4)).

More specifically: For every point $y \in \mathbf{y}$ we compute a set of energy terms $V_e(\mathbf{y}, \mathbf{X}, \mathcal{N}_y^{\mathbf{y}})$, $e \in \xi$ and combine those—per point—as a weighted sum [20,21,23]:

$$\begin{aligned} U(\mathbf{y}, \mathbf{X}, \boldsymbol{\theta}) &= \sum_{y \in \mathbf{y}} V\left(y, \mathbf{X}, \mathcal{N}_y^{\mathbf{y}}, \boldsymbol{\theta}\right) \\ &= \sum_{y \in \mathbf{y}} w_0 + \sum_{e \in \xi} w_e V_e\left(y, \mathbf{X}, \mathcal{N}_y^{\mathbf{y}}, \boldsymbol{\theta}\right) \end{aligned} \tag{5}$$

with $V(y, \mathbf{X}, \mathcal{N}_y^{\mathbf{y}}, \boldsymbol{\theta})$ is the energy contribution of a single object y. Energy terms can be grouped into two categories:

- *Prior terms* (of the form $V_e(y, \mathcal{N}_y^{\mathbf{y}}, \boldsymbol{\theta})$) that only depend on y and its neighborhood $\mathcal{N}_y^{\mathbf{y}}$; they measure the coherence of the configuration itself considering the known properties of the objects of interest (e.g., objects do not overlap).
- *Data terms* ($V_e(y, \mathbf{X}, \boldsymbol{\theta})$) are a function of y and the image $\mathbf{X}$ (also referred to as observation in remote sensing); these terms measure the correspondence of the points with the image.

The weights w are part of the parameters $\boldsymbol{\theta}$ that will need to be set before inference on any unseen image. We discuss the estimation of parameters $\boldsymbol{\theta}$ in Section 2.5.

In the rest of this section, we define the multiple energy terms V_e: First, the data terms , then the prior terms. Contrary to previous Point Process (PP) approaches, we do not rely on contrast measures but rather on data terms pulled from the output of a CNN: this allows for more reliable measures and faster computation. Moreover, this new model formulation opens up new improvements in sampling the energy model and estimating its parameters.

2.3.1. Data Terms from a CNN

Classical likelihood measurements [18,20,23] are based on contrast measures that are crafted to fit each application. Moreover, these measures rely heavily on the high contrast between objects and their background, along with limited background diversity. We illustrate the limitations of those in Section 3.1.

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 a scalar 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.

Position Likelihood Term

CNN-based object-detection methods such as [7,8,27] make use of a heatmap to find object centers. This object center probability map is obtained by passing the image $\mathbf{X}$ of shape $H, W, 3$ (here with 3 color channels) into a fully convolutional model such as Unet [28]. It outputs a tensor $\widehat{\mathbf{Z}}_{pos} \in \mathbb{R}^{H \times W}$, from which a probability-like measure of an object center at coordinates (y_i, y_j) is obtained as $p(y_i, y_j | \mathbf{X}) = \sigma(\widehat{\mathbf{Z}}_{pos}[y_i, y_j])$ where $\sigma(\cdot)$ is the sigmoid function, and $\widehat{\mathbf{Z}}_{pos}[y_i, y_j]$ the value of map $\widehat{\mathbf{Z}}_{pos}$ at coordinates (y_i, y_j).

We propose to reinterpret this output as the position energy as follows:

$$V_{pos}(y, \mathbf{X}, \boldsymbol{\theta}) = \ln\Big(1 + \exp\Big(-\widehat{\mathbf{Z}}_{pos}[y_i, y_j] + t_{pos}\Big)\Big), \tag{6}$$

with t_{pos} a scalar threshold, allowing the movement of the inflection point of the Softplus function $x \mapsto \ln(1 + \exp(-x))$.

Mark Likelihood Term

To compute the energy associated with marks, 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:

$$c_\kappa(v) = n_\kappa \frac{v - \kappa_{\min}}{\kappa_{\max} - \kappa_{\min}}, \forall v \in [\kappa_{\min}, \kappa_{\max}], \tag{7}$$

with $\lceil c_\kappa(v) \rceil$ being the corresponding integer class.

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

$$\hat{p}(c|i, j, \mathbf{X}) = \text{Softmax}(\widehat{\mathbf{Z}}_\kappa^{y_i, y_j})_c = \frac{\exp\left(\widehat{\mathbf{Z}}_\kappa^{y_i, y_j}[c]\right)}{\sum_{c'=1}^{n_\kappa} \exp\left(\widehat{\mathbf{Z}}_\kappa^{y_i, y_j}[c']\right)}. \tag{8}$$

As in [29], we can reinterpret the model output into a potential, giving:

$$V_\kappa(y, \mathbf{X}) = -\widehat{\mathbf{Z}}_\kappa^{y_i, y_j}[\lceil c_\kappa(y_\kappa) \rceil] + \ln\left(\sum_{c=1}^{n_\kappa} \exp\left(\widehat{\mathbf{Z}}_\kappa^{y_i, y_j}[c]\right)\right). \tag{9}$$

As the CNN outputs a tensor $\widehat{\mathbf{Z}}_\kappa$ of shape (H, W, n_κ), we obtain vector $\widehat{\mathbf{Z}}_\kappa^{y_i, y_j}$ by sampling tensor $\widehat{\mathbf{Z}}_\kappa$ at location $(\lceil y_i \rceil, \lceil y_j \rceil)$, as illustrated in Figure 3a. Vector $\widehat{\mathbf{Z}}_\kappa^{y_i, y_j}$ gives an estimate of the probability of each class c once passed through a Softmax (as given by (8)); We illustrate this with the histogram, on top of $\widehat{\mathbf{Z}}_\kappa^{y_i, y_j}$ in Figure 3a. The value of $V_\kappa(y, \mathbf{X})$ is then obtained by applying the formula in (9) over vector $\widehat{\mathbf{Z}}_\kappa^{y_i, y_j}$ (as illustrated on the right-hand side of Figure 3).

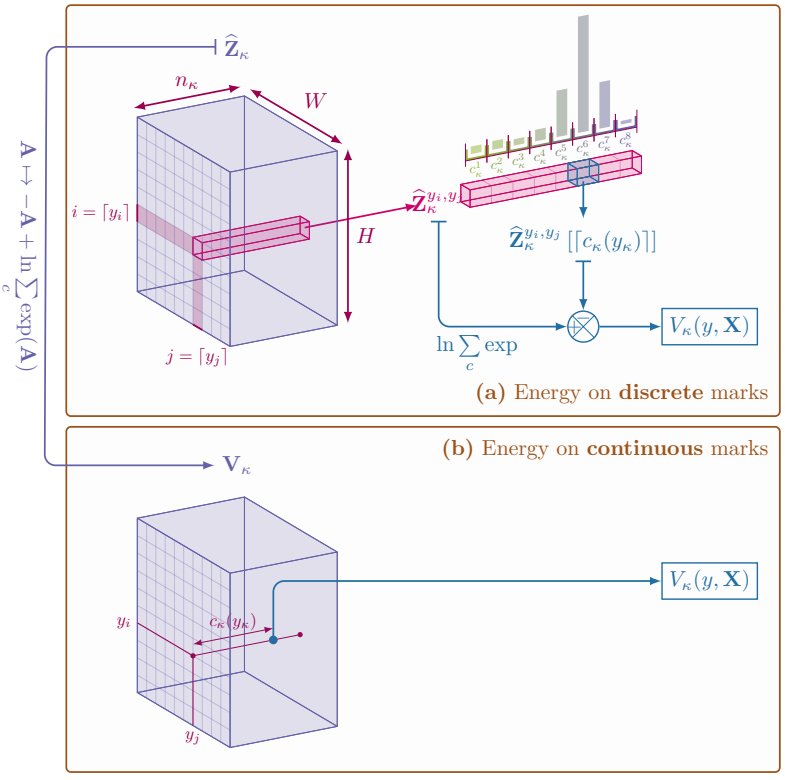


Figure 3. Energy over discrete marks (**a**) and continuous marks (**b**).

Interpolation

The energies described in (6) and (9) are defined over integer coordinates in $\mathcal{S}$ and integer classes in $\mathcal{M}$. However, points y in the configuration $\mathbf{y}$ have real-valued locations $(y_i, y_j) \in \mathbb{R}^2$ as is for marks in $\mathcal{M}$. Thus, we propose interpolating values between exact pixel locations. For (9) we now have :

$$V_\kappa(y, \mathbf{X}) = \mathbf{V}_\kappa[y_i, y_j, c_\kappa(y_\kappa)] \tag{10}$$

$$\mathbf{V}_\kappa[i,j] = -\widehat{\mathbf{Z}}_\kappa[i,j] + \ln \sum_{c=1}^{n_\kappa} \exp\Big(\widehat{\mathbf{Z}}_\kappa[i,j,c]\Big),\ \forall i,j \in \mathcal{S}_d. \tag{11}$$

The above has two benefits: first, (10) defines the energy for continuous values of the marks using bilinear interpolation, thus ensuring Lipschitz continuity of the energy. Second, we can pre-compute (11) once for a given image, making the mark energy computation a simple value lookup and interpolation for each point. In practice, we proceed as illustrated in Figure 3b, where a function is applied to the CNN output $\widehat{\mathbf{Z}}_\kappa$ to obtain $\mathbf{V}_\kappa$, which is stored in memory. The interpolation gives V_κ for any continuous position and mark. 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 [30]. The same holds for the position term (6).

2.3.2. Energy Priors

The total energy model encompasses several priors as energies, which allows regularizing configuration against the data terms.

Object Priors

These are functions of the current point y only. For $k = ratio, area$:

$$V_k(y, \boldsymbol{\theta}) = -\exp\Big(-0.5(f_k(y) - \mu_k)^2 \sigma_k^{-2}\Big) \tag{12}$$

with $f_{ratio}(y) = y_b/y_a$ and $f_{area}(y) = y_a y_b$, and μ_k,σ_k being, respectively, the target value and the standard deviation. These two are illustrated in the first line of Figure 4.

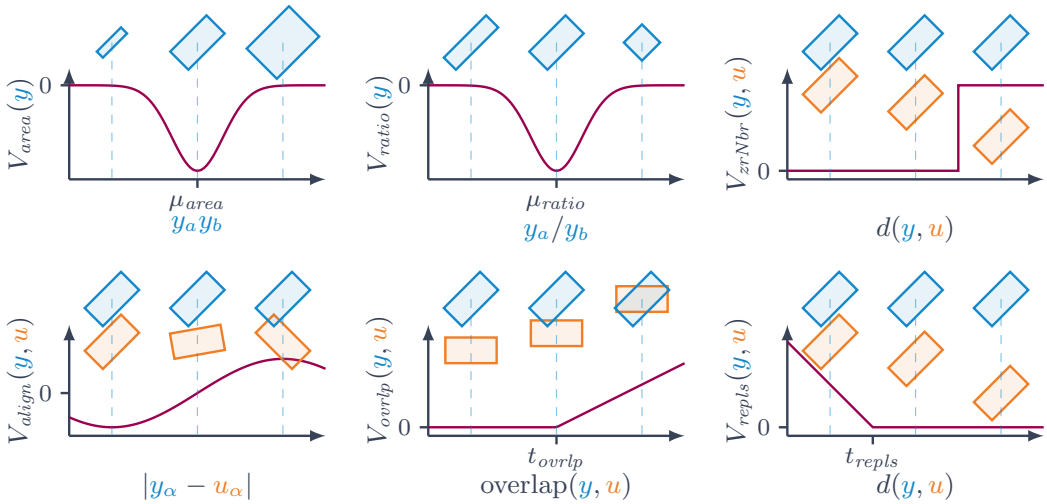


Figure 4. Illustration of some priors. For priors acting on pairs of objects (*zrNbr*, *align*, *ovrlp*, *repls*), we show the energy for a pair of objects y and u (see (14) to (18) for the aggregation of multiple pair interactions into a single energy scalar).

