Point process and graph cut applied to 2D and 3D object extraction

Ahmed Gamal Eldin

UNIVERSITY OF NICE - SOPHIA ANTIPOLIS - FRANCE

GRADUATE SCHOOL STIC
INFORMATION AND COMMUNICATION TECHNOLOGIES AND SCIENCES

THESIS

to fulfill the requirements for the degree of

D P C S

from the University of Nice - Sophia Antipolis

Specialized in : C , S I P

presented by

Ahmed GAMAL ELDIN

P 2D 3D

Supervised by Xavier D , and Josiane Z and prepared at INRIA Sophia-Antipolis Méditerranée in the A research team

Defended the 24^{th} of October, 2011

Jury:

M. Marc B		Emeritus Professor, INRIA	President
M. Fionn M		Professor, University of London	Reviewer
Mrs. Florence T		Professor, Telecom ParisTech	Reviewer
M. Jean-Denis D		Associate Professor, University of Toulouse	Reviewer
Mrs. Josiane Z		Director of Research, INRIA	Director
M. Xavier D		Director of Research, INRIA	Supervisor
M. Michel G	-C	Director of Research, Tour du Vala	Examiner
M. Guillaume P		Research engineer, Thales Alenia Space	Examiner

Dedication

I dedicate this to my very exceptional parents,

to my mother who taught me every thing good in my life, I would not be what I am without her, to my father who created for me a beautiful environment where I can live and work, I owe them everything in my life and I will never be able to thank them for all what they did for me.

God bless both of you...

Acknowledgements

I owe a very special acknowledgement to my supervisors, Josiane ZERUBIA and Xavier DESCOMBES.

Josiane, thank you for having accepting me as part of your team, for your great support and encouragement. Thank you for always being there during and even after my thesis. Xavier, thank you believing in me even when I did not, thank you for being so understanding. I really learned a lot from you. Thank you for giving me all the freedom during my thesis, working with you was a real pleasure for me. After my two supervisors, I want to thank my friend Guillaume CHARPIAT. Thank you Guillaume for all your help, your generous discussions, availability, it was a lot of fun to work with you. This work could not have been accomplished without you all.

I would like to thank Professors MARC BERTHOD, Fionn MURTAGH, Florence TUPIN, Jean-Denis DUROU, Michel GAUTHIER-CLERC and Guillaume PERRIN. Thank you for accepting to be members of my thesis jury. Thank you for your effort, your interest and your kind attention during the defence.

I owe a special acknowledgement to our ecologist collaborators. I would like to thank Yvon le MAHO, Michel Gauthier-Clerc and Arnaud Béchet for this collaboration, for all the interesting discussions and the efforts they made. I would like to thank every researcher on site who participated in this work: Onésime PRUD'HOMME, Celine Le BOHEC, Benjamin FRIESS, Yan ROPERT-COUDERT, Maryline Le VAILLANT, Claire SARAUX, Marion RIPOCHE, Rémi GENER.

I would like to start listing by friends who I want to thank with Alexandre FOURNIER who gave me the most significant advices during the early days of my PhD. Thanks to all my friends with whom I shared these three years, work and fun. Thank you Maria, Aymen, Giovanni, Csaba, Praveen, Aurélie, Vladmir, Sayma, Ioan, Guillaume Perrin, Mikael and Alexis. A special thank you as well to our very nice project assistants Corinne, Laurie and Christine.

I would like to thank Giovanni GHERDOVICH with whom I worked a lot during my first PhD year. I would like to thank Professor Elena Zhizhina and the many other great visitors of our team with whom I had very enriching discussions.

I would like to thank every person who helped during these three years.

Summary

The topic of this thesis is to develop a novel approach for 3D object detection from a 2D image. This approach takes into consideration the occlusions and the perspective effects. This work has been embedded in a marked point process framework, proved to be efficient for solving many challenging problems dealing with high resolution images. The accomplished work during the thesis can be presented in two parts:

First part:

We propose a novel probabilistic approach to handle occlusions and perspective effects. The proposed method is based on 3D scene simulation on the GPU using OpenGL. It is an object based method embedded in a marked point process framework. We apply it for the size estimation of a penguin colony, where we model a penguin colony as an unknown number of 3D objects. The main idea of the proposed approach is to sample some candidate configurations consisting of 3D objects lying on the real plane. A Gibbs energy is define on the configuration space, which takes into account both prior and data information. The proposed configurations are projected onto the image plane, and the configurations are modified until convergence. To evaluate a proposed configuration, we measure the similarity between the projected image of the proposed configuration and the real image, by defining a data term and a prior term which penalize objects overlapping. We introduced modifications to the optimization algorithm to take into account new dependencies that exists in our 3D model.

Second part:

We propose a new optimization method which we call "Multiple Births and Cut" (MBC). It combines the recently developed optimization algorithm Multiple Births and Deaths (MBD) and the Graph-Cut. MBD and MBC optimization methods are applied for the optimization of a marked point process. We compared the MBC to the MBD algorithms showing that the main advantage of our newly proposed algorithm is the reduction of the number of parameters, the speed of convergence and the quality of the obtained results. We validated our algorithm on the counting problem of flamingos in a colony.

Résumé en Français

L'objectif de cette thèse est de développer une nouvelle approche de détection d'objets 3D partir d'une image 2D, prenant en compte les occultations et les phénomènes de perspective. Cette approche est fondée sur la théorie des processus ponctuels marqués, qui a fait ses preuves dans la solution de plusieurs problèmes en imagerie haute résolution. Le travail de la thèse est structuré en deux parties:

Première partie:

Nous proposons une nouvelle méthode probabiliste pour gérer les occultations et les effets de perspective. Le modèle proposé est fondé sur la simulation d'une scène 3D utilisant OpenGL sur une carte graphique (GPU). C'est une méthode orientée objet, intégrée dans le cadre d'un processus ponctuel marqué. Nous l'appliquons pour l'estimation de la taille d'une colonie de manchots, là où nous modélisons une colonie de manchots comme un nombre inconnu d'objets 3D. L'idée principale de l'approche proposée consiste échantillonner certaines configurations candidat composé d'objets 3D s'appuyant sur le plan réel. Une densité de Gibbs est définie sur l'espace des configurations, qui prend en compte des informations a priori et sur les données. Pour une configuration proposée, la scène est projetée sur le plan image, et les configurations sont modifiées jusqu'à convergence. Pour évaluer une configuration proposée, nous mesurons la similarité entre l'image projetée de la configuration proposée et l'image réelle, définissant ainsi le terme d'attache aux données et l'a priori pénalisant les recouvrements entre objets. Nous avons introduit des modifications dans l'algorithme d'optimisation pour prendre en compte les nouvelles dépendances qui existent dans notre modèle 3D.

Deuxième partie:

Nous proposons une nouvelle méthode d'optimisation appelée "Naissances et Coupe multiples" ("Multiple Births and Cut" (MBC) en Anglais). Cette méthode combine la fois la nouvelle méthode d'optimisation Naissance et Mort multiples (MBD) et les "Graph-Cut". Les méthodes MBC et MBD sont utilisées pour loptimisation d'un processus ponctuel marqué. Nous avons comparé les algorithmes MBC et MBD montrant que les principaux avantages de notre algorithme nouvellement proposé sont la réduction du nombre de paramètres, la vitesse de convergence et de la qualité des résultats obtenus. Nous avons validé notre algorithme sur le problème de dénombrement des flamants roses dans une colonie.

Contents

General introduction

Part I

Markov Marked Point Process

Chapter 1

Introduction

Recent technological advances in the development of airborne sensors have significantly increased the number of available satellites, and their spatial and spectral resolutions. These advances, combined with the need for an effective environmental control, have favoured the emergence of new remote sensing applications. These applications include:

- Territorial Surveillance: characterization of urban development, such as new roads and buildings.
- Management of natural resources: deforestation characterization, species preservation (animals,...).
- Mapping the damage in natural disasters: volcanic eruptions, earthquakes, ...

1.1 Remote sensing

A typical example in remote sensing is given an input image, being able to classify it into different zones. Figure 1.1 shows a SPOT 5 image (of 5 m resolution), containing two zones, urban and fields. These two zones can easily be characterized by their textural content, more precisely the conditional variance of the luminance inside each zone. Assuming a Gaussian Markov model, the variance of the pixel given the mean of its neighbors provides a textural feature characterization for the urban area [31] [66]. The availability of large amounts of rich data has induced many new applications and new demands. This large amount of data requires rigorous mathematical models to create automated or semi-automated algorithms.

1.1.1 Markov Random Field

Among the numerous developed techniques, statistical methods, which rely on probabilistic models, have been among the most successful. The most popular stochastic model is known as Markov random field (MRF) model. MRF modeling is based on an inverse problem approach, where each zone can be modeled by a hidden class with a set of parameters.