For our objects of interest, we find two modes in the distribution of areas and ratios: Cars and trucks. To make sure to only favor these two modes and not objects with the area of a truck and ratio of a car, we introduce a joint area and ratio prior:

$$V_{jntAR}(y,\boldsymbol{\theta}) = -\exp\left(-\frac{\left(\frac{y_a}{y_b}-\mu_{ratio}\right)^2}{2\sigma_{ratio}^2} - \frac{(y_a y_b - \mu_{area})^2}{2\sigma_{area}^2}\right). \tag{13}$$

Interaction Priors

The following priors depend on the neighborhood of the point y. The term in (14) penalizes overlapping objects (with $t_{overlap}$ the overlap threshold), (15) favors aligned objects ($t_\alpha = 0$ favors parallel objects, while $t_\alpha = \pi/2$ prefers perpendicular objects), (16) and (17) are, respectively, repulsive and attractive priors. Finally, (18) allows the adjustment of the energy of neighborless points. Some of these priors are illustrated in Figure 4.

$$V_{ovrlp}(y,\mathcal{N}_y^{\mathbf{y}}) = \max_{y'\in\mathcal{N}_y^{\mathbf{y}}}\left\{\text{Relu}\left(\frac{\text{area}(y'\cap y)}{\min\{\text{area}(y'),\text{area}(y)\}} - t_{ovrlp}\right)\right\} \tag{14}$$

$$V_{align}(y,\mathcal{N}_y^{\mathbf{y}}) = \min_{y'\in\mathcal{N}_y^{\mathbf{y}}}\left\{-\cos\left(|y_\alpha - y'_\alpha|\right)\right\} \tag{15}$$

$$V_{repls}(y,\mathcal{N}_y^{\mathbf{y}}) = \max_{y'\in\mathcal{N}_y^{\mathbf{y}}}\left\{\text{Relu}\left(1 - \frac{d(y,y')}{d_{max}} - t_{repls}\right)\right\} \tag{16}$$

$$V_{attrc}(y,\mathcal{N}_y^{\mathbf{y}}) = \min_{y'\in\mathcal{N}_y^{\mathbf{y}}}\left\{\text{Relu}\left(\frac{d(y,y')}{d_{max}} - t_{attrc}\right)\right\} \tag{17}$$

$$V_{zrNbr}(y,\mathcal{N}_y^{\mathbf{y}}) = \mathbb{1}_{|\mathcal{N}_y^{\mathbf{y}}|=0} \tag{18}$$

where $\text{Relu}(x) = \max(0,x)$ for $x \in \mathbb{R}$. Note we introduced a few parameters $\mu_{ratio}, \sigma_{ratio}, t_{ovrlp}, \ldots$, these will have to be either set by trial and error or through the parameter estimation method presented in Section 2.5.

2.3.3. Model Pipeline

We summarize the implementation of the overall energy model in Figure 5. As is, our architecture has the following advantages:

- We simplify the data term computation from complex contrast measures [18,20,23] into a simple sampling and interpolation of a tensor (highlighted in green in Figure 5).
- The backbone CNN architecture remains very simple as it does not need to model any object interactions.
- The CNN inference—which is the most complex operation within the graph in Figure 5 (FCN block)—is limited to once per image, as the energy U can be computed for any configuration $\mathbf{y} \in \mathcal{Y}$ without having to infer $\mathbf{V}_{pos}$ and $\mathbf{V}_\kappa$ a second time (pre-computed maps highlighted in lilac in Figure 5).
- New interactions or priors can be easily added to the model as energies are aggregated through a simple linear combination (see right-hand part of the pipeline in Figure 5).
- The implementation with tensor computation libraries (here PyTorch [30]) allows for implicit parallelization on GPU (or CPU) using the tensors batch dimension.
- The latter also enables automatic differentiation of energy U, both relative to configuration $\mathbf{y}$ and to parameters $\boldsymbol{\theta}$, which will be, respectively, useful in Sections 2.4.3 and 2.5.4.
- Lastly, as the energy maps are defined over a raster space (thus easy to normalize), those will be of use to design the birth map in Section 2.4.1.

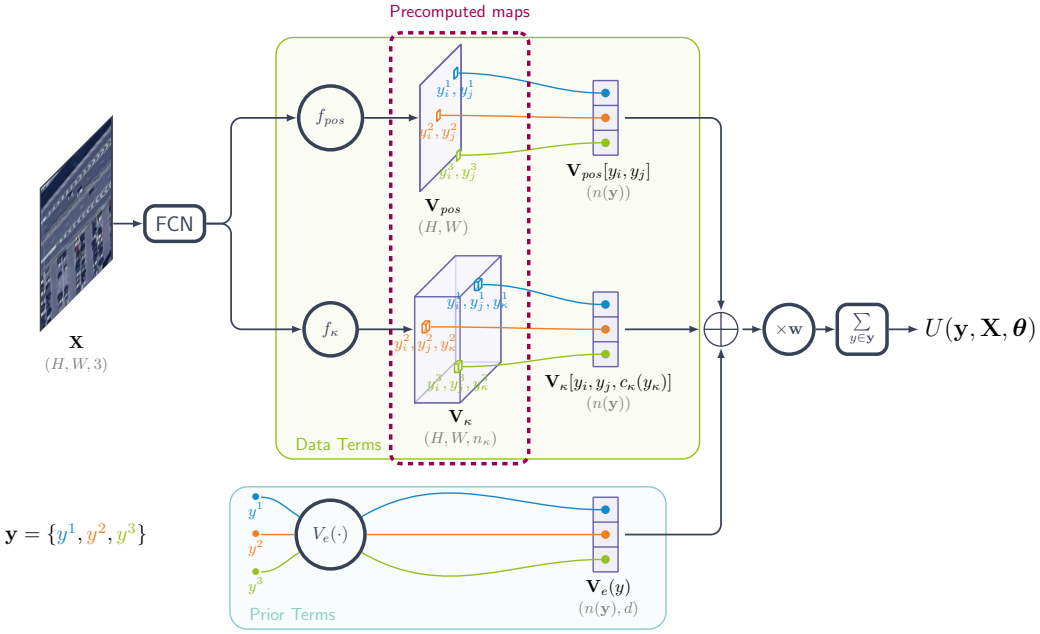


Figure 5. Energy model pipeline. The *pre-computed* tensors can be reused for computing different $\mathbf{y} \in \mathcal{Y}$ for a given image $\mathbf{X}$.

2.4. Sampling Configurations from the Energy Model

For a given energy model U (supposing parameters $\boldsymbol{\theta}$ are set), we aim at sampling configurations that follow the Gibbs density defined in (1). This will allow us to find the most compatible configuration $\mathbf{y}^*$ for any image $\mathbf{X}$, i.e.:

$$\mathbf{y}^* = \arg\min_{\mathbf{y} \in \mathcal{Y}} U(\mathbf{y}, \mathbf{X}, \boldsymbol{\theta}). \tag{19}$$

As the space $\mathcal{Y}$ in which we are trying to minimize the energy is not of fixed density, we must resort to an adapted sampling method such as Reversible Jump Monte Carlo Markov Chain (RJMCMC) [31]. While this method ensures proper sampling of the desired law, it often requires some application-specific adaptations to improve sampling time and efficiency (e.g., [20,23]).

Here, we propose to use the already computed energy maps $\mathbf{V}_{pos}$ and $\mathbf{V}_\kappa$ to add new points in relevant areas and to focus on the parallel sampling of the algorithm. Moreover, the implementation of our energy model allows us to leverage automatic gradient computation to implement diffusion dynamics and explore at a fixed number of points (i.e., within $\mathcal{Y}_n$) guided by the whole energy model.

2.4.1. Birth Maps for Efficient Point Proposals

At their core, both RJMCMC [31] (an adaptation of the Metropolis-Hastings algorithm [32] to variable dimension problems) and Jump Diffusion [33] make use of Birth and Death moves to build a Markov chain $(\mathbf{y}_t)_{t=1,\dots}$ of stationary density $h(\cdot|\mathbf{X})^{1/T_t}$ where T_t is a temperature parameter that decreases towards zero (i.e. Simulated Annealing) (Convergence is proven for a logarithmic temperature decrease, here (and in the literature) we approximate it with a geometric decrease at it is faster [17]). That way, the chain converges towards the global minimum energy. Proposed Birth (addition of points to the current y_t) and Deaths (removal of points) are accepted or rejected given an acceptance ratio that depends on the energy change induced by the respective addition or removal of points (for more details see [31,34]).

The simplest birth move is to propose new points uniformly in $\mathcal{S} \times \mathcal{M}$. However, this proves highly inefficient as the density of objects in the image varies a lot. Ideally, the birth move would propose a new point u to the current configuration $\mathbf{y}$, sampled with the

marginal density $p(u|\mathbf{y})$ as is carried out within Gibbs sampling (maximizing the chance for the point addition to be accepted). Knowing that $p(u|\mathbf{y}) \propto h(\mathbf{y} \cup \{u\})/h(\mathbf{y})$, we could compute the marginal density using energy change induced by adding the point. However, this cannot easily be computed because of the interaction terms. Thus, we propose to approximate it by only considering the data terms (i.e. bypassing the energy change on prior terms) :

$$\frac{h(\mathbf{y} \cup \{u\})}{h(\mathbf{y})} \simeq \exp\left(-w_{pos} V_{pos}(u, \mathbf{X}) - \sum_{\kappa \in \{a,b,\alpha\}} w_\kappa V_\kappa(u, \mathbf{X})\right). \tag{20}$$

From this, we define the density d to sample a new point u:

$$d(u) = \frac{1}{Z_d} \exp\left(-w_{pos} \mathbf{V}_{pos}[u] - \sum_{\kappa \in \{a,b,\alpha\}} w_\kappa \mathbf{V}_\kappa[u])\right), \tag{21}$$

with $\mathbf{V}_{pos}$ and $\mathbf{V}_\kappa$ the pre-computed position and marks energy maps. Since both are defined as tensors with a finite number of elements, the normalizing constant Z_d in (21) can be computed over this discretized space of pixels and mark classes.

This allows for efficient sampling of new points, without any application-specific heuristics, by taking into account a truncated version of the energy model U that can be normalized easily.

2.4.2. Focused Parallel Sampling

In its canonical implementation, the RJMCMC algorithm operates sequentially over the points in the image—It may only add/remove/transform one point at a time —, which increases simulation time linearly with the number of objects. With parallelization, we aim to take advantage of the spatial Markovianity of the Point Process, as Theorem 1 states that moves further than $2d_{max}$ apart are independent.

As in [35], we split the image into square cells of size d_c that are each assigned one set s (each set corresponding to one color in Figure 6). These sets are referred to as mic-sets in [35] for Mutually Independent Cells. Set size d_c is chosen as $2d_{max} + 2\delta_{max}$ (δ_{max} is the maximum point translation distance, introduced in the next Section), so that moves in cells of one set (or color) have independent acceptance ratios: i.e., those moves can be performed in any sequential order –thus in parallel—without any change in the probability of acceptance. In practice, we have $d_c = 48$ ($d_{max} = 16$, $\delta_{max} = 8$).

We aim at leveraging the birth map stemming from the pre-computed energy maps $\mathbf{V}_{pos}$ and $\mathbf{V}_\kappa$ to focus the sampling on parts of the image where the density of objects is high. Contrary to [35], we do not use a quadtree structure of cells, but rather a regular grid of d_c sized cells. In order to achieve an efficient sampling of cells (i.e., avoiding spending time in cells with no evidence of objects), weights are assigned to each cell to focus the sampling on the relevant ones. The sampling procedure goes as follows:

1. pick a move type (Birth, Death, or Diffusion),
2. pick one set s with probability $p(s)$,
3. keep each cell in set s with probability $p(c|s)$
4. for every cell c in set s run move restricted to c (Birth density becomes $d_c(u)$)

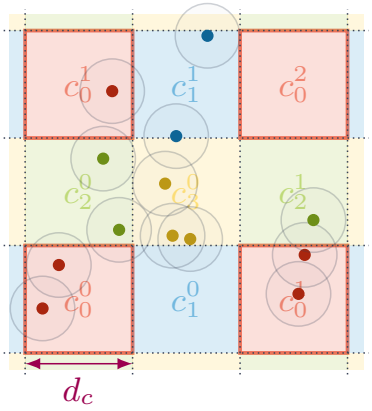


Figure 6. Space partitioning, where each color corresponds to a different set s. Each cell c_s^z (square of size d_c), is the z^{th} cell of set s. For every point y, we display its interaction radius d_{max} as a semi-transparent circle.