Figure 1.1: Urban zone taken by Spot 5 ©CNES

e.g. we assume that the urban and the field classes present in figure 1.1, each was generated by a Gaussian distribution of certain mean and variance. Actually, just modeling each class with a Gaussian, and classifying pixels assuming their independence is a simple Maximum Likelihood (ML) approach. The MRF in addition to this ML approach, adds the class dependence between neighbor pixels to solve complex situations, when the independent pixel based approach is not sufficient. MRF modeling has emerged in the image processing and computer vision field since the 80s [42]. Thanks to the equivalence established by the Hammersly-Clifford theorem [11], the full conditional probability of an image (all pixels) can be written in the Gibbs field form, which is summarized by the sum of potentials over cliques. The segmentation result using MRF modeling of the scene presented in figure 1.1 into two classes is given in figure 1.2.

This type of modeling is useful in many remote sensing applications, based on a *data term* which measures how likely this pixel comes from that class, and a *prior term* to insert prior information and neighboring dependence for regularization. During the last decade, sensors have largely improved, and obtained images have a much higher resolutions than before. These higher image resolutions, which offer richer information, opened the door (appetite) for new applications, which consequently requires new modeling paradigms.

Given an aerial image of some field area such as the one shown in figure 1.3.(a), or an urban area, as shown in figure 1.3.(b), the aim is not any more to segment to regions like urban or plantation. Now, we are interested in extracting the roads from figure 1.3.(a), as shown in figure 1.4.(a) and extract the buildings from figure 1.3.(b) as shown in figure 1.4.(b).

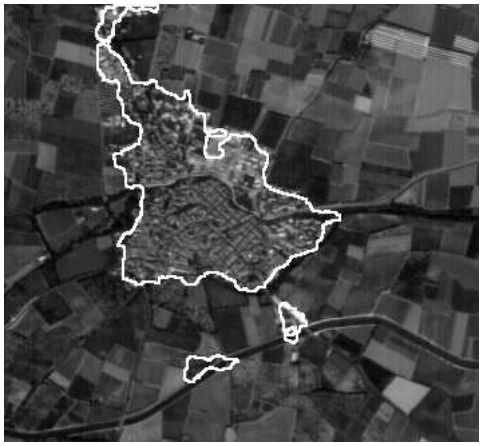


Figure 1.2: Segmentation of the urban zone using a Markov random field ©Ariana/INRIA.

1.1.2 From MRF to Point Process

While the idea of *data term* and *prior term* existing in the MRF modeling approach remains interesting, we need a more flexible model. MRF modeling is a pixel based approach, and extracted objects shown in figure 1.4 are better described by their *geometrical features*. The roads from figure 1.3.(a) can be represented by a set of segments and the buildings from figure 1.3.(b) can be represented by a set of rectangles. This type of geometrical modeling allows working on the *object level*, and not at the pixel level which is how it is usually done in MRF.

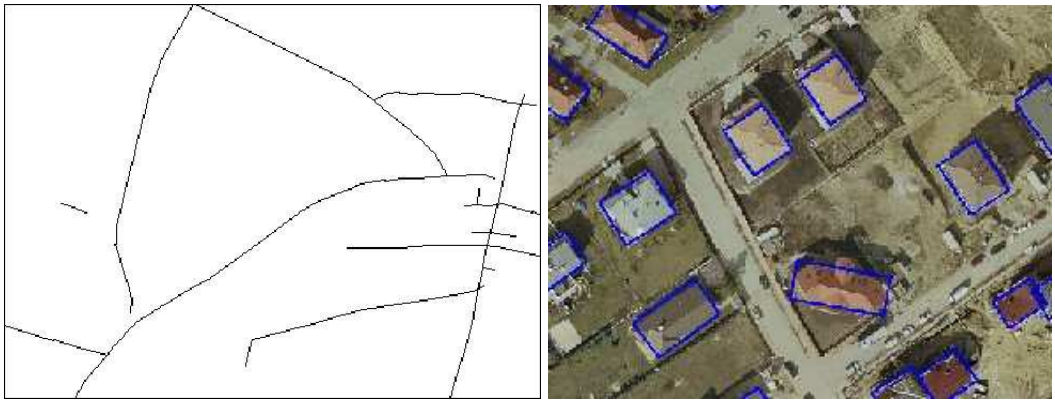
Insertion of geometrical information in MRF modeling is of very limited usage since it requires a lot of prior information such as the location and the number of objects which is not realistic in most applications since the aim is a detection algorithm.

In the literature, there exists many data terms, developed for roads and buildings detection and extraction. Only few of them use *prior information* as the *geometrical information* which is actually very important for a proper detection. Using geometrical forms regularize the detection, *e.g.* taking a road extraction problem, a tree shadow in the middle of the road is acting as noise, using the prior information stating that a road is a group of connected segments will bypass this noise easily.

We propose the usage of the Markov Point Process (MPP) framework. MPP framework is adapted to define some probabilistic models on configuration spaces consisting of an unknown number of parametric objects. An MPP model also contains two terms, a data term and a prior term. The data term is very flexible, we can easily insert any of the state-of-the-arts method for any application such as: buildings or trees detection. The prior term

(a) (b)

Figure 1.3: (a) Aerial image of a field with a resolution of 25 cm ©IGN. (b) An aerial image of an urban zone in Budapest ©András Goro.



(a) (b)

Figure 1.4: (a) Extraction result of roads using a segment process ©Ariana/INRIA. (b) Extraction result of building using a rectangle process ©Ariana/INRIA.

is also very flexible, we can integrate the geometrical features of the object of interest, relation between objects, and any other prior information about the problem that can help obtaining better results.

In the building detection problem, we can insert information stating that buildings have a tendency to be aligned. In the road detection problem, we can set a minimal angle between detected roads like prohibiting the detection of two road with less than 5^0 between them. So based on the application and the prior information we have about the problem, we can insert such information into the MPP model to regularize the result.

1.2 Ecological application

In the history of the Earth, mass extinctions are far from rare events. However, the current speed of planetary extinctions of plant and animal species could be hundred to a thousand times faster than the most brutal of previous mass extinctions. The unprecedented biotic crisis due to human activities follows from the destruction and fragmentation of natural habitats, overfishing, introduction of exotic species, pollution and climate change.

In this thesis we concentrate on two ecological applications which are based on the counting of birds in flamingo and penguin colonies.

1.2.1 Flamingo counting

Taking into account the population dynamics can be used to anticipate events and thus provides valuable recommendations for the species preservation. Modern statistical methods make the usage of demographic models possible to assess the factors likely to influence the population dynamics and then make recommendations for the management and preservation of endangered species. These demographic patterns are usually calibrated by expert counts, allowing to validate predictions. The accuracy of their counting is essential. During the breeding season or during migration, flamingos (Phoenicopterus roseus) are grouped. The observers take this opportunity to do the counting. These techniques are based on counting from aerial photographs. A statement completed by local interpolation is often imprecise because of the non-stationary density of populations [2]. A count of a complete colony is very long and tedious [8], and even an expert counting is not 100% correct. An automated tool for performing this count is therefore very interesting for ecologists.

Here, we propose a new method for estimating the size of the populations of a flamingo population, based on object based process from aerial images. In figure 1.5, we present an aerial image of a full flamingo colony with estimated size of 16,000 bird. In figure 1.6, we present two samples from two different colonies. From those images, we can start to imagine how the data term will be designed, and what type of prior information can be used to enhance the results [25].

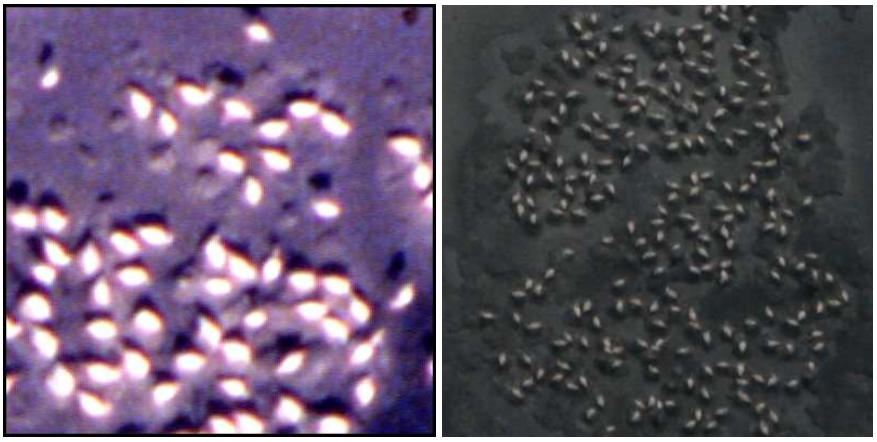
Figure 1.5: An aerial image presenting a flamingo colony in Camargue, in Fangassie island ©Tour du Valat.

1.2.2 Penguin counting

Predicting the impact of future climate changes on populations and biodiversity is a central issue in the context of global climate warming. Seabirds are sensitive indicators of changes in marine ecosystems and might integrate and/or amplify the effects of climate forcing on lower levels in food chains. It is crucial to understand how and to what extent organisms are able to cope with climatic variations, especially in polar environments where the effect of climate change is the strongest. Warm events negatively affect both breeding success and adult survival of King and Emperor penguins [39]. For King penguins, breeding reveals an immediate response to forcing during warm phases of El Nino Southern Oscillation affecting food availability close to the colony. Conversely, adult survival decreases with a remote sea-surface temperature forcing. All previous studies of the breeding cycle of king penguins depended on time methods that are time consuming for researcher and sometimes harmful for the birds [27].

Human monitoring and discerning for penguins is very hard. Previous, researchers used to resort on the usage of flipper band for identification of each bird. This method was often used until it was found that it harm the birds. In the late 1990s, on a research was accomplished by Yvon Le Maho [1] and his team to settle the debate over banding. The result of this research showed that banded birds have a lower survival rate over unbanded birds [40].

[1] A physiologist at the French National Center for Scientific Research's Centre d'Ecologie et Physiologie Energtiques in Strasbourg, France.



(a) (b)

Figure 1.6: (a) A sample of a flamingo colony in Tuz lac, in Turkey, 2004 ©Tour du Valat. (b) A sample from the same flamingo colony at 2006 ©Tour du Valat.

Based on this research, Le Maho and his team began using RFID tags, microchips that are injected under the skin of the penguin instead of tapes. These chips emit radio waves that researchers can use to track the movements of penguins. The only major advantage of bands over the microchips is there visibility, since microchips tags requires antennas everywhere the animal moves. We consequently need long term monitoring system allowing to estimate each year the number of breeding penguins in the colonies and their breeding success, that means the number of chicks.

The development of automatic counting system by photographic analysis of penguin colonies (number of adults and number of chicks) could be a powerful and cheap tool to monitor demography of penguins in remote areas and to study the impact of climate changes.

We consequently need a long term monitoring system allowing to estimate each year the number of breeding penguins in the colonies and their breeding success, that means the number of chicks. The development of an automatic counting system by photographic analysis of penguin colonies (number of adults and number of chicks) could be a powerful and cheap tool to monitor demography of penguins in remote areas and to study the impact of climate changes [82].

Our ecologist collaborators are interested in studying three different types of penguin colonies which are: Emperor, King and Adelie penguins. These penguin colonies live in the following three locations, also indicated in the map in figure 1.7.

1. Petrels island, Adelie land; it's a part of the Antarctica.

Coordonnates: −66.662778° 140.001389°

2. Possession island, in Crozet archipel, in the Indian Ocean.
   Coordonnates: −46.4° 51.766667°

3. Kerguelen island.

Figure 1.7: A map representing the three mentioned types of penguins live.

The penguin counting problem in a colony is very different from the flamingo counting problem (and others) since no top view imaging is possible. In figure 1.8, we present a sample of an emperor penguin colony. In the flamingo counting problem we had a top view of the scene and imaging was aerial. In the penguin counting case, this is not possible (feasible). Current civil satellite resolution is not enough (50 cm/pixel by US regulations), and it is forbidden to fly over these colonies.

1.3 Thesis organization

This thesis is organized as follows, it is composed of three parts: point process models, optimization methods, and applications.

Figure 1.8: Photo presenting a part of a King penguin, 2010 ©DEPE / CNRS.

- The first part holds two chapters, the general introduction covered in this chapter and an introduction to point process models. We start from simple models to the Markov model that we use in this thesis. In the second chapter, we only consider point process models of interest, or models that serve as a basis for the selected model in this thesis.

- The second part of this thesis is dedicated to the presentation of our contributions in two fields, optimization and 3D modeling. We start by covering the existing optimization algorithms dedicated to point process model. Then we introduce our proposed model. The proposed optimization algorithm is inspired from the Multiple birth-and-death (MBD) algorithm, where we take advantage of the successful graph-cut algorithm to overcome most of the limitations of the MBD algorithm. This algorithm is presented in three versions. The first version introduce the idea of the algorithm and present how is can replace existing algorithms for solving this type of problems, while have a speed drawback. In a second version we present a possible solution to the speed limitation where we take advantage of the efficient belief propagation algorithm to boost the performance of the proposed algorithm. Last, we present a third version which is superior to the previous versions and to the other existing algorithms.

  We next introduce a 3D point process model, explaining the basics behind the 3D model. We also covers how the optimization methods are adapted to handle this type

of models.

- The last part of the thesis is dedicated to the applications and conclusions. The proposed algorithms are first tested on synthetic data to study the minimal energy that can be reached on controlled situation and how the algorithm scales with the problem size. Next, we validate the proposed optimization algorithms on the flamingo counting problem where our results are validated by ecologists studying the evolution of the species.

  The second application is treated in chapter seven. In this problem, we employ the proposed 3D point process model introduced for addressing the counting of penguin colonies.

  Finally, we end this thesis by a general conclusion on the contributions. We conclude on the promising new optimization method methods, and mentioning potential future work on the other classes of point process models. We also comment on the proposed 3D model, and comment about potential future work for the penguin counting problem.

Chapter 2

Point Process

By looking to bacteria (or cells) under a microscope, such as those shown in figure 2.1, we may start asking: Are they randomly distributed? Do they hold a pattern? If there is a pattern, can we characterize it? Actually, we can describe this distribution: it forms some clusters (groups), groups are somehow distant, there is a sort of repulsion on the short range, and they do not overlap. Therefore, new questions arise: Is there a formal way to characterize this? Can we quantify it? Fortunately the answer is "yes", we can do this using *Point Process theory* [87].

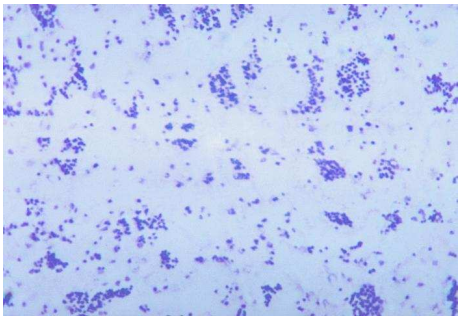


Figure 2.1: A microscopic photo of Francisella tularensis bacteria stained with methylene blue

2.1 Spatial Point Process

A spatial point process is a random pattern of points in $\mathbb{R}^d$ (d-dimensional Euclidean space). For most applications, $d \in \{1, 2, 3\}$. Spatial point processes are useful as statistical models in the analysis of observed patterns of points, where points represent the locations of some objects of study, like trees, cells, flamingos, and others.

2.1.1 Basics of 1D and 2D Point Process

The one dimensional model is useful to model sequence of events, *e.g.* time based events. For example, the call arrival time in a telephone switch can be modeled as a 1D point process, as shown in figure 2.2.

Figure 2.2: Time arrival of events, *e.g.* call to telephone switch.

One dimensional point process has an extra intrinsic property compared to higher dimensional version which is the natural ordering (causality). Usually, they are studied by considering the inter-arrival times, $S_i = T_{i+1} - T_i$, for $T_1 \leq T_2 \leq \ldots$, as presented in figure 2.3. An alternative way to handle this process mathematically is in term of cumulative counting process. Let $N_t = \sum_{i=1}^{\infty} \mathbf{1}\{T_i \leq t\}$, for defining the number of arriving points up to time t where $\mathbf{1}\{\ldots\}$ denotes the indicator function. We can also use the interval counting, $N(a, b] = N_b - N_a$, for $0 \leq a \leq b$ which counts the arriving number of points in this interval [4].

Figure 2.3: Inter-arrival time of events.

In higher dimensions, there is no natural ordering as in one dimension, so inter-arrival counting concept has to be generalized. The alternative for "interval counting" is "region counting". Lets define $n(B)$ as a counting function giving the number of points living in a region B (cardinality), or more precisely in a bounded closed set $B \subset \mathbb{R}^d$, as illustrated in figure 2.4.

Point processes can also be characterized by what is called the void or vacancy indicator $\mathcal{V}(.)$, where $\mathcal{V}(B) = \mathbf{1}\{n(B) = 0\}$, means that there are no points falling in the set B, as presented in figure 2.5.

Figure 2.4: Counting variable $n(B)$ for a spatial point process.

Figure 2.5: Void function $v(B)$ for a spatial point process.

The values of the counting variables $n(B)$ for all subsets B gives sufficient information to reconstruct completely the position of all the points in the process, the same concept applies to vacancy indicator.