The process of cell selection (step 3) allows for limiting the number of cells processed at once on big images, hence limiting the computational cost and focusing the sampling on areas with high object density. We pick the following densities:

$$p(s) = d(s), \tag{22}$$

$$p(c|s) = \min(1, n_p \frac{d(c)}{d(s)}), \tag{23}$$

$$d_c(u) = \frac{d(u)}{d(c)}, \tag{24}$$

denoting (by extension) $d(s)$ and $d(c)$ the density d integrated over all cells of set s and cell c, respectively. In practice, parameter $n_p = 1$ ensures new points are sampled over the whole image with density d; higher values of n_p allow for more parallel cells to be sampled each step at the cost of straying away from density d.

2.4.3. Jump Diffusion: Leveraging Gradients

The canonical RJMCMC algorithm also uses local transform moves to explore $\mathcal{Y}$ at a fixed number of points, picking one point to translate, rotate, and scale at random in $\mathbf{y}_t$. This is wildly inefficient as it does not consider the energy function U it is trying to minimize. Instead, we propose to use the energy gradient to explore the space more efficiently. This is the idea behind stochastic diffusion (or Langevin) dynamics [36,37]. If $\mathbf{y}_t$ and T_t denote, respectively, the configuration and temperature at time t, then the configuration for the next step of the Markov chain is given by $\mathbf{y}_{t+1} = \mathbf{y}_t + d\mathbf{y}_t$, with step size δ:

$$d\mathbf{y}_t = -\delta \frac{\partial U(\mathbf{y}_t, \mathbf{X}, \boldsymbol{\theta})}{\partial \mathbf{y}_t} + dw_t \sqrt{2T_t}, \quad dw_t \sim \mathcal{N}(0, \delta). \tag{25}$$

At high temperatures, the Brownian motion from dw_t ensures an exploration of space. At low temperatures, the Brownian motion is negligible, and the diffusion performs as a gradient descent. This allows fine-tuning the configuration at low temperatures while considering the whole energy model (contrary to the truncated energy used for birth maps in Section 2.4.1).

The diffusion alone allows for the exploration of space $\mathcal{Y}$ locally around $\mathbf{y}_t$ at a fixed dimension (fixed number of points). To explore (or *jump*) across dimensions, we make use of the Birth and Death moves as defined previously (see Section 2.4.1).

In practice, as parallelization requires some maximum displacement on points (see Section 2.4.2), the step $d\mathbf{y}_t$ from $\mathbf{y}_t$ to $\mathbf{y}_{t+1}$ is clipped; for each point $y \in \mathbf{y}_t$ updated to y', we bound the i and j components by δ_{max}(i.e., $|y_i - y'_i| \leq \delta_{max}$, and similarly for j).

2.4.4. Resulting Sampling Method

The resulting sampling method is outlined in Algorithm 1. The method requires the following inputs:

- $\mathbf{x}_0$: initial configuration;
- $\mathbf{X}$: image;
- $\boldsymbol{\theta}$: energy model parameters;
- T_0: initial temperature;
- n_s: number of samples;
- α: temperature decay rate.

Here, we denote by $C_c(\mathbf{x})$ the restriction of configuration $\mathbf{x}$ to cells c. Please note that to compute U for iteration of the loop on t, and the energy maps $\mathbf{V}_{pos}$, $\mathbf{V}_\kappa$ are only computed once as stated in Section 2.3.3.

Algorithm 1: Sampling method.

Input: $\mathbf{x}_0, \mathbf{X}, \boldsymbol{\theta}, T_0, n_s, \alpha$

1: **for** $t = 0, \dots, n_s - 1$ **do**
2: Pick `diffusion` with probability 0.8, else `jump`
3: Pick mic-set s with probability $p(s)$
4: Keep each c in s with probability $p(c|s)$ to make $\tilde{s}$
5: $\mathbf{x}_{t+1} \leftarrow \mathbf{x}_t$ *unvisited cells will keep the previous state*
6: **for all** $c \in \tilde{s}$ **do**
7: **if** `diffusion` **then**
8: $dw \sim \mathcal{N}(0, \beta)$
9: $\Delta C_c(\mathbf{x}_t) = -\beta \nabla_{C_c(\mathbf{x}_t)} U(\mathbf{x}_t, \mathbf{X}, \boldsymbol{\theta}) + dw\sqrt{2T_t}$
10: $C_c(\mathbf{x}_{t+1}) \leftarrow C_c(\mathbf{x}_t) + \Delta C_c(\mathbf{x}_t)$
11: **else**
12: $Q_c \leftarrow Q_{c,\mathrm{B}}$ with probability 0.5 else $Q_{c,\mathrm{D}}$ *pick birth or death*
13: $\mathbf{x}' \sim Q_c(\mathbf{x} \to \cdot)$ *remove a point at random or add one using the birth map*
14: $r \leftarrow \frac{Q_c(\mathbf{x}' \to \mathbf{x})}{Q_c(\mathbf{x} \to \mathbf{x}')} \exp\left(-\frac{U(\mathbf{x}', \mathbf{X}, \boldsymbol{\theta}) - U(\mathbf{x}, \mathbf{X}, \boldsymbol{\theta})}{T_t}\right)$ *compute Green ratio*
15: $C_c(\mathbf{x}_{t+1}) \leftarrow C_c(\mathbf{x}')$ with probability $\min(1, r)$
16: **end if**
17: **end for**
18: $T_{t+1} \leftarrow \alpha T_t$ *decrease temperature*
19: **end for**

Output: $\mathbf{x}_{n_s}$

The above implements the sampling improvements described previously, namely:

- sampling new points using the birth map (line 13),
- sampling in parallel over the whole image (line 6), in practice using tensor batch dimensions,
- using the energy gradient to perform Diffusion (line 9).

2.5. Estimating Model Parameters

In this section, we estimate the parameters $\boldsymbol{\theta}$, so that, for each image $\mathbf{X}$, the associated ground truth configuration $\mathbf{y}^{GT}$ matches the most compatible configuration, i.e.:

$$\mathbf{y}^{GT} = \arg\min_{\mathbf{y} \in \mathcal{Y}} U(\mathbf{y}, \mathbf{X}, \boldsymbol{\theta}) \tag{26}$$

Previous PP approaches for object detection often relied on trial-and-error parameter estimation or used linear programming [23,38]. The latter method would generate a set of constraints (These constraints can be seen as a local reformulation of global constraint (26).) $U(\mathbf{y}^-) \leq U(\mathbf{y}^{GT})$ where *wrong configurations* $\mathbf{y}^-$ are generated by applying strong

perturbations on the ground truth configuration $\mathbf{y}^{GT}$. However, this only estimates the weights w_e and is prone to over-constraining when the ground truth is noisy or when considering too many constraints. Moreover, this method requires designing the procedure to generate the configurations $\mathbf{y}^-$, which we find has a great effect on the estimated parameters.

2.5.1. Maximum Likelihood Learning

To tackle the above limitations, we turn towards the Energy-Based Model (EBM) literature for new parameter estimation methods. The authors in [26] propose to learn EBM parameters by maximizing the likelihood for the data $\mathcal{D}$:

$$p\left(\mathbf{y}_1^{GT}, \dots, \mathbf{y}_{|\mathcal{D}|}^{GT} | \mathbf{X}_1, \dots, \mathbf{X}_{|\mathcal{D}|}, \boldsymbol{\theta}\right) = \prod_{k=1}^{|\mathcal{D}|} p\left(\mathbf{y}_k^{GT} | \mathbf{X}_k, \boldsymbol{\theta}\right). \tag{27}$$

The negative log-likelihood is then given, for parameters $\boldsymbol{\theta}_n$ at step n, as:

$$\mathcal{L}_{NLL}(\boldsymbol{\theta}_n, \mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{k=1}^{|\mathcal{D}|} \left(U\left(\mathbf{y}_k^{GT}, \mathbf{X}_k^+, \boldsymbol{\theta}_n\right) + \frac{1}{\beta} \log \int_{\mathbf{y} \in \mathcal{Y}} \exp(-\beta U(\mathbf{y}, \mathbf{X}_k, \boldsymbol{\theta}_n)) \right) \tag{28}$$

with β an inverse temperature parameter (from the Gibbs distribution) that does not affect the position of the minimum [26]. The gradient of $\mathcal{L}_{NLL}$ on $\boldsymbol{\theta}_n$ is shown to be

$$\frac{\partial \mathcal{L}_{NLL}(\boldsymbol{\theta}_n, \mathbf{y}_i^{GT}, \mathbf{X}_i)}{\partial \boldsymbol{\theta}_n} = \frac{\partial U(\mathbf{y}_i^{GT}, \mathbf{X}_i, \boldsymbol{\theta}_n)}{\partial \boldsymbol{\theta}_n} - \int_{\mathbf{y} \in \mathcal{Y}} \frac{\partial U(\mathbf{y}, \mathbf{X}_i, \boldsymbol{\theta}_n)}{\partial \boldsymbol{\theta}_n} p(\mathbf{y} | \mathbf{X}_i, \boldsymbol{\theta}_n) \tag{29}$$

where $p(\mathbf{y} | \mathbf{X}_i, \boldsymbol{\theta}_n) = \exp(-\beta U(\mathbf{y}, \mathbf{X}_i, \boldsymbol{\theta}_n)) / \int_{\tilde{\mathbf{y}} \in \mathcal{Y}} \exp(-\beta U(\tilde{\mathbf{y}}, \mathbf{X}_i, \boldsymbol{\theta}_n))$.

While the integral $\int_{\mathbf{y} \in \mathcal{Y}} \frac{\partial U(\mathbf{y}, \mathbf{X}_i, \boldsymbol{\theta}_n)}{\partial \boldsymbol{\theta}_n} p(\mathbf{y} | \mathbf{X}_i, \boldsymbol{\theta}_n)$ remains intractable, it can be approximated through Monte Carlo sampling, where $\tilde{\mathbf{y}}^0, \dots, \tilde{\mathbf{y}}^N$ are drawn from the law defined by $p(\cdot | \mathbf{X}_i, \boldsymbol{\theta}_n)$, which yields:

$$\int_{\mathbf{y} \in \mathcal{Y}} \frac{\partial U(\mathbf{y}, \mathbf{X}_i, \boldsymbol{\theta}_n)}{\partial \boldsymbol{\theta}_n} p(\mathbf{y} | \mathbf{X}_i, \boldsymbol{\theta}_n) \simeq \frac{1}{N} \sum_{k=1}^{N} \frac{\partial U\left(\tilde{\mathbf{y}}^k, \mathbf{X}_i, \boldsymbol{\theta}_n\right)}{\partial \boldsymbol{\theta}_n} \tag{30}$$

2.5.2. Contrastive Divergence

The authors in [39,40] propose to use a single sample in their Contrastive Divergence (CD) method. This method also uses a 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 $U(\cdot, \mathbf{X}, \boldsymbol{\theta}_n)$ at step n of the optimization. Then we proceed to update $\boldsymbol{\theta}_n$ to $\boldsymbol{\theta}_{n+1}$, by gradient descent, to minimize the energy of the *valid sample* $\mathbf{y}^{GT}$, while maximizing the energy of the *contrastive sample* $\mathbf{y}^-$ (see Figure 7). 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 [41]. In the case of imperfect GT, this models the uncertainty over the labels while providing some data augmentation.

The loss to minimize is then:

$$\begin{aligned} \mathcal{L}(\boldsymbol{\theta}_n, \mathbf{y}^+, \mathbf{y}^-, \mathbf{X}) &= U(\mathbf{y}^+, \mathbf{X}, \boldsymbol{\theta}_n) - U(\mathbf{y}^-, \mathbf{X}, \boldsymbol{\theta}_n) + \gamma R_V \\ R_V &= \sum_{\mathbf{y} \in \{\mathbf{y}^+, \mathbf{y}^-\}} \frac{1}{n(\mathbf{y})} \sum_{y \in \mathbf{y}} \left| V(\mathbf{y}, \mathbf{X}, \mathcal{N}_y^{\mathbf{y}}, \boldsymbol{\theta}_n) \right|, \end{aligned} \tag{31}$$

with $\gamma > 0$ the weight of regularization term R_V. We introduce the regularization term to avoid an explosion of the per-point energy $V(y, \mathbf{X}, \mathcal{N}_y^{\mathbf{y}}, \boldsymbol{\theta}_n)$. To ensure 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 [42]:

$$R_1 = \sum_{e \in \xi} |w_e|. \tag{32}$$

The broad strokes of the estimation procedure for parameters $\boldsymbol{\theta}$ 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 $\mathbf{y}^- \sim \exp(-\beta^{-1} U(\mathbf{y}^-, \mathbf{X}, \boldsymbol{\theta}_n))$.
4. Compute the loss $\mathcal{L}(\boldsymbol{\theta}_n, \mathbf{y}^+, \mathbf{y}^-, \mathbf{X})$ (see (31)).
5. Update $\boldsymbol{\theta}_n$ to $\boldsymbol{\theta}_{n+1}$ according to the gradient $\nabla \mathcal{L}$, with the Stochastic Gradient Descent (SGD) method [43], as illustrated in Figure 7.
6. Loop back to step 1 until convergence.

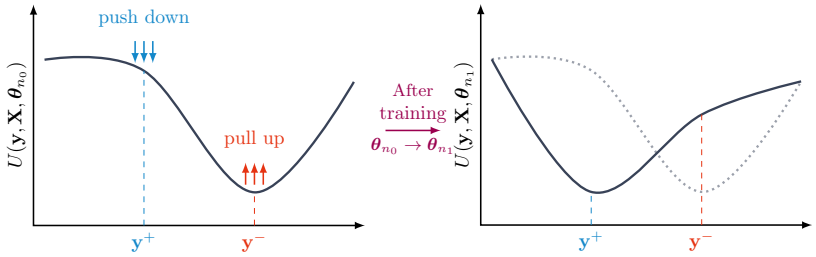


Figure 7. Effect of training the energy with contrastive samples generated from the current $\boldsymbol{\theta}_n$, amplified for illustrative purposes, as the gradient step that updates $\boldsymbol{\theta}_n$ is small (Figure adapted from [26]).

2.5.3. Replay Buffer

As sampling contrastive samples $\mathbf{y}^-$ can be time-consuming, we use the replay buffer introduced by [41]. The replay buffer saves Markov chain results at the current optimization step to use for initialization in the next steps, thus saving computing time. This allows for reducing the long simulation time necessary to pass the burn-in period of the Markov Chain [44] and obtain samples $\mathbf{y}^-$. We use a sample from the law derived from $U(\cdot, \mathbf{X}, \boldsymbol{\theta}_{n-1})$ to initialize the chain that samples the law derived from $U(\cdot, \mathbf{X}, \boldsymbol{\theta}_n)$. 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 $U(\cdot, \mathbf{X}, \boldsymbol{\theta}_n)$.
3. Use $\mathbf{y}^- = \mathbf{x}_{n_s}$ as a negative sample for the loss computation and update of $\boldsymbol{\theta}_n$ to $\boldsymbol{\theta}_{n+1}$.
4. Update buffer $\mathcal{B} \leftarrow \mathcal{B} \cup \{\mathbf{y}^-\}$.

This buffer allows running longer chains *across* epochs (i.e. obtaining higher-quality samples) while maintaining a low computational cost.

2.5.4. Parameters Estimation Algorithm

The final parameter estimation algorithm is as follows :

At line 4 we select a temperature at which to run the Markov chain to sample the negative configuration $\mathbf{y}^-$ ($\beta_1 = 1$, $\beta_2 = 10^{-2}$, $\beta_3 = 10^{-4}$); higher temperatures allow sample diversity, while lower ones seek points of minimal energy within the current energy model. Scaling ς computed at line 6 allows rescaling the energy to a unit variance model in order to ensure consistency in sample diversity for a set temperature. It is computed over 1000 samples $\mathbf{y} \sim \mathcal{U}(\mathcal{Y})$. Finally, for each temperature β and image $\mathbf{X}$, we set a specific buffer $\mathcal{B}_{\mathbf{X},\beta}$. The parameter K is the number of steps the Markov chain runs for every gradient descent step. In practice, we set $K = 0.02 d_c^2$ so that the number of iterations scales with the number of pixels in a cell.

Algorithm 2 we propose above provides a single procedure to estimate all parameters in $\boldsymbol{\theta}$ rather than only the weights w_e as done previously with linear programming in [23,38]. On top of avoiding over-constraining by having no hard constraints, our method also removes the need to design a procedure to sample configurations $\mathbf{y}^-$ as those are sampled from the current energy model $U(\cdot, \mathbf{X}, \boldsymbol{\theta}_n)$. Finally, the replay buffer allows for qualitative samples while maintaining a limited computational burden.

Algorithm 2: Contrastive Divergence parameters estimation.

$\mathcal{B} \leftarrow \emptyset$, $n \leftarrow 0$
while not converged **do**
  for all $(\mathbf{y}^{GT}, \mathbf{X})$ in $\mathcal{D}$ **do**
    $\beta \sim \mathcal{U}(\{\beta_1, \beta_2, \beta_3\})$ *select temperature to sample negative config.*
    $\mathbf{x}_0 \sim \mathcal{U}(\mathcal{B}_{\mathbf{X},\beta})$ with probability $p_{\mathcal{B}}$, else $\mathcal{U}(\mathcal{Y})$ *retrieve relevant buffer*
    $\varsigma \leftarrow \sqrt{\mathrm{Var}(U_{T=\infty}(\cdot, \mathbf{X}, \boldsymbol{\theta}_n))}$ *compute energy scale*
    $\mathbf{y}^- \leftarrow \mathrm{Sample}(\mathbf{x}_0, \mathbf{X}, \boldsymbol{\theta}_n, \varsigma\beta, K, 1)$ *see Algorithm 1*
    $\mathbf{y}^+ \leftarrow \mathbf{y}^{GT} + \mathcal{N}(0, \sigma^+)$
    $\Delta\boldsymbol{\theta}_n \leftarrow \nabla_{\boldsymbol{\theta}_n} \mathcal{L}(\boldsymbol{\theta}_n, \mathbf{y}^+, \mathbf{y}^-, \mathbf{X})$
    Update $\boldsymbol{\theta}_{n+1}$ with $\Delta\boldsymbol{\theta}_n$ using SGD
    $\mathcal{B}_{\mathbf{X},\beta} \leftarrow \{\mathbf{y}^-\} \cup \mathcal{B}_{\mathbf{X},\beta}$ *update the buffer*
    $n \leftarrow n + 1$
  end for
end while

2.6. Papangelou Intensity as a Confidence Score

Classical CNN-based object-detection models for object detection (such as [8,45,46]) 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 positive, $s(y) \propto p(y|\mathbf{X})$. Applying a score (or confidence) 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. This allows adapting the threshold according to the need for the application; i.e., some applications may require low false positives (high precision) while others require fewer missed detections (high recall). To assess the performance independently of the threshold selection, the Average Precision (AP) metric sums up the performance of a model as the area under the Precision-Recall curve.

Previous Point Process approaches [18,47,48] only compute simple metrics such as precision, recall, or F1 score for the configuration given by the sampling procedure, as no score is associated with each object detection.

2.6.1. Papangelou Intensity as Score

With our PP approach, we propose to introduce a scoring function, first to filter the detections given a confidence threshold and second to be able to compare our method to others using the widely used AP metric. Within the PP framework, 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 reflects it: $s(y|\hat{\mathbf{y}} \setminus \{y\}) \propto p(y|\hat{\mathbf{y}} \setminus \{y\}, \mathbf{X})$. From (3), we have that the Papangelou conditional 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 conditional intensity as a score :

$$s(y|\mathbf{y} \setminus \{y\}) = \lambda(y; \mathbf{y} \setminus \{y\}) \tag{33}$$

2.6.2. Pruning Sequence

However, the dependency of the score on the current configuration yields a complication while computing the AP: 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 $\mathbf{y}_1 = \hat{\mathbf{y}}$, for $n = 1, \dots, |\hat{\mathbf{y}}|$:

$$\mathbf{y}_{n+1} = \mathbf{y}_n \setminus \{y_n\},\ y_n = \underset{y \in \mathbf{y}_n}{\arg\min} \lambda(y; \mathbf{y}_n \setminus \{y\}), \tag{34}$$

$$s(y_n|\mathbf{y}_n \setminus \{y_n\}) = s(y_n|\mathbf{y}_{n+1}) = \lambda(y; \mathbf{y}_{n+1}). \tag{35}$$

Equation (34) provides a pruning order $y_1, \dots, y_{n(\hat{\mathbf{y}})}$ of points in $\hat{\mathbf{y}}$. This ordering allows plotting 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 (35) provides a score for each point y_n.

2.6.3. Contrastive Divergence Loss and Papangelou Intensity

On the one hand, the energy model is trained by minimizing the loss function in (31) derived from the likelihood maximization of the parameters regarding the annotated data. On the other, we evaluate the performance of the inferred configuration with the scoring method in (35) sourced from the Papangelou intensity. Here, we show that while the two are derived differently, the minimization of the loss function leads to good properties on the score function.

Here, we consider a simplified loss with only the two energy terms (as $\gamma \simeq 0$). Denoting the energy change induced by the move from configuration $\mathbf{y}$ to $\mathbf{x}$ as $\Delta U(\mathbf{y} \to \mathbf{x}) = U(\mathbf{x}) - U(\mathbf{y})$, we have :

$$\mathcal{L}(\boldsymbol{\theta}, \mathbf{y}^+, \mathbf{y}^-) = \Delta U(\mathbf{y}^- \to \mathbf{y}^+). \tag{36}$$

Similarly, the Papangelou intensity can be rewritten as such:

$$\lambda(u; \mathbf{y}) = \exp(\Delta U(\mathbf{y} \cup \{u\} \to \mathbf{y})) \tag{37}$$

Single Point Addition

Thus, for a simple negative sample $\mathbf{y}^- = \mathbf{y}^+ \cup \{u\}$ in which we add a non-valid point u to $\mathbf{y}^+$, we have:

$$\mathcal{L}(\boldsymbol{\theta}, \mathbf{y}^+, \mathbf{y}^-) = \log(\lambda(u; \mathbf{y}^+)). \tag{38}$$

This leads to the expected behavior: minimizing the loss $\mathcal{L}$ leads to minimizing the score of non-valid point u. The same stands for the removal of a valid point $y \in \mathbf{y}^+$ and maximizing its score.