Let X be a spatial point process on a space S, where $S \subseteq \mathbb{R}^d$. We only consider locally finite (lf) realisations, which means the number of points $n(x_B) < \infty$ where $x_B = x \cap B$ for all Borel sets B; and also simple point processes where each realization cannot include the same point twice. Thus X takes values in the space defined by:

$$N_{lf} = \{x \subseteq S : n(x_B) \leq \infty \text{ for all bounded } B \subseteq S\}.$$

2.1.2 Marked Point Processes

In many stochastic process models, as point process, point locations may not arise as the primary object of study, but as a component of a more complex model. Often, the point

process is the component that carries the information about the locations in time or space of objects that may themselves have a stochastic structure and stochastic dependency relations. Going back to the cells example of figure 2.1, we may want to encode some extra information in the point process, other than the cell location, such as cell size, or cell type if there are different types of cells and any other information.

Let Y be a point process on $T \subseteq \mathbb{R}^d$, given a mark space $\mathbb{M}$, if a random mark $m \in \mathbb{M}$ is attached to each point $x \in Y$, then

$$X = \{(x, m_x) : x \in Y\}$$

is a realization of a marked point process with points in T and mark space $\mathbb{M}$. In figure 2.6, we present a realization of a marked point process, objects are disks specified by a random radius as marks using Spatstat package [1]. If $\mathbb{M} = \{1, \ldots, k\}$, are the marks specifying different types of points, then the point process becomes a multi-type point process.

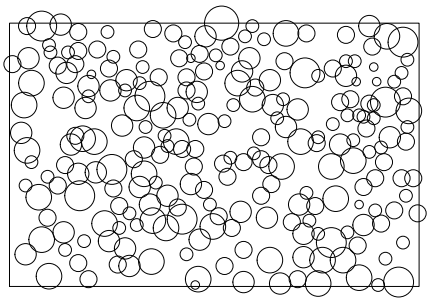


Figure 2.6: A realization of marked point process of disks of varying radius. Image simulated using SpatStat package.

2.1.3 Point Processes Models

In this section, we will introduce some of the existing point process models, starting by the simplest model being the Binomial process.

[1] Spatial Statistics is an open source software for Spatial Statistics (Spatstat) developed by Adrian Baddeley. Source: www.spatstat.org

Binomial Point Processes

Let W be a window (a compact set), within which n points are independently and uniformly distributed (i.i.d.).

Properties of Binomial Point Processes [84]:

- The n points $x_1, \ldots, x_n$ are stochastically independent, *i.e.* the probability that x_1 lies in the Borel set $B_1 \subset W$, ..., that x_n lies in the Borel set $B_n \subset W$ satisfy the formula

$$P(x_1 \in B_1, ..., x_n \in B_n) = P(x_1 \in B_1) \times P(x_2 \in B_2) \cdots P(x_n \in B_n),$$

  which means they are disjoint.

- Each of the $x_1, \ldots, x_n$ points is uniformly distributed in W, *i.e.* for $i = 1, \ldots, n$ and any Borel set $B \subset W$

$$P(x_i \in B_1) = \frac{A(B)}{A(W)},$$

  where $A(.)$ denotes the area.

The mean number of points per unit area is

$$\lambda = \frac{n}{A(W)},$$

where n is the total number of objects. The mean number of point in the Borel set B is

$$\mathbf{E}[N(B)] = \lambda A(B).$$

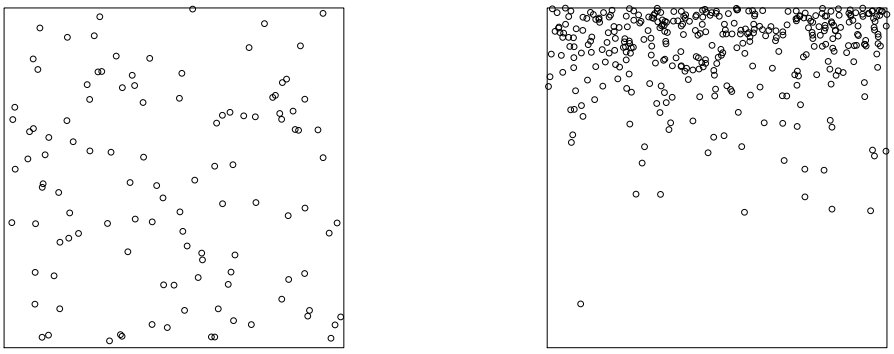
The one dimensional number distributions are:

$$P(N(B) = k) = \binom{n}{k} p_B^k (1 - p_B)^{n-k} \quad \text{,where } (k = 0, \ldots, n),$$

where $p_B = \frac{A(B)}{A(W)}$. While the m-dimensional number distributions are:

$$P(N(B_1) = k_1, \ldots, N(B_m) = k_m) = \frac{n!}{k_1! \ldots k_m!} \frac{A(B_1)^{k1} \ldots A(B_m)^{k_m}}{A(W)^n},$$

where $k_1 + \cdots + k_m = n$.



(a) (b)

Figure 2.7: (a) Uniform Poisson intensity 100, n= 114, (b) Non uniform $\rho(x, y) \propto \exp(8 * y)$, $(x, y) \in S$ n=356, where $S = [0, 1] \times [0, 1]$ for both.

Poisson Point Processes

Poisson point process is the simplest process, from which other models can be extended. The points distribution satisfy very strong independence conditions, in particular, the number of points in disjoint sets are stochastically independent. They often serve as a reference model for complete spatial randomness. The difference between a Binomial and a Poisson point process is that the number of point in a Poisson process is not fixed contrary to the Binomial. This number follows a Poisson distribution based on the intensity of the process. Different realizations of Binomial point process gives the same number of points, while a Poisson point process gives different number of points.

Although the two simulations presented in figure 2.7 are obtained with a Poisson point process, we can obviously notice that there is a main difference between them. To characterize this difference, let us consider a Poisson point process defined on S and specified by the intensity function $\rho : S \rightarrow [0, \infty)$. The intensity measure μ of the Poisson process is given by:

$$\mu(B) = \int_B \rho(\xi) d(x), \ B \subseteq S.$$

The intensity measure μ determines the expected number of points in B and we can say that $\rho(x)d(x)$ is the probability for the occurrence of a point in an infinitesimally small ball with center x and volume dx.

Obviously, based on the intensity ρ, we have two types of point process (Poisson or others), if ρ is constant, the process is *homogeneous*, otherwise it is inhomogeneous on S. The simulation in figure 2.7.(a) is homogeneous, with intensity $\rho = 100$, while the one in figure 2.7.(b) is inhomogeneous where the intensity $\rho(x, y) \propto \exp(8 * y)$, $(x, y) \in S$.

A point process X on S is Poisson with intensity function ρ, if the following properties are satisfied [4]:

- for every compact set $B \subseteq S$, the count $N(B)$ has a Poisson distribution with mean $\mu(B)$.
- if $B_1, \ldots, B_m$ are disjoint compact sets, then $N(B_1), \ldots, N(B_m)$ are independent.

Poisson point process is usually used to construct other models. Starting with a Poisson point process, while there exist simple ways for extensions such as mapping, thinning, superimposing, we can not construct all types of model with those basic techniques. Some models have to be defined by their probability density function.
We shall expand the Poisson process with density f, with respect to a $Poisson(S, \rho)$, where $S \subseteq \mathbb{R}^d$, $F \subseteq S$ and $\mu(S) < \infty$ by [71]:

$$P(X \in F) = \sum_{n=0}^{\infty} \frac{\exp(-\mu(B))}{n!} \int_B \ldots \int_B \mathbf{1}[\{x_1, \ldots, x_n]\} \in F] f(\{x_1, \ldots, x_n]\}) \prod_{i=1}^{n} \rho(x_i) dx_1 \ldots dx_n \tag{2.1}$$

where the integral for $n = 0$ is read as $\mathbf{1}[\emptyset \in F]$, f is said to be the probability density of the point process with distribution P.

2.2 Markov Point Processes

Markov point processes are point processes with interacting points. They are constructed by considering a density with respect to a Poisson process and imposing certain conditions ensuring the Markovian properties.

2.2.1 Conditional Intensity

By reformulating the density function f from equation 2.1, we get [4]:

$$\begin{aligned} f(\{x_1, \ldots, x_n\}) &= f(\emptyset) \frac{f(\{x_1\})}{f(\emptyset)} \frac{f(\{x_1, x_2\})}{f(\{x_1\})} \ldots \frac{f(\{x_1, \ldots, x_n\})}{f(\{x_1, \ldots, x_{n-1}\})} \\ &= f(\emptyset) \beta^*(x_1; \emptyset) \beta^*(x_2; x_1) \ldots \beta^*(x_n; \{x_1, \ldots, x_{n-1}\}). \end{aligned}$$

In this equation, we have introduced a new characterization of the process called *exterior conditional distribution*, also known as *Papangelou conditional intensity*, it is defined by [6]:

$$\beta^*(u; \mathbf{x}) = \frac{f(\mathbf{x} \cup u)}{f(\mathbf{x})}\ , \tag{2.2}$$

where $\mathbf{x}$ is a configuration $\mathbf{x} = \{x_1, \ldots, x_n\}$. It can be more convenient to formulate a point process model in term of its conditional intensity $\beta^*(u; \mathbf{x})$ rather than its probability density $f(\mathbf{x})$ since it has a natural interpretation which is easier to understand. In terms of statistical physics, $\log\ f(\mathbf{x} \cup u) - \log\ f(\mathbf{x})$ is the energy required to add a new point u to an existing configuration $\mathbf{x}$. In terms of probabilities, $\beta^*(u; \mathbf{x})$ is the (conditional) probability of having a point in an infinitesimal region around u given that the rest of X is $\mathbf{x}$.