Arbitrary Sequence of Moves

This is also valid for the generic case where $\mathbf{y}^-$ is generated from an arbitrary sequence of additions or removal of points (A translation/rotation/scaling can be viewed as removal and addition.) from $\mathbf{y}^+$. This defines a sequence $(\mathbf{y}_k)_{k=0,\dots,n}$ of n configurations as:

$$\forall k = 1, \dots, n,\ \mathbf{y}_k = \begin{cases} \mathbf{y}_{k-1} \setminus \{y_k\} & \text{if } y_k \in \mathbf{y}^+ \\ \mathbf{y}_{k-1} \cup \{y_k\} & \text{otherwise,} \end{cases} \tag{39}$$

with $\mathbf{y}_0 = \mathbf{y}^+$, $\mathbf{y}^- = \mathbf{y}_n$, and y_k elements of either $\mathcal{S} \times \mathcal{M}$ or $\mathbf{y}^+$. Without loss of generality, we can reorder the sequence to match the pruning order defined in (34). The energy change for one move is given as:

$$\Delta U(\mathbf{y}_{k-1} \to \mathbf{y}_k) = \begin{cases} \log(\lambda(y_k; \mathbf{y}_{k-1} \setminus \{y_k\})) & \text{if } y_k \in \mathbf{y}^+ \\ -\log(\lambda(y_k; \mathbf{y}_{k-1})) & \text{otherwise.} \end{cases} \tag{40}$$

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:

$$\mathcal{L} = \sum_{y_k \notin \mathbf{y}^+} \log(\underbrace{\lambda(y_k; \mathbf{y}_{k-1})}_{\text{(a)}}) - \sum_{y_k \in \mathbf{y}^+} \log(\underbrace{\lambda(y_k; \mathbf{y}_{k-1} \setminus \{y_k\})}_{\text{(b)}}). \tag{41}$$

By ordering the $y_k, \mathbf{y}_k$ to match the pruning order in (34) each $\lambda(y_k; \dots)$ can be matched to their respective score:

- (41)(a) corresponds to non-valid points added to $\mathbf{y}^+$, their score is minimized as the loss is decreased;
- (41)(b) corresponds to valid points removed from $\mathbf{y}^+$. Their score is increased as the loss is minimized.

With this, we showed that minimization of the loss at a configuration level leads to the expected results on object scores.

While this score function allows taking into account the interaction between objects that the PP model introduces, the decomposable nature of the PP (see (5)) allows for the interpretability of the results as demonstrated in Section 3.4.

2.7. Method Summary

In this section, we proposed a model for object detection that combines CNN architectures with a Point Process (PP) framework. Before applying our approach to real data, we shall summarize its function here. Considering configurations of objects cy, image $\mathbf{X}$, and parameters $\boldsymbol{\theta}$, we look to build an energy model $U(\mathbf{y}, \mathbf{X}, \boldsymbol{\theta})$ such that the minimum energy point of it corresponds to the ground truth configuration for that image $\mathbf{y}^{GT}$. As such, we build U to be a combination of multiple energy terms (Equation (5)), with prior terms that encode the interaction model of our objects (where high energies repel unwanted configurations and lower ones favor others), and data terms that we build from the features from a CNN (Section 2.3.1). We refer the reader to Figure 5 for an overall view of the model pipeline. Within this framework, we can combine the information from the image with the prior knowledge of the configurations.

To estimate the parameters $\boldsymbol{\theta}$, we propose to use Contrastive Divergence. In short, we initialize the parameters at random and alternate between sampling low-energy configurations with the current parameters—that are nowhere close to the ground truth at the beginning of the procedure—and updating the parameters to increase the energy of these samples while maintaining low energy for the ground truth (Section 2.5). This allows us to estimate any differentiable parameters in the model, while previous methods would often become over-constrained and could only estimate linear parameters.

Now, given parameters $\boldsymbol{\theta}$ are set, we want to estimate the configuration of an unseen image $\mathbf{X}$. This is achieved by sampling the configuration $\mathbf{y}^*$ that minimizes the energy through a Jump Diffusion mechanism. Alternatively, proposing to add or remove points in the configuration guided by the energy maps the CNN provides (Section 2.4.1) and lowering the energy locally at a fixed number of points with gradient descent (Section 2.4.3). The latter can be applied in parallel over the image while focusing on patches of the image with higher expected object density (Section 2.4.2).

Lastly, we propose a metric based on the Papangelou intensity that allows us to compare our approach to others and takes into account the object interactions in the likelihood of detection (Section 2.6).

3. Results

3.1. Comparison to PP Based on Contrast Measures

To demonstrate the need for learned likelihood measures for our data, we evaluate a few contrast measures found in the literature [20,24] in the task of classification of objects and non-objects (Figure 8). Table 1 formulates these measures by denoting μ_y, σ_y^2 the empirical mean and variance of pixels inside shape y; μ_s, σ_s^2 mean and variance on shape contour s; n_y the normal vector of contour s, and $\nabla \mathbf{X}$ the image gradient.

To assert the viability of such measures for these data, we compute a score $s(y) = \exp(-V_c(y)$ for proposed detections y (both ground truth and randomly scattered rectangles). The true objects represent 1% of this experiment's dataset (A higher proportion of true objects would allow high average precision for useless estimators predicting a constant value.). Here we consider around 32k object proposals (i.e., 313 true objects, and 31,300 non-objects). A well-suited measure $s(y)$ should allow classifying easily between ground truth objects and random, non-valid objects. We plot Precision-Recall curves for several of those contrast measures in Figure 8 both for a sample image of $DOTA_{0.5}$ and a synthetic highly contrasted image. The Average Prevision (AP) values are reported in Table 1. We compare with the proposed detection energy given by $V_{pos} + V_a + V_b + V_\alpha$.

Table 1. Various contrast measures from literature and Average Precision (AP) computed on the DOTA image (AP_{real}) or synthetic image ($AP_{synthetic}$) from Figure 8.

Measure	V_c Formulation	AP_{real}	$AP_{synthetic}$
CNN output (ours)	$V_{pos}(y, \mathbf{X}) \sum_\kappa V_\kappa(y, \mathbf{X})$	0.99	0.88
t-test [20]	$\frac{\lvert\mu_y - \mu_s\rvert}{\sqrt{\frac{\sigma_y^2}{n_y} + \frac{\sigma_s^2}{n_s}}}$	0.13	0.99
Gradient [24]	$\int \frac{n_y(t) \cdot \nabla \mathbf{X}(s(t))}{\sqrt{\lvert\nabla \mathbf{X}(s(t))\rvert^2 + \epsilon}} dt$	0.27	0.99
Constant	1.0	0.01	0.01

The proposed measure, based on CNN output, outperforms classical contrast-based measures, as the CNN has learned to differentiate between relevant and irrelevant contrasted objects (Irrelevant objects include, for instance, AC units—a highly contrasted, car-sized object—or some pavement features.).

3.2. Models

In this article, we show results on two CNN-based models and two PP models. A Python implementation of our models is available at https://github.com/Ayana-Inria/PP-EBM (accessed on 31 January 2024).

- 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$_m$: PP model with minimal inferred parameters: Pre-set priors (i.e., with manually set parameters), only the weight vector w_e are estimated with the CD method. The values of manually set parameters are detailed in Table 2.
- CNN-PP$_w$: PP model with the whole parameter vector $\widehat{\theta}$ (which encompasses the weights w_e and priors' parameters) estimated with the CD method.
- BBA-Vec. and YOLOV5-OBB: Lastly, we compare all our models above with BBA-Vec. from [8] and YOLOV5-OBB from [46]. These models are trained on the same data as the above-mentioned models.

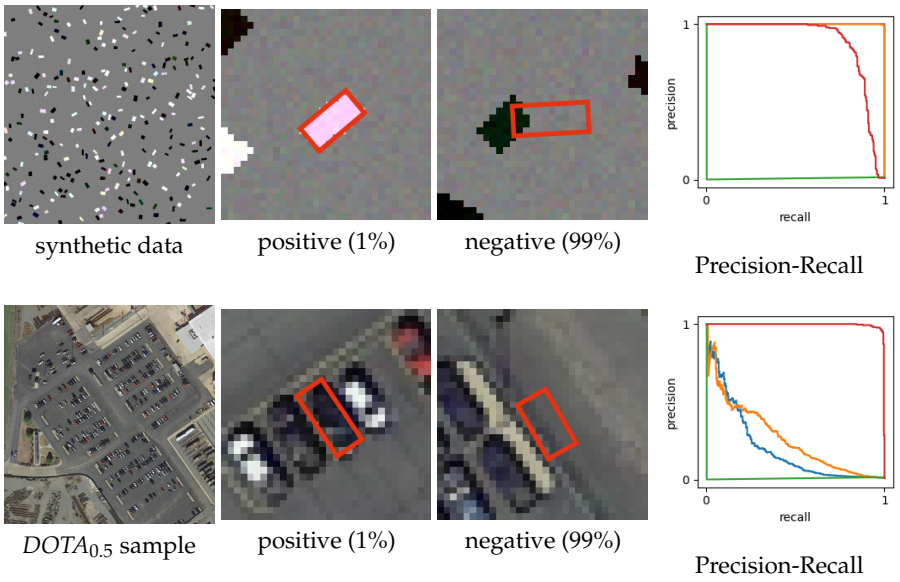


Figure 8. Comparing contrast measures on synthetic (**first line**) and real data (**second line**). Left: Positive and negative samples represent, respectively, 1% and 99% of the samples for which the metric is computed. Right: Precision-Recall curves for the measures proposed in Table 1; blue: *t*-test [20], orange: image gradient [24], red (our data term), green: constant value.

Table 2. Values for manually set parameters. Some energy terms such as V_a have no parameters.

Energy Term	Parameter	Value
V_{pos}	t_{pos}	0
V_a*	-	-
V_b*	-	-
V_α*	-	-
V_{align}	-	-
V_{ovrlp}	t_{ovrlp}	0
V_{repls}	t_{repls}	0
V_{attrc}	t_{attrc}	0
$V_{jntAR,car}$	μ_{ratio} μ_{area} σ_{ratio} σ_{area}	0.46 42 0.1 20
$V_{jntAR,truck}$	μ_{ratio} μ_{area} σ_{ratio} σ_{area}	0.23 123 0.1 20

* Mark energy terms V_κ, $\kappa \in \{a, b, \alpha\}$.

3.3. Results on Remote-Sensing Data

The models are evaluated on a test split of the $DOTA_{0.5}$ dataset (for quantitative and qualitative results) and on *ADS* (for qualitative results as these data are not annotated). Metrics are computed by matching the object proposals and ground truth using the IOU (Intersection Over Union) with a 0.25 threshold. The Average Precision (AP) metrics (which are threshold independent) are shown in Table 3 and are derived from Precision-Recall (PR) curves shown in Figure 9. We show detection results in Figure 10 (lines 1–2 for $DOTA_{0.5}$ and line 3 for $DOTA_{0.5}$+*noise*), with a score threshold set for each model to the one maximizing

the F1 score. The *ADS* data inference results in Figure 11 are obtained with the models trained on the $DOTA_{0.5}$ training set.

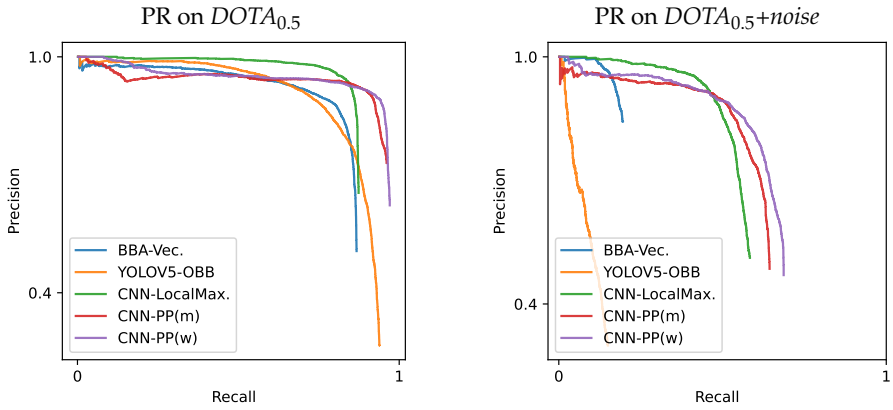


Figure 9. Precision-Recall (PR) curves for the models listed in Table 3.

Table 3. Average Precision (AP) values. AP_0 and AP_N correspond, respectively, to the AP on the $DOTA_{0.5}$ and $DOTA_{0.5}$+*noise* datasets. Precision-recall curves from which the AP is derived are shown in Figure 9.