Using Papangelou conditional intensity we can characterize X (or f):

It is attractive if:

$$\beta^*(u; \mathbf{x}) \leq \beta^*(u; \mathbf{y}) \text{ whenever } \mathbf{x} \subset \mathbf{y}$$

and repulsive if:

$$\beta^*(u; \mathbf{x}) \geq \beta^*(u; \mathbf{y}) \text{ whenever } \mathbf{x} \subset \mathbf{y}.$$

As a simple example, for an inhomogeneous Poisson process with density $\rho(.)$, its Papangelou conditional intensity will be $\rho(u)$ since the points are independent.

Markov Properties

The practical appeal of Markov models lies in the form of the joint probability distribution for many variables: it is expressible as the product of many conditional probabilities, each depending on a small number of variables, defined only on *adjacent* points. This then raises the possibility of specifying the model purely in terms of local conditional probabilities. The Papangelou conditional intensity plays this role in point process modeling.

Let $\sim$ be a reflexive and symmetric relation on S, for all $u, v \in S$, $u \sim u$ and $u \sim v$ implies $u \sim v$. We say that u and v are neighbors if $u \sim v$, and define the neighborhood of u by $N_u = \{v \in S : v \sim u\}$.

Let us introduce another important property called the *hereditary*, which states that:

$$f(\mathbf{x}) > 0 \Rightarrow f(\mathbf{y}) > 0 \text{ for } \mathbf{y} \subset \mathbf{x},$$

which means that if the probability of the set $\mathbf{x}$ is greater than zero, this implies that any subset of $\mathbf{x}$ also has a probability greater than zero.

Definition [24]: A simple finite point process with density function f is a Markov point process if for every $\mathbf{x}$ with $f(\mathbf{x}) > 0$ (hereditary) its Papangelou conditional intensity $\beta^*(u;\mathbf{x}) = \frac{f(\mathbf{x}\cup u)}{f(\mathbf{x})}$ satisfies

$$\beta^*(u;\mathbf{x}) = g(u, \mathbf{x} \cap N_u). \tag{2.3}$$

In other words, for f to be the density function of a Markov point process, we require that, for all $\mathbf{x} \in S$, $y \in S \setminus \mathbf{x}$, the *(n+1) dimensional* joint density function $f_{n+1}(\mathbf{x} \cup y)$ must be expressible as a product of the *n dimensional* joint density function $f_n(\mathbf{x})$ and some function $g(.,.)$ that depends only on y and those elements of $\mathbf{x}$ that lie in the neighborhood of the extra point y.

For readers who are familiar with Markov random fields, the following properties in Markov point process have equivalents in MRF theory: the *Papangelou conditional intensity* is similar to that of the *local characteristic* of a MRF, the concept of *adjacency* is analogous to the *neighborhood* in Markov random fields (MRF), while the *hereditary condition* corresponds to the *positivity condition* in the Hammersley-Clifford theorem for MRF.

Given an interaction function $\phi : N_f \to [0, \infty)$, and $\phi(x_i) = 1$ whenever there exist $x_i, x_j \in \mathbf{x}$ with $x_i \nsim x_j$. A Markov point process x on S, with density f with respect to a homogeneous Poisson point process with intensity 1, is a Markov point process if and only if

$$f(\mathbf{x}) = \prod_{\mathbf{y}\subseteq\mathbf{x}} \phi(\mathbf{y})$$

Knowing that the Papangelou conditional intensity $\beta^*(u;\mathbf{x}) = \frac{f(\mathbf{x}\cup u)}{f(\mathbf{x})}$, $u \in S \setminus u$ depends only points of $\mathbf{x}$ which are neighbors of u.

Gibbs Models

Gibbs and Markov represent the same models. Gibbs models arise in statistical physics for the description of large interacting particle systems.

In term of statistical physics, X is a finite Gibbs process with energy:

$$U(\mathbf{x}) = -\sum_{\mathbf{y}\subseteq\mathbf{x}:\mathbf{y}\neq\emptyset} \log \phi(\mathbf{y}) - \log Z \tag{2.4}$$

where Z is the partition function. For a finite point process X with probability density f, the

Gibbs density is defined by:

$$f(\mathbf{x}) = \frac{1}{Z} \exp\Big(V_0 + \sum_{x_i \in \mathbf{x}} V_1(x) + \sum_{\{x_i, x_j\} \subset \mathbf{x}} V_2(x_i, x_j) + \dots\Big), \tag{2.5}$$

where $V_k(.)$ is the potential of order k. It is often assumed that the effective interaction is dominated only by interactions between pairs of points and the higher order interactions are neglected.

2.2.2 Cliques and Interactions Order

A finite subset $\mathbf{x}$ on S is called a clique if all points of $\mathbf{x}$ are neighbors. By convention, the empty set and singletons are cliques. The set of cliques is denoted $\mathcal{C}$.

Pairwise Interaction. The density of a pairwise interaction point process is given by:

$$f(\mathbf{x}) \propto \prod_{x_i \in \mathbf{x}} \phi(\{x_i\}) \prod_{\{x_i, x_j\} \subset \mathbf{x}} \phi(\{x_i, x_j\}), \tag{2.6}$$

where $\phi(.)$ is an interaction function. In a Poisson process, $\phi(x_i)$ is the intensity and $\phi(\{x_i, x_j\}) = 1$. The range of interaction is defined by:

$$R = \inf\{r > 0 : \text{for all } \{x_i, x_j\} \subset S,\ \phi(x_i, x_j) = 1 \text{ if } \|x_i - x_j\| > r\}$$

The notation $\|.\|$ denotes the usual Euclidean distance.

Examples of pairwise interaction point processes

We consider the homogeneous case. Let $\phi_k()$ be the restriction of $\phi()$ to subsets consisting of k points. *e.g.* for a pairwise interaction process, the interaction function $\phi_k() \equiv 1$ for $k > 2$.

Example: Strauss process
One of the most well-known homogeneous Markov point processes is the Strauss process. The pairwise interaction of the process is given by [53]:

$$\phi_k(\mathbf{x}) = \begin{cases} \alpha & \text{if } k = 0 \\ \beta & \text{if } k = 1 \\ \gamma & \text{if } k = 2,\ x \in \mathcal{C} \\ 1 & \text{otherwise} \end{cases}$$

where $\beta \in]0, 1[$ is the interaction parameter, α and γ are parameters. Let $n(\mathbf{x})$ be the number of points in $\mathbf{x}$, the density is usually written as:

$$f(\mathbf{x}) = \alpha \beta^{n(\mathbf{x})} \gamma^{s(\mathbf{x})} \tag{2.7}$$

and $s(x)$ is the number of neighbor pairs in x:

$$s(x) = \sum_{\{x_i, x_j\} \subseteq \mathbf{x}} \mathbf{1}[\|x_i - x_j\| \leq R] \tag{2.8}$$

$R > 0$ is the interaction range. The neighborhood relation is given by:

$$x_i \sim x_j \iff \|x_i - x_j\| < R,$$

then the process is called a Strauss process with interaction of radius R. This process is repulsive. A special case when $\gamma = \infty$ is called the hard core process with hard core R, where points are prohibited from being closer than distance R apart. The interaction parameter can be written differently, $\phi_2(r) = \gamma^{\mathbf{1}[r \leq R]}$, where $\gamma \in [0, 1]$, $R > 0$ and when $\gamma = 0$ it becomes a hard core process.

Ex: Another example with finite range of interaction
By replacing the step interaction function of Strauss by a linear decreasing one, we can get:

$$\phi_2(r) = \mathbf{1}_{[r \leq R]} \frac{r}{R}.$$

This interaction function is not differentiable at $r = R$, but it gives an idea of linear interaction functions.

Ex: An example with infinite range of interaction
A very soft core process can be obtained by:

$$\phi_2(r) = 1 - \exp(-(r/\theta)^2),$$

where $\theta > 0$.

n-order interaction point processes

Pairwise interaction processes provides the simplest examples of Markov point processes. An example of a process with higher order of interactions is called *Geyer's triplet process*, it is a modification of the Strauss process with density [71]:

$$f(x) \propto \beta^{n(x)} \gamma^{s(x)} \delta^{t(x)}$$

where $s(x)$ is the same as in Strauss process (equation 2.8), and:

$$t(x) = \sum_{\{x_i, x_j, x_k\} \subseteq x} \mathbf{1}[\|x_i - x_j\| \leq R, \|x_i - x_k\| \leq R, \|x_j - x_k\| \leq R]$$

$\beta > 0$, and either:

1. $0 \leq \gamma \leq$ and $0 \leq \delta \leq$, or
2. $\gamma > 1$ and $0 < \delta < 1$

This process is clearly Markov with respect to R-close relation, with interaction function:

$$\phi(x_i) = \beta,\ \phi(\{x_i, x_j\}) = \gamma^{\mathbf{1}[\|x_i - x_j\| \leq R]}$$

$$\phi(\{x_i, x_j, x_k\}) = \delta^{\mathbf{1}[\|x_i - x_j\| \leq R, \|x_i - x_k\| \leq R, \|x_j - x_k\| \leq R]}$$

and $\phi(y) = 1$ for $n(y) \geq 4$. The process is repulsive in case (1) and neither repulsive nor attractive in case (2).

Inhomogeneous Markov point processes

Inhomogeneous Markov point processes, one way to be is, when the intensity function is not constant and the interaction between neighboring points is translation invariant. With a density given by equation 2.6, the first order interaction $\phi(x_i)$ will become non-constant.

2.2.3 Markov Marked Point Processes

In this section [71], we develop the concept of adding marks which was introduced in section 2.1.2 to Markov point processes. Let $Poisson(T \times \mathbb{M}, \rho)$ be a marked Poisson process defined on $S = T \times \mathbb{M}$, where $T \subset \mathbb{R}^d$, $|T| < \infty$, $\mathbb{M} \subseteq \mathbb{R}^p$, and $\rho(\xi, m) = p(m)$, where p is a discrete or continuous density on $\mathbb{M}$. In the $Poisson(T \times \mathbb{M}, \rho)$ point process, the point and the marks are independent, the points follow a Poisson process on T, and the marks are i.i.d. with mark density p. Let $X = \{(\xi, m_\xi) : \xi \in Y\}$ be a marked point process with respect to $Poisson(T \times \mathbb{M}, \rho)$. The density f of X is defined on the N_f set of finite marked point configurations in $S = T \times \mathbb{M}$

$$N_f = \{\{(\xi_1, m_1), \ldots, (\xi_n, m_n)\} \subset T \times \mathbb{M} : n < \infty\}$$

For $F \subseteq N_f$, for the continuous case of p,

$$\begin{aligned} P(X \in F) \quad = \quad & \sum_{n=0}^{\infty} \frac{\exp(-\mu(B))}{n!} \int_B \int_M \ldots \int_B \int_M \mathbf{1}[\{(\xi_1, m_1) \ldots, (\xi_n, m_n)]\} \in F] \\ & f(\{(\xi_1, m_1), \ldots, (\xi_n, m_n)]\}) p(m_1) \ldots p(m_n) d\xi_1 dm_1 \ldots d\xi_n dm_n \end{aligned}$$

where the integral for $n = 0$ is read as $\exp(-\mu(B))\mathbf{1}[\emptyset \in F] f(\emptyset)$. This is a special case of (2.1), where X can be viewed as a finite point process with a density proportional to

$$f(\{(\xi_1, m_1) \ldots, (\xi_n, m_n)]\}) p(m_1) \ldots p(m_n)$$

The Papangelou conditional intensity is defined by

$$\beta^*(u; (\xi, m)) = \frac{f((\xi, m) \cup u)}{f((\xi, m))},\ u \in N_f,\ (\xi, m) \in (T \times \mathbb{M}) \setminus u$$

The neighborhood relation $\sim$ is defined on $T \times \mathbb{M}$. The density of the pairwise interaction becomes

$$f(\mathbf{x}) \propto \prod_{\xi \in y} \phi((\xi, m_\xi)) \prod_{\{\xi,\eta\} \subset y} \phi(\{(\xi, m_\xi), (\eta, m_\eta)\}), \tag{2.9}$$

for $x = \{(\xi, m_\xi) : \xi \in y\} \in N_f$. *e.g.*, a Strauss disc process with $\mathbb{M} \subseteq (0, \infty)$ is given by

$$\phi((\xi, m_\xi)) = \beta,\ \phi(\{(\xi, m_\xi), (\eta, m_\eta)\}) = \gamma^{\mathbf{1}[\|\xi-\eta\| \leq m_\xi + m_\eta]}$$

where $\beta > 0$ and $0 \leq \gamma \leq 1$. This is Markov with respect to the overlapping disc relation defined by $(\xi, m_\xi) \sim (\eta, m_\eta)$ if and only if $\|\xi - \eta\| \leq m_\xi + m_\eta$.

2.2.4 Markov Marked Point Processes

In this section, we develop the concept of adding marks which was introduced in section 2.1.2 to Markov point processes.

Let $Poisson(T \times \mathbb{M}, \rho)$ be a marked Poisson process defined on $S = T \times \mathbb{M}$, where $T \subset \mathbb{R}^d$, $|T| < \infty$, $\mathbb{M} \subseteq \mathbb{R}^p$, and $\rho(x, m_x) = p(m)$, where p is a discrete or continuous density on $\mathbb{M}$. In the $Poisson(T \times \mathbb{M}, \rho)$ point process, the point and the marks are independent, the points follow a Poisson process on T, and the marks are i.i.d. with mark density p. Let $X = \{(x, m_x) : x \in Y\}$ be a marked point process with respect to $Poisson(T \times \mathbb{M}, \rho)$. The density f of X is defined on the N_f set of finite marked point configurations in $S = T \times \mathbb{M}$ as follows:

$$N_f = \{\{(x_1, m_{x_1}), \ldots, (x_n, m_{x_n})\} \subset T \times \mathbb{M} : n < \infty\}$$

For $F \subseteq N_f$, for the continuous case of p,

$$\begin{aligned} P(X \in F) \quad = \quad & \sum_{n=0}^{\infty} \frac{\exp(-\mu(B))}{n!} \int_B \int_M \cdots \int_B \int_M \mathbf{1}[\{(x_1, m_{x_1}) \ldots, (x_n, m_{x_n})]\} \in F] \\ & f(\{(x_1, m_{x_1}), \ldots, (x_n, m_{x_n})]\}) p(m_1) \ldots p(m_n) dx_1 dm_1 \ldots dx_n dm_{x_n} \end{aligned}$$

where the integral for $n = 0$ is read as $\exp(-\mu(B))\mathbf{1}[\emptyset \in F] f(\emptyset)$. This is a special case of (2.1), where X can be viewed as a finite point process with a density proportional to

$$f(\{(x_1, m_{x_1}) \ldots, (x_n, m_{x_n})]\}) p(m_1) \ldots p(m_n)$$

The Papangelou conditional intensity is defined by

$$\beta^*(u; (x, m_x)) = \frac{f((x, m_x) \cup u)}{f((x, m_x))},\ u \in N_f,\ (x, m_x) \in (T \times \mathbb{M}) \setminus u$$

The neighborhood relation $\sim$ is defined on $T \times \mathbb{M}$. The density of the pairwise interaction becomes

$$f(\mathbf{x}) \propto \prod_{x \in y} \phi((x, m_x)) \prod_{\{x,\eta\} \subset y} \phi(\{(x, m_x), (\eta, m_\eta)\}) \tag{2.10}$$

A Strauss disc process with $\mathbb{M} \subseteq (0, \infty)$ is given by

$$\phi((x, m_x)) = \beta, \ \phi(\{(x, m_x), (\eta, m_\eta)\}) = \gamma^{\mathbf{1}[\|x-\eta\| \leq m_x + m_\eta]}$$

where $\beta > 0$ and $0 \leq \gamma \leq 1$. This is Markov with respect to the overlapping disc relation defined by $(x, m_x) \sim (\eta, m_\eta)$ if and only if $\|x - \eta\| \leq m_x + m_\eta$.

2.3 Conclusion

In this chapter we briefly presented an interesting mathematical framework for studying objects with particular spatial structures.

We started by the very basic 1D point process model to link ideas that most people are familiar with and developed them one step further to treat 2D problems. For the selected applications, modeling only objects positions will not be enough. Therefore we introduced the idea of marks, and showed how on top of any point process model a mark can be associated to each point to hold properties of objects of interest.

We presented some of the basic types of point process models, such as the Binomial process and the Poisson process. Poisson process is essential for the construction of more sophisticated models such as the Markov process which we introduced later. We discussed important topics such as the conditional intensity, adjacency and the hereditary condition. We presented different models with different interaction orders and ranges.

In order to use point process models for objects detection from images, a sampling mechanism will be required. By sampling from a selected density, we will obtain candidate configurations. These configurations are modified inside an optimization algorithm until finding the optimal configuration that matches the selected model to the input image. Optimization methods for point process models will be introduced in the next chapter.