Method	AP_0	AP_N
BBA-Vec.	0.82	0.19
YOLOV5-OBB	0.86	0.10
CNN-LocalMax.	0.86	0.55
CNN-PP$_m$	**0.91**	**0.58**
CNN-PP$_w$	**0.92**	**0.62**

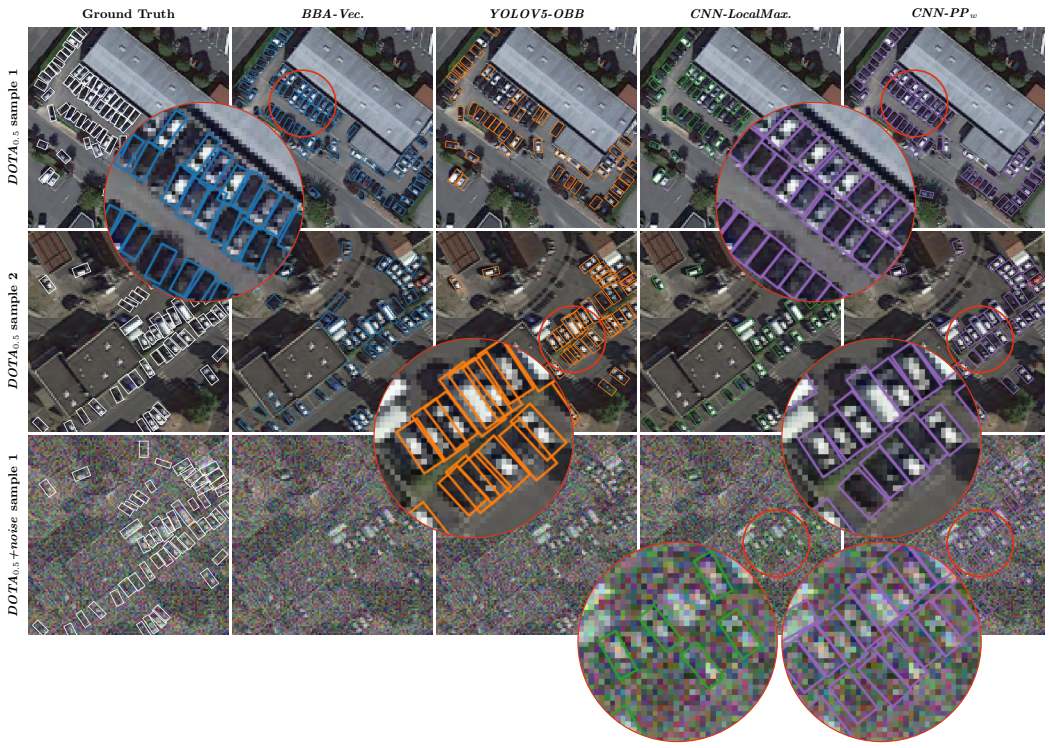


Figure 10. Samples of detection on the test dataset. The score threshold (to not display low-score objects) is set to maximize the *F*1 score for each model. The first two lines correspond to samples from $DOTA_{0.5}$ while the third one corresponds to $DOTA_{0.5}$+*noise*.

Figure 11. Model applied to *ADS* data. The first column shows the input image, as there are no ground truth annotations for this dataset.

3.3.1. Computational Complexity

Our two PP-based methods—running on an *Nvidia Quadro RTX 8000* Graphics Processing Unit—execute in 300 s on average, for 16 k parallel iterations (equivalent to 77 k sequential iterations) for one image. With an efficient implementation, considering a density of 1.9×10^{-3} (95th percentile of the observed object densities (i.e., 95% of observed object densities are below this value.)), 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 B. As a point of comparison, Transformer models for image classification range from 5×10^6 to 1.8×10^7 operations (see [49]). The 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.

3.4. Results Interpretability

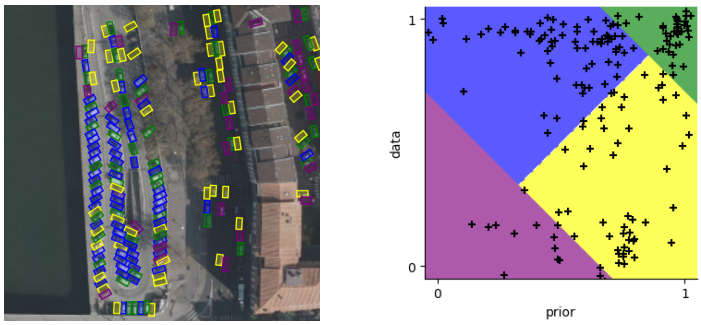
Due to the decomposition of the total energy into energy terms introduced in (5), the object score can be decomposed similarly:

$$s(y|\mathbf{y} \setminus \{y\}) = \prod_{e \in \tilde{\xi}} s_e(y|\mathbf{y} \setminus \{y\}) \tag{42}$$

with $s_e(y|\mathbf{y} \setminus \{y\}) = \exp(w_e \Delta V_e(\mathbf{y} \to \mathbf{y} \setminus \{y\}))$, the Papangelou intensity obtained by considering the single energy term e. This allows viewing the contribution of each component. Moreover, we propose grouping these contributions into the data and prior contributions to, respectively, obtain s_{data} and s_{prior} such that the final score is a product of the two: $s(y|\mathbf{y} \setminus \{y\}) = s_{data}(y) s_{prior}(y|\mathbf{y} \setminus \{y\})$.

In Figure 12, we illustrate how the two components of the score s can help analyze the results. Here we manually cluster the results into 4 basic categories: green objects correspond to detection with high prior and data scores (high s, top 25%), while blue detections have a higher data contribution ($s_{prior} < s_{data}$). The yellow detections correspond to objects with lower data scores ($s_{prior} > s_{data}$), often located in ambiguous locations.

Purple objects have a low total score s (i.e., low confidence, bottom 15%). These detections are often pruned out when using a score threshold (as used in Figures 10 and 11). This allows for identifying where the model is confident, either due to the data from the image or the interactions with nearby objects. One can access even more detailed information by looking into the individual scores s_e for each energy term V_e.



(**a**) Inferred configuration (**b**) Color correspondence

Figure 12. Inferred configuration on an ADS data sample (**a**), colored according to their prior/data scores plotted in (**b**): yellow $s_{prior} > s_{data}$; blue $s_{prior} < s_{data}$; purple low s; green high s. Each point in (**b**) corresponds to a detection in s_{data},s_{prior} space (log values scaled to $[0,1]$).

4. Discussion

4.1. Quantitative and Qualitative Results

In Figure 10, we observe that despite noisy ground truth labels—a potential cause for over-constraining issues when using linear programming (see Section 2.5)—our models infer more regular configurations that align better with the physical constraints of the objects of interest. Sample 1 in Figure 10 BBA-Vec. struggles to properly align and detect all the objects. In sample 2, model YOLOV5-OBB shows patches of objects with no overlap regularization. Meanwhile, our model outputs regularized configurations even in these dense areas. Figure 10 (line 3) and Figure 11 showcase the robustness of our PP-based models when tested on noisy data or areas of limited information, such as shaded areas. The inference results on the *ADS* dataset, obtained with models trained on the $DOTA_{0.5}$ training set, highlight the model's ability to generalize to unseen, albeit similar, data.

Comparison between CNN-PP$_m$ and CNN-PP$_w$ against CNN-LocalMax. in Figure 10 and Table 3 reveals the improvement brought by the PP-based approach, as the performance of the CNN alone (CNN-LocalMax.) falls short of our proposed method especially when confronted with noisy data.

Lastly, the comparable performances of CNN-PP$_m$ and CNN-PP$_w$, both qualitatively and quantitatively, highlight the efficacy of the CD method in inferring model parameters. This approach significantly reduces the need for manual parameter tuning (more specifically, prior parameters), where traditional PP methods, such as linear programming, fall short.

4.2. Limitations

Our method, while promising, faces several limitations that warrant consideration. First, the computational complexity involved in sampling a PP can be significant, particularly with dense configurations of objects. To address this, we could explore approximate sampling methods, although this might come at the expense of result accuracy, for instance, by implementing faster annealing techniques or discretizing the state space [48].

Second, while the model we present is tailored to a specific type of object, the underlying tools are adaptable to various object types. First, the PP framework allows for a wide diversity of objects to be modeled, thanks to the versatility of marks. Moreover, the energy formulation of our model allows for easy integration or removal of interaction models and priors as simple energy functions. For instance, we could incorporate classification as a

mark in the Point Process model, utilizing features from the CNN or a prior law based on the geometric features of the objects (e.g., distinguishing trucks from cars based on length).

Lastly, our method relies on annotated data for training. However, energy models offer compositional flexibility; one potential avenue is to train the data terms of the model on annotated images while utilizing synthetic datasets (devoid of visual components) for prior terms training. This approach could help mitigate the reliance on annotated data and improve generalization to diverse scenarios.

5. Conclusions

In this paper, we present a method that integrates interaction models into object-detection algorithms, leveraging deep convolutional neural networks. We address challenges inherent in satellite imagery, such as atmospheric disturbances and limited resolution, by incorporating prior knowledge about object arrangements.

While CNN-based methods excel at pattern extraction, they struggle with learning object-to-object interactions without complex attention mechanisms. Conversely, Point Process methods offer a promising alternative, addressing both object likelihoods and arrangement coherence. However, existing approaches relying on contrast measures have limitations.

Our method combines CNN pattern extraction with the Point Process framework, simplifying contrast measure computations into efficient energy maps. By leveraging pre-computed potential maps, we enhance Point Process sampling methods, improving exploration efficiency within the state space.

We depart from conventional parameter estimation approaches for Point Process methods like linear programming and adopt Contrastive Divergence to estimate energy weights and internal parameters automatically. This improves accuracy and reduces reliance on manual parameter tuning.

To facilitate model comparison, we introduce a novel scoring function based on the Papangelou conditional intensity, providing a more comprehensive evaluation metric that takes into account interactions and allows for the explainability of results thanks to the easy decomposition of the energy model.

Experimental results on diverse datasets demonstrate our method's effectiveness, particularly in noisy environments. Our model's ability to generalize to new data showcases its robustness and potential for practical applications.

In conclusion, our method offers a promising approach to object detection, combining the strengths of CNNs with Point Process models. It shows potential for improving accuracy and efficiency in satellite imagery analysis or other applications where the priors are strong: e.g. object tracking (priors on dynamics) or road extraction (geometrical priors).

Author Contributions: Conceptualization, M.O. and J.Z.; Data curation, J.M.; Formal analysis, J.M.; Funding acquisition, M.O. and J.Z.; Investigation, J.M.; Methodology, J.M., M.O., and J.Z.; Project administration, J.Z.; Resources, J.M., M.O., and J.Z.; Software, J.M.; Supervision, M.O. and J.Z.; Validation, J.M., M.O., and J.Z.; Visualization, J.M.; Writing—original draft, J.M.; Writing—review and editing, J.M., M.O., and J.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by BPI France via the LiChIE project.

Data Availability Statement: The DOTA dataset is available at https://captain-whu.github.io/DOTA/dataset.html (accessed on 31 January 2024). The ADS data are not publicly available, due to industrial property restrictions.

Acknowledgments: Thanks to the OPAL infrastructure from Université Côte d'Azur for providing computational resources and support.

Conflicts of Interest: J. Zerubia and J. Mabon are employed by Inria. M. Ortner is employed by Airbus Defense and Space. J. Mabon's PhD grant is part of the LiChIE contract, financed by BPI France, in partnership with Airbus Defense and Space. The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

ADS	Airbus Defense and Space
AP	Average Precision
CD	Contrastive Divergence
CNN	Convolutional Neural Network
EBM	Energy-Based Model
FCN	Fully Convolutional Network
GT	Ground Truth
PP	Point Process
(RJ)MCMC	(Reversible Jump) Monte Carlo Markov Chain

Appendix A. Proofs

Appendix A.1. Markovianity of the Point Process

We denote $\sim$ the interaction relation $y \sim y' \Leftrightarrow ||y - y'|| < d_{max}$ resulting in neighborhood $\mathcal{N}_y^{\mathbf{y}}$. We need to determine the relation $\underset{m}{\sim}$ (with associated neighborhood $\partial_{\{y\}}^{\mathbf{y}}$) for which the Point Process in Markovian. Given Definition 1, 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):

$$\frac{h(\mathbf{y} \cup \{u\})}{h(\mathbf{y})} = \exp(U(\mathbf{y}, \mathbf{X}) - U(\mathbf{y} \cup \{u\})) \tag{A1}$$

$$= \exp\left(\sum_{y \in \mathbf{y}} \left[V\left(y, \mathbf{X}, \mathcal{N}_y^{\mathbf{y}}\right) - V\left(y, \mathbf{X}, \mathcal{N}_y^{\mathbf{y} \cup \{u\}}\right)\right] - V\left(u, \mathbf{X}, \mathcal{N}_u^{\mathbf{y} \cup \{u\}}\right)\right) \tag{A2}$$

$$= \exp\left(\sum_{y \in \mathcal{N}_u^{\mathbf{y}}} \left[V\left(y, \mathbf{X}, \mathcal{N}_y^{\mathbf{y}}\right) - V\left(y, \mathbf{X}, \mathcal{N}_y^{\mathbf{y} \cup \{u\}}\right)\right] - V\left(u, \mathbf{X}, \mathcal{N}_u^{\mathbf{y}}\right)\right). \tag{A3}$$

The above follows from the fact that $y \notin \mathcal{N}_u^{\mathbf{y}} \implies \mathcal{N}_y^{\mathbf{y}} = \mathcal{N}_y^{\mathbf{y} \cup \{u\}}$ (as $\sim$ 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 $y \underset{m}{\sim} y' \Leftrightarrow \exists y'' \in \mathbf{y}, y \sim y'' \sim y'$ is impractical as it depends on $\mathbf{y}$. With $\sim$ derived from maximum distance d_{max}, the following relation is sufficient to have Markovianity w.r.t. relation $\underset{m}{\sim}$:

$$y \underset{m}{\sim} y' \Leftrightarrow ||y - y'|| < 2d_{max}. \tag{A4}$$

Appendix A.2. Existence of Z

We denote $f = \exp(-U(\mathbf{y}, \mathbf{X}, \boldsymbol{\theta}))$ the unnormalized density from which h is derived in (1). The Ruelle condition [50] ensures the Point Process is stable (i.e. the density can be normalized) given the following condition :

Condition A.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 exists $M \leq 1$ such that*

$$f(\mathbf{y}) \leq M^{n(\mathbf{y})}, \forall \mathbf{y} \in \mathcal{Y}. \tag{A5}$$

From the above, it follows that the unnormalized density can be bounded:

$$\int_{\mathbf{y} \in \mathcal{Y}} f(\mathbf{y}) d\mu(\mathbf{y}) \leq \sum_{n=0}^{\infty} \frac{M^n \nu(\mathcal{S})}{n!} = \exp(M\nu(\mathcal{S})), \tag{A6}$$

with μ the Poisson Point Process density and ν the associated measure of space (e.g. Lebesgue measure).

It is enough to show the following for Condition A.1 to be true:

$$f(\mathbf{y} \cup \{y\}) \leq Mf(\mathbf{y}),\ \forall \mathbf{y} \in \mathcal{Y},\ \forall u \in \mathcal{S}. \tag{A7}$$

By construction, the energy of a single point is bounded (see (5) and Section 2.3.2.1), we have:

$$\exists A > 0,\ |V(y, \mathbf{X}, \mathcal{N}_y^{\mathbf{y}}, \boldsymbol{\theta})| < A,\ \forall y \in \mathbf{y},\ \forall \mathbf{y} \in \mathcal{Y}. \tag{A8}$$

Then, from the results of (A3), we have:

$$\frac{f(\mathbf{y} \cup \{u\})}{f(\mathbf{y})} \leq \exp\left(2A|\mathcal{N}_y^{\mathbf{y}}| + A\right). \tag{A9}$$

In practice, the number of points in a cell is bounded by $n_{c,max}$. Thus, the maximum number of points in a neighborhood is bounded by $|\mathcal{N}_y^{\mathbf{y}}| \leq 4n_{c,max} - 1$. We can then find a bound that satisfies (A7) as :

$$\frac{f(\mathbf{y} \cup \{u\})}{f(\mathbf{y})} \leq \exp(A(8n_{c,max} - 1)). \tag{A10}$$

Appendix B. Complexity

From Algorithm 1, we derive the theoretical computation cycle cost of one sampling loop over a single cell:

$$\mathbf{F}_c \simeq 3.8 \times 10^2 + 4.9 \times 10^6 \lambda + 5.9 \times 10^8 \lambda^2, \tag{A11}$$

where λ is the density of objects in the image (For details on the derivation see [51].) To obtain the operations per pixel (ops/px/iter), we compute $\mathbf{F}_c / d_c^2$. In practice, we use $n_s = 77000$ iterations of the Markov chain, so the total number of operations (ops/px) is $n_s \mathbf{F}_c / d_c^2$. 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 A2. For these density values, we compute the cost per pixel with diffusion on whole cells, on single objects, and without diffusion in Table A1. By default, the model performs diffusion over a whole cell c. Alternatively, it can do diffusion upon a single point y in cell c for a lower cost. Finally, we report the cost without the diffusion mechanism (although the results are of lesser quality for the same number of iterations).

Table A1. Number of theoretical operations depending on object density.

	λ [(a)]	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

[(a)] Values of λ reported in Table A2.

Table A2. Values for density lambda .

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}

Appendix C. FCN for Small Objects in Large Images

For our application, we need a Fully Convolutional Network (FCN) [52] capable of responding to small objects in large images.

Definition A1. *A Fully Convolutional Network (FCN) is a type of CNN architecture making use of locally connected layers such as convolution pooling or upsampling [53].*

A straightforward approach to learning a position probability map $\widehat{\mathbf{Z}}_{pos}$ would be to directly infer a heatmap of centers, i.e., obtained by applying a Gaussian blur on a binary map of object centers (Figure A1b). However, these maps are sparse (non-zeros probabilities are under-represented), and at inference, we observe some connectivity added in between blobs, making the separation of instances more difficult (see Figure A1b).

We circumvent this problem by learning a proxy task: The model is first trained to infer a map of 2D unit-length vectors $\mathbf{H}_\nabla$ that point towards the closest instance center (Figure A1c) similar to [54]. We then apply the divergence operator, as object centers now correspond to convergence points in this vector field [55] (Figure A1d):

$$\widehat{\mathbf{Z}}_{pos} = a.\mathrm{div}\left(\widehat{\mathbf{H}}_\nabla\right) + b. \tag{A12}$$

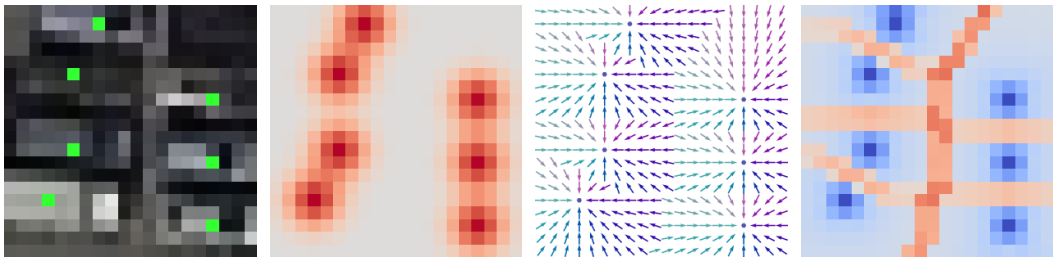
The backbone architecture corresponds to a Unet [28], which outputs for an image $\mathbf{X}$ a tensor $F(\mathbf{X})$ with N_F features per pixel. From that, feature representation is extracted $\widehat{\mathbf{Z}}_{pos}$ and the $\widehat{\mathbf{Z}}_\kappa m \; \forall m \in \{a, b, \alpha\}$.

In our experiments, we train the Unet to minimize the following cost function for position term [55] and mark terms:

$$\mathcal{L}_{pos}\left(\mathbf{X}, \mathbf{y}^{GT}\right) = \mathrm{MSE}\left(\widehat{\mathbf{H}}_\nabla, \mathbf{H}_\nabla\right) + \mathrm{BCE}\left(\widehat{\mathbf{Z}}_{pos}, \mathbf{Z}_{pos}\right), \tag{A13}$$

$$\mathcal{L}_\kappa\left(\mathbf{X}, \mathbf{y}^{GT}\right) = \frac{1}{|\mathcal{S}_d|} \sum_{p \in \mathcal{S}_d} \mathrm{CE}\left(\mathrm{Softmax}\, \widehat{\mathbf{Z}}_\kappa[p], \widehat{\mathbf{Z}}_\kappa[p]\right), \tag{A14}$$

with MSE the Mean Squared Error, BCE the Binary Cross Entropy, and CE the Cross Entropy. The tensors $\mathbf{H}_\nabla$, $\mathbf{Z}_{pos}$, and $\widehat{\mathbf{Z}}_\kappa$ are, respectively, the vector, position, and mark tensors built from the annotated ground truth.



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

Figure A1. Closeup image with centers overlay (**a**), centers heatmap (**b**), vector field (**c**), and divergence (**d**) (red > 0, blue < 0).