Part II

Optimization Methods

Chapter 3

Optimization

The optimization process can be a bottleneck for many sophisticated models. The richer the model is, the more complex its optimization becomes. This phenomenon makes most of the research focus on very simple models that can easily be optimized, even if these models can be over-simplistic for some applications.

In the previous chapter, we introduced Point process models ranging from complete randomness as Poisson model to Gibbs models where interaction exists on cliques. Marks defining the parameters of a geometric object, coupled with a Gibbs density provide a powerful tool for modeling a scene of objects and interaction between them.

The density of a Gibbs process is defined by an energy written as the sum of potential over interacting objects (cliques):

$$f(\omega) = \frac{1}{Z} \exp[-U(\omega)] \tag{3.1}$$

where

$$U(\omega) = \left(V_0 + \sum_{\omega_i \in \omega} V_1(\omega_i) + \sum_{\{\omega_i, \omega_j\} \in \omega} V_2(\omega_i, \omega_j) + \dots \right) \tag{3.2}$$

Z is the partition function (normalizing constant), and V_k are the potential on a clique of order k. Minimizing the energy $U(\omega)$ corresponds to the detection of the target configuration. This energy takes into account the interactions between geometric objects U_p (prior energy) and a data energy U_d to fit the configuration to the image:

$$U(\omega) = U_d(\omega) + \gamma_p U_p(\omega)$$

where γ_p is the weight assigned to the prior term.

3.1 Optimization

Now that we defined the density, prior and data term (that matches our application) by an energy function $U(.)$, we want to compute the Maximum Likelihood (ML) estimate with respect to the density f. The ML estimate corresponds to the configuration that matches best our model and the input image. The estimate of the global mode of a density is given by:

$$\hat{\omega} = argmax\ f(\omega) = argmin\ U(\omega)$$

Finding the minimum (or maximum), there exist a lot of optimization methods based on $f(\omega)$ properties. For a function such as $f(\omega)$ defined in 3.2, unfortunately this is a highly non-convex function with many local minima. So this requires the usage of a global optimization method.

3.1.1 Simulated annealing

One of the very popular global minimization algorithms is the Simulated Annealing algorithm. When $f(\omega)$ is highly non-convex (especially if it is very peaky), simulated annealing proposed to deal with a slightly different function $f^{1/T}(\omega)$. T is a free parameter, usually referred to as the temperature, it originates from statistical physics. The idea of the simulated annealing is, starting with a large value for T will make $f^{1/T}(\omega)$ a smoother version of $f(\omega)$, which will make it easier to find the global minimum of this smoother version. The global minimum of the smoother $f(\omega)$ and the smoothed version $f^{1/T}(\omega)$ should be close. The simulated annealing algorithm keeps decreasing the temperature T parameter until zero, where $f^{\infty}(\omega)$ will be concentrated on the set of the global maximum of $f(\omega)$. Stopping the temperature at $T = 1$ gives the same function, and this is used for sampling, not optimization.

Simulated annealing may be the most popular algorithm for global minimization for a large class of problems. To obtain an efficient annealing algorithm, the temperature value during the algorithm is of crucial importance. The initial temperature is a function of the problem. The temperature has to be decreased to zero respecting a specific scheduling to converge to the global minimum. Logarithm scheduling guarantees this global minimum convergence, but is very slow in practice. It is usually replaced by a geometric one. The rule of thumb will be, if there is no other existing technique that can solve your problem, and you seek the global minimum, then you should use simulated annealing.

To find the global minimum of the density $f(\omega)$ given by equation 3.2, we need to draw samples from this density. Due to the complexity of this density, and due to the unknown dimensionality in advance (we do not know the number of objects we want to detect), we have to use special sampling algorithms.

3.2 Sampler

The basic idea of sampling methods is to obtain a set of samples drawn independently from the target distribution. Samplers can be divided into two categories, perfect samplers and non-perfect samplers. For several reasons, some samplers can not be exact in the precise sense, reasons are mainly due to the fact that random number generator are not perfect or the algorithm converges to the desired distribution at infinity (limit theorems). Perfect samplers also do not suffer from the burn-in required by non-perfect samplers [71] [77].

3.2.1 Birth and Death

Samplers such as those based on Markov chain often require a burn-in time, one often discards an initial set of samples. On the contrary, a perfect sampler does not require this burn-in time. Birth and death is such a sampler for point process models [71].

Perfect samplers such as birth and death are considered as advances in sampling theory, and they can guarantee independent samples under certain conditions. Unfortunately, this algorithm is very slow for image processing applications. The main reasons for birth and death to be slow are:

- The algorithm is based on very basic kernels, only birth and death, which will hardly (in very long time) explore the configuration space. A well designed sampler should incorporate *global proposal* to explore vast regions of the space and *local proposals* to discover finer details of the target distribution [2].

- The algorithm is based on *simple perturbations* to the current configuration, only one object is added or removed.

- These simple perturbations, the birth and death kernels, are computationally expensive.

Let u be a newly added object (by birth) on K with respect to $\frac{\nu(.)}{\nu(K)}$, where ν is a measure on K. Let B be a bounded closed set $B \subset \mathbb{R}^d$, and $A \in B$. Consider that the death choses uniformly an object from the current configuration to remove. The birth and death kernel can be written as:

$$Q_{BD}(\omega, .) = p_b(\omega)Q_b(\omega, .) + p_d(\omega)Q_d(\omega, .), \tag{3.3}$$

where the two kernels are given by:

$$Q_b(\omega, A) = \int_{u \in K} 1_A(\omega \cup u)\frac{\nu(du)}{\nu(K)} \tag{3.4}$$

and

$$Q_d(\omega, A) = \sum_{u \in \omega} 1_A(\omega \setminus u) \frac{1}{n(\omega)}, \tag{3.5}$$

where $n(\omega)$ is the number of objects in configuration ω.

Let us define a point process by its density $h()$ with respect to the Poisson measure on K. Let $\mathbf{x}$ be a point configuration on K. We define a birth measure on K by $b(\mathbf{x}, u), \forall u \in K$, and a death measure $d(\omega, \omega_i), \forall \omega_i \in \omega$. Let $B(\omega) = \int_K b(\omega, u)du$ and $D(\omega) = \sum_{\omega_i \in \omega} d(\omega, \omega_i)$. The Birth-and-Death algorithm can be summarized by algorithm 3.1.

Algorithm 3.1 Birth-and-Death algorithm

while Not converged **do**
  Calculate $B(\omega)$ and $D(\omega)$
  $p \sim U_{(0,1)}$
  if $p < \frac{B(\omega)}{B(\omega)+D(\omega)}$ **then**
    get a random point $u \in K$ by sampling $\frac{b(\omega,.)}{B(\omega)}$
    $\omega^{[n+1]} \leftarrow \omega^{[n]} \cup u$
  else
    get a random point $\omega_i \in \omega$ by sampling $\frac{d(\omega,.)}{D(\omega)}$
    $\omega^{[n+1]} \leftarrow \omega^{[n]} \setminus \omega_i$
  end if
end while

To guarantee convergence to the target density, the following condition known as *detailed balance condition* should be met:

$$b(\omega, u)h(\omega) = d(\omega \cup u)h(\omega, u), \forall \omega \in \Omega, \forall u \in K.$$

3.2.2 Metropolis Based Samplers

The Monte Carlo Principle

Monte Carlo simulation method is a computational method that makes use of random numbers simulation [77] [45]. Monte Carlo method is generally defined to solve the following problems:

- generate a set of i.i.d set of samples from a target distribution $p(\mathbf{x})$ defined on a high dimensional space.
- estimate the expectation of a function under this distribution.

We are interested here only in the first usage. The set of samples can be used to approximate a target distribution with the following empirical point-mass function:

$$p_N(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N}\delta_{x_i}(\mathbf{x}),$$

where $\delta_{x_i}(\mathbf{x})$ is the delta Dirac mass located at x_i.

Rejection Sampling

The development of this idea coupled with early days of computer opened the door for a new sampling category. It was rapidly followed by the development of the Rejection sampling and Importance sampling. Rejection sampling can be summarized as follows. Sampling the target distribution $p(\omega)$, is accomplished by sampling of another easy to sample distribution $q(\omega)$ with an *accept/reject* procedure. A condition on $q(\omega)$ to be satisfied is, $p(\omega) \leq Mq(\omega),\ M < \infty$. This condition means that the support of the density $q(\omega)$ should contains the support of $p(\omega)$. In other words, moves inside $q(x)$ include $p(\omega)$ modes, no part of $p(\omega)$ is uncovered or can not be reached by $q(\omega)$ moves. The rejection sampling is presented in algorithm 3.2

Algorithm 3.2 Rejection sampling algorithm

1: $i \leftarrow 1$
2: **repeat**
3: Sample $\omega_i \sim q(\omega)$ and $u \sim U_{(0,1)}$
4: **if** $u < \frac{p(\omega_i)}{Mq(\omega_i)}$ **then**
5: accept ω_i
6: $i \leftarrow i + 1$
7: **else**
8: reject sample ω_i
9: **end if**
10: **until** $i = N$

There are two ideas to retain from this algorithm:

1. acceptance/rejection idea.