References

1. Zou, Z.; Chen, K.; Shi, Z.; Guo, Y.; Ye, J. Object Detection in 20 Years: A Survey. *Proc. IEEE* **2023**, *111*, 257–276. https://doi.org/10.1109/JPROC.2023.3238524.
2. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In *Advances in Neural Information Processing Systems (NIPS)*; Cortes, C., Lawrence, N., Lee, D., Sugiyama, M., Garnett, R., Eds.; MIT Press: Montréal, Canada, 2015; Volume 28.
3. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788. https://doi.org/10.1109/CVPR.2016.91.
4. Lin, T.Y.; Goyal, P.; Girshick, R.; He, K.; Dollar, P. Focal Loss for Dense Object Detection. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017. https://doi.org/10.1109/ICCV.2017.324.
5. Li, Y.; Xing, Y.; Wang, Z.; Xiao, T.; Song, Q.; Li, W.; Wang, J. A Framework of Maximum Feature Exploration Oriented Remote Sensing Object Detection. *IEEE Geosci. Remote Sens. Lett.* **2023**, *20*, 6001505. https://doi.org/10.1109/LGRS.2022.3228689.
6. Yao, Y.; Cheng, G.; Wang, G.; Li, S.; Zhou, P.; Xie, X.; Han, J. On Improving Bounding Box Representations for Oriented Object Detection. *IEEE Trans. Geosci. Remote Sens.* **2022**, *61*, 5600111. https://doi.org/10.1109/TGRS.2022.3231340.
7. Zhao, T.; Liu, N.; Celik, T.; Li, H.C. An Arbitrary-Oriented Object Detector Based on Variant Gaussian Label in Remote Sensing Images. *IEEE Geosci. Remote. Sens. Lett.* **2021**, *19*, 8013605. https://doi.org/10.1109/LGRS.2021.3087492.
8. Yi, J.; Wu, P.; Liu, B.; Huang, Q.; Qu, H.; Metaxas, D. Oriented Object Detection in Aerial Images with Box Boundary-Aware Vectors. In Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV), Waikoloa, HI, USA, 3–8 January 2021; pp. 2149–2158. https://doi.org/10.1109/WACV48630.2021.00220.
9. Cheng, G.; Wang, J.; Li, K.; Xie, X.; Lang, C.; Yao, Y.; Han, J. Anchor-Free Oriented Proposal Generator for Object Detection. *IEEE Trans. Geosci. Remote Sens.* **2022**, *60*, 3183022. https://doi.org/10.1109/TGRS.2022.3183022.
10. Yang, Y.; Tang, X.; Cheung, Y.M.; Zhang, X.; Liu, F.; Ma, J.; Jiao, L. AR^{2}Det: An Accurate and Real-Time Rotational One-Stage Ship Detector in Remote Sensing Images. *IEEE Trans. Geosci. Remote. Sens.* **2021**, 60, 5605414. https://doi.org/10.1109/TGRS.2021.3092433.
11. LaLonde, R.; Zhang, D.; Shah, M. 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), Salt Lake City, UT, USA, 18–23 June 2018; pp. 4003–4012. https://doi.org/10.1109/CVPR.2018.00421.
12. Corsel, C.W.; van Lier, M.; Kampmeijer, L.; Boehrer, N.; Bakker, E.M. Exploiting Temporal Context for Tiny Object Detection. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision Workshops (WACVW), Waikoloa, HI, USA, 2–7 January 2023; pp. 1–11. https://doi.org/10.1109/WACVW58289.2023.00013.
13. Cheng, G.; Li, Q.; Wang, G.; Xie, X.; Min, L.; Han, J. SFRNet: Fine-Grained Oriented Object Recognition via Separate Feature Refinement. *IEEE Trans. Geosci. Remote Sens.* **2023**, *61*, 5610510. https://doi.org/10.1109/TGRS.2023.3277626.
14. LeCun, Y.; Bengio, Y.; Hinton, G. Deep Learning. *Nature* **2015**, *521*, 436–444. https://doi.org/10.1038/nature14539.
15. Zeng, Q.; Ran, X.; Zhu, H.; Gao, Y.; Qiu, X.; Chen, L. Dynamic Cascade Query Selection for Oriented Object Detection. *IEEE Geosci. Remote Sens. Lett.* **2023**, *20*, 6008605. https://doi.org/10.1109/LGRS.2023.3304023.
16. Lu, X.; Sun, X.; Diao, W.; Mao, Y.; Li, J.; Zhang, Y.; Wang, P.; Fu, K. Few-Shot Object Detection in Aerial Imagery Guided by Text-Modal Knowledge. *IEEE Trans. Geosci. Remote Sens.* **2023**, *61*, 5604719. https://doi.org/10.1109/TGRS.2023.3250448.
17. Lieshout, M.C.V. *Markov Point Processes and Their Applications*; Imperial College Press: London, UK, 2000.
18. Descombes, X. Multiple Objects Detection in Biological Images Using a Marked Point Process Framework. *Methods* **2017**, *115*, 2–8. https://doi.org/10.1016/j.ymeth.2016.09.009.
19. Verdié, Y.; Lafarge, F. Detecting Parametric Objects in Large Scenes by Monte Carlo Sampling. *Int. J. Comput. Vis.* **2014**, *106*, 57–75. https://doi.org/10.1007/s11263-013-0641-0.
20. Lacoste, C.; Descombes, X.; Zerubia, J. Point Processes for Unsupervised Line Network Extraction in Remote Sensing. *IEEE Trans. Pattern Anal. Mach. Intell.* **2005**, *27*, 1568–1579. https://doi.org/10.1109/TPAMI.2005.206.
21. Ortner, M.; Descombes, X.; Zerubia, J. A Marked Point Process of Rectangles and Segments for Automatic Analysis of Digital Elevation Models. *IEEE Trans. Pattern Anal. Mach. Intell.* **2008**, *30*, 105–119. https://doi.org/10.1109/TPAMI.2007.1159.
22. Perrin, G.; Descombes, X.; Zerubia, J. Tree Crown Extraction Using Marked Point Processes. In Proceedings of the 12th European Signal Processing Conference (EUSIPCO), Vienna, Austria, 6–10 September 2004; pp. 2127–2130.
23. Craciun, P.; Ortner, M.; Zerubia, J. 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), Waikoloa, HI, USA, 6–9 January 2015; pp. 177–184. https://doi.org/10.1109/WACV.2015.31.
24. Kulikova, M.S.; Jermyn, I.H.; Descombes, X.; Zhizhina, E.; Zerubia, J. Extraction of Arbitrarily-Shaped Objects Using Stochastic Multiple Birth-and-Death Dynamics and Active Contours. In Proceedings of the Computational Imaging VIII, San Jose, CA, USA, 18–19 January 2010, Volume 7533, pp. 58–64.
25. Xia, G.S.; Bai, X.; Ding, J.; Zhu, Z.; Belongie, S.; Luo, J.; Datcu, M.; Pelillo, M.; Zhang, L. 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), Salt Lake City, UT, USA, 18–23 June 2018; pp. 3974–3983. https://doi.org/10.1109/CVPR.2018.00418.
26. LeCun, Y.; Chopra, S.; Hadsell, R.; Ranzato, M.; Huang, F.J. A Tutorial on Energy-Based Learning. *Predict. Struct. Data* **2006**, *1*, 59.

27. Huang, Z.; Li, W.; Xia, X.G.; Tao, R. A General Gaussian Heatmap Label Assignment for Arbitrary-Oriented Object Detection. *IEEE Trans. Image Process.* **2022**, *31*, 1895–1910. https://doi.org/10.1109/TIP.2022.3148874.
28. Ronneberger, O.; Fischer, P.; Brox, T. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*; Navab, N., Hornegger, J., Wells, W.M., Frangi, A.F., Eds.; Lecture Notes in Computer Science; Springer: Munich, Germany, 2015; pp. 234–241. https://doi.org/10/gcgk7j.
29. Grathwohl, W.; Wang, K.C.; Jacobsen, J.H.; Duvenaud, D.; Norouzi, M.; Swersky, K. Your Classifier Is Secretly an Energy Based Model and You Should Treat It like One. In Proceedings of the International Conference on Learning Representations (ICLR), New Orleans, IL, USA, 6–9 May 2019.
30. Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; DeVito, Z.; Lin, Z.; Desmaison, A.; Antiga, L.; Lerer, A. Automatic Differentiation in PyTorch. In Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS), Long Beach, CA, USA, 9 May 2017. https://openreview.net/forum?id=BJJsrmfCZ (accessed 8 March 2024).
31. Green, P.J. Reversible Jump Markov Chain Monte Carlo Computation and Bayesian Model Determination. *Biometrika* **1995**, *82*, 711–732. https://doi.org/10/bt2s2t.
32. Metropolis, N.; Rosenbluth, A.W.; Rosenbluth, M.N.; Teller, A.H.; Teller, E. Equation of State Calculations by Fast Computing Machines. *J. Chem. Phys.* **1953**, *21*, 1087–1092. https://doi.org/10.1063/1.1699114.
33. Grenander, U.; Miller, M.I. Representations of Knowledge in Complex Systems. *J. R. Stat. Soc. Ser. B: Stat. Methodol.* **1994**, *56*, 549–581. https://doi.org/10.1111/j.2517-6161.1994.tb02000.x.
34. Green, P.J.; Hastie, D.I. Reversible Jump MCMC. *Genetics* **2009**, *155*, 1391–1403.
35. Verdié, Y.; Lafarge, F. Efficient Monte Carlo Sampler for Detecting Parametric Objects in Large Scenes. In Proceedings of the European Conference on Computer Vision (ECCV), Florence, Italy, 7–13 October 2012; pp. 539–552. https://doi.org/10.1007/978-3-642-33712-3_39.
36. Miller, M.; Grenander, U.; OSullivan, J.; Snyder, D. Automatic Target Recognition Organized via Jump-Diffusion Algorithms. *IEEE Trans. Image Process.* **1997**, *6*, 157–174. https://doi.org/10.1109/83.552104.
37. Lafarge, F.; Gimel'farb, G.; Descombes, X. Geometric Feature Extraction by a Multimarked Point Process. *IEEE Trans. Pattern Anal. Mach. Intell.* **2010**, *32*, 1597–1609. https://doi.org/10.1109/TPAMI.2009.152.
38. Yu, Q.; Medioni, G. Multiple-Target Tracking by Spatiotemporal Monte Carlo Markov Chain Data Association. *IEEE Trans. Pattern Anal. Mach. Intell.* **2009**, *31*, 2196–2210. https://doi.org/10.1109/TPAMI.2008.253.
39. Hinton, G. Training Products of Experts by Minimizing Contrastive Divergence. *Neural Comput.* **2002**, *14*, 1771–1800. https://doi.org/10.1162/089976602760128018.
40. Teh, Y.W.; Welling, M.; Osindero, S.; Hinton, G.E. Energy-Based Models for Sparse Overcomplete Representations. *J. Mach. Learn. Res.* **2003**, *4*, 1235–1260.
41. Du, Y.; Mordatch, I. Implicit Generation and Modeling with Energy Based Models. In Proceedings of the Advances in Neural Information Processing Systems (NeurIPS), Wallach, H., Larochelle, H., Beygelzimer, A., dAlché-Buc, F., Fox, E., Garnett, R., Eds.; Vancouver, Canada, 8–14 December 2019; Vol. 32.
42. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016.
43. Bottou, L. Stochastic Gradient Descent Tricks. In *Neural Networks: Tricks of the Trade*, 2nd ed.; Montavon, G., Orr, G.B., Müller, K.R., Eds.; Springer: Berlin/Heidelberg, Germany, 2012; pp. 421–436. https://doi.org/10.1007/978-3-642-35289-8_25.
44. Robert, C.P.; Casella, G. Diagnosing Convergence. In *Monte Carlo Statistical Methods*; Robert, C.P., Casella, G., Eds.; Springer: Berlin/Heidelberg, Germany, 2004; pp. 459–509. https://doi.org/10.1007/978-1-4757-4145-2_12.
45. Law, H.; Deng, J. CornerNet: Detecting Objects as Paired Keypoints. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 734–750.
46. Yang, X.; Yan, J. On the Arbitrary-Oriented Object Detection: Classification Based Approaches Revisited. *Int. J. Comput. Vis.* **2022**, *130*, 1340–1365. https://doi.org/10.1007/s11263-022-01593-w.
47. Li, T.; Comer, M.; Zerubia, J. An Unsupervised Retinal Vessel Extraction and Segmentation Method Based On a Tube Marked Point Process Model. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Barcelona, Spain, 4–8 May 2020; pp. 1394–1398. https://doi.org/10.1109/ICASSP40776.2020.9054023.
48. Pham, T.T.; Hamid Rezatofighi, S.; Reid, I.; Chin, T.J. Efficient Point Process Inference for Large-Scale Object Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, USA, 27–30 June 2016; pp. 2837–2845. https://doi.org/10.1109/CVPR.2016.310.
49. Zhao, Y.; Wang, G.; Tang, C.; Luo, C.; Zeng, W.; Zha, Z.J. A Battle of Network Structures: An Empirical Study of CNN, Transformer, and MLP. *arxiv* **2021**, arxiv:2108.13002. https://doi.org/10.48550/arXiv.2108.13002.
50. Ruelle, D. Superstable Interactions in Classical Statistical Mechanics. *Commun. Math. Phys.* **1970**, *18*, 127–159. https://doi.org/10.1007/BF01646091.
51. Mabon, J. Learning Stochastic Geometry Models and Convolutional Neural Networks. Application to Multiple Object Detection in Aerospatial Data Sets. Ph.D. Thesis, Université Côte d'Azur, Nice, France, 2023. https://theses.hal.science/tel-04404849.
52. Long, J.; Shelhamer, E.; Darrell, T. Fully Convolutional Networks for Semantic Segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015; pp. 3431–3440. https://doi.org/10.1109/CVPR.2015.7298965.

53. Shelhamer, E.; Long, J.; Darrell, T. Fully Convolutional Networks for Semantic Segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 640–651. https://doi.org/10.1109/TPAMI.2016.2572683.
54. Neven, D.; Brabandere, B.D.; Proesmans, M.; Van Gool, L. Instance Segmentation by Jointly Optimizing Spatial Embeddings and Clustering Bandwidth. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019; pp. 8829–8837. https://doi.org/10.1109/CVPR.2019.00904.
55. Mabon, J.; Ortner, M.; Zerubia, J. CNN-Based Energy Learning for MPP Object Detection in Satellite Images. In Proceedings of the IEEE 32nd International Workshop on Machine Learning for Signal Processing (MLSP), Xi'an, China, 22–25 August 2022; pp. 1–6. https://doi.org/10.1109/MLSP55214.2022.9943312.