2. the concept of a proposal distribution whose support includes the support of the target distribution.

The rejection sampling algorithm is of little usage, it suffers from many problems which we are not going to cover.

Monte Carlo Markov Chain

Rejection sampling, importance sampling, and their variants are memory-less samplers, they do not keep track of previously accepted/rejected samples, and this is one of their major drawbacks.

A good and feasible approach to maintain a record of generated samples, is using a Markov chain. The proposal distribution $q(\omega|\omega^{[\tau]})$ depends on the current state $\omega^{[\tau]}$. The sequences of generated samples $\omega^{[1]}, \omega^{[2]}, \ldots$ form a Markov chain, which mimics drawing from the target distribution $p(\omega)$. The aim is to construct a chain that explores the state space Ω, while spending more time in the most important regions.

A stochastic process $\omega^{[i]}$ is called a Markov chain of first order when the following conditional independence property holds for:

$$p(\omega^{[i]}|\omega^{[i-1]}, \ldots, \omega^{[1]}) = p(\omega^{[i]}|\omega^{[i-1]}) = T(\omega^{[i]}|\omega^{[i-1]}),$$

where $T(.,.)$ is the transition matrix.

A Markov chain for sampling is designed with, a target density, the required density $p(\omega)$. A first order Markov chain can simply be represented by an initial state probability $p(\omega)^{[0]}$ and a transition matrix T. A homogeneous Markov chain, has an invariant transition matrix. Starting from any initial distribution, and given a homogeneous chain, the chain will *stabilize* at the target density. This *stability* result plays a fundamental role in Monte Carlo Markov Chain (MCMC) simulation.

A distribution is *invariant* for a Markov chain, and also called *stationary*, if each step in the chain leaves that distribution invariant. For a homogeneous Markov chain, with a transition probability $T(\omega', \omega)$, the distribution $p^*(\omega)$ is invariant if:

$$p^*(\omega) = \sum_{\omega'} T(\omega', \omega) p^*(\omega')$$

A sufficient, but not necessary condition, to ensure that the required distribution $p(\omega)$ is invariant is to choose a transition probability that satisfies the detailed balance condition:

$$p^*(\omega)T(\omega, \omega') = p^*(\omega')T(\omega', \omega). \tag{3.6}$$

A Markov chain that satisfies the detailed balance condition is said to be *reversible*. An alternative condition to ensure an invariant distribution $p(\omega)$, is that the transition matrix T should respect the following properties:

- Irreducibility: For any state of the Markov state, there is a positive probability of visiting all other states.

- Aperiodicity: The chain should not get trapped in a cycle.

To sample from a given distribution, it requires not only that the desired distribution is invariant, but also *ergodic*. Ergodicidy means that $p(\omega)^{\tau \to \infty}$ converges to the required invariant distribution $p^*(\omega)$, irrespective of the initial distribution choice of $p(\omega)^{[0]}$. With ergodicity, the invariant distribution is then called the *equilibrium* distribution [12].

All these properties guarantee the convergence of the chain to the desired invariant distribution. More attention should be given for accelerating the speed of convergence by a proper selection of the proposition kernels, which defines the transition matrix.

Metropolis Hastings algorithm

The Metropolis Hastings (MH) is the most popular MCMC algorithm. Previously, the transition matrix $T(\omega, \omega')$ should respect the symmetric condition. Metropolis Hastings algorithm generalizes this case, and symmetry condition is not required any more. In a Metropolis Hastings algorithm, with an invariant distribution $p(\omega)$, and proposal distribution $q(\omega'|\omega)$, the sampling gives a candidate value ω' given the current value ω. The Markov chain then moves toward ω' with acceptance probability:

$$A(\omega, \omega') = min\left\{1, \frac{p(\omega')q(\omega|\omega')}{p(\omega)q(\omega'|\omega)}\right\},$$

otherwise, the chain remains at ω. The MH method is summarized in algorithm 3.3

Algorithm 3.3 Metropolis Hastings algorithm

1: Initialize $\omega^{[0]}$
2: **for** For $i = 0$ to $N - 1$ **do**
3: Sample $u \sim U_{[0,1]}$
4: Sample $\omega' \sim q(\omega'|\omega^{[i]})$
5: **if** $u < A(\omega^{[i]}, \omega') = min\left\{1, \frac{p(\omega')q(\omega^{[i]}|\omega')}{p(\omega^{[i]})q(\omega'|\omega^{[i]})}\right\}$ **then**
6: $\omega^{[i+1]} = \omega'$
7: **else**
8: $\omega^{[i+1]} = \omega^{[i]}$
9: **end if**
10: **end for**

Metropolis Hastings algorithm replaced the need for symmetric kernels by the *acceptance/rejection* phase, which was previously proposed in older sampling algorithms such as Rejection sampling. In Metropolis Hastings also kernels support should enclose the targeted invariant density.

The Metropolis algorithm assumes a symmetric random walk proposal $q(\omega'|\omega^{[i]}) = q(\omega^{[i]}|\omega')$, which reduces the acceptance ratio to:

$$A(\omega^{[i]}, \omega') = min\left\{1, \frac{p(\omega')}{p(\omega^{[i]})}\right\}$$

This algorithm does not required the normalization constant of the target distribution.

In section 3.1.1, we mentioned the interest of using simulated annealing. MCMC algorithm can also be used in a simulated annealing scheme to converge to the global minimum of a target distribution. The MCMC method will have a minor modification by inserting a simulated annealing scheme, it is summarized in algorithm 3.4. MCMC in a simulated annealing scheme, not only requires suitable proposal distributions but also an appropriate cooling schedule.

Algorithm 3.4 Metropolis Hastings algorithm with Simulated Annealing

Initialize $\omega^{[0]}$, set $T_0 = T$
for For $i = 0$ to $N - 1$ **do**
  Sample $u \sim U_{[0,1]}$
  Sample $\omega' \sim q(\omega'|\omega^{[i]})$
  if $u < A(\omega^{[i]}, \omega') = min\{1, \frac{p(\omega')^{\frac{1}{T}} q(\omega^{[i]}|\omega')}{p^{\frac{1}{T}}(\omega^{[i]}) q(\omega'|\omega^{[i]})}\}$ **then**
    $\omega^{[i+1]} = \omega'$
  else
    $\omega^{[i+1]} = \omega^{[i]}$
    Set T_{i+1} according to a chosen cooling schedule
  end if
end for

Multiple MCMC Kernels

MCMC has a major advantage, which is the possibility of using multiple kernels. In section 3.2.1, we mentioned that a good sampler should incorporate two proposal properties: *global proposal* to explore vast regions of the space and *local proposals* to discover finer details of the target distribution. This property which was missed in the birth and death algorithm, can be integrated in the MCMC using the multiple kernels idea.

If the transition kernel K_1 and K_2 have as invariant distribution $p(\omega)$ each, then the hybrid kernel K_1K_2 and the mixture kernel $\nu K_1 + (1 - \nu)K_2$ for $0 \leq \nu \leq 1$, are also transition kernels with invariant distribution $p(\omega)$.

3.2.3 Reverse Jump MCMC

The standard MCMC algorithm can not sample densities of varying dimensionality. A typical example would be model selection, where usually models are of different dimensions (different number of parameters). Point process models in image processing (multiple object detection) is another example, where we do not know the number of objects in advance. It was only after the work presented in [43] and [46] that it was possible to make a space dimension jump and with interesting kernels.

Informally, the basic idea is, by adding the birth and death kernel from [43] as an extra kernel to the MCMC, in [46] the author was able to make an MCMC algorithm which is capable of making a jump in the dimension space. For the mathematical details, please refer to this paper [46]].

3.3 Multiple Birth and Death

Let us consider a finite system of disks $\{(x_1, m_1), \dots (x_k, m_k)\}$ with elements x_i lying in a compact $K \subset \mathbb{R}^2$ and marks m_i lying in space $\mathbb{M}$. Therefore, an object is defined as $\omega_i = (x_i, m_i) \in K \times \mathbb{M}$ and $\omega = \{\omega_i, i = 1, \dots, n\}$ is a configuration. We consider a hard core distance ϵ between any two elements, such that $\forall i, j,\ d(x_i, x_j) > \epsilon$. The configuration space is then defined as:

$$\Omega = \bigcup_{n=0}^{N} \Omega_n, \quad \Omega_n = \{\{\omega_1, \dots, \omega_n\}, \omega_i \subset K \times \mathbb{M}\}\ , \tag{3.7}$$

where Ω_n is the subset of configurations containing exactly n objects ($\Omega_0 = \emptyset$), and N is the maximum number of discs of a certain fixed radius. We define a reference measure as the product of the Poisson measure $\nu(x)$ on K and the Lebesgue measures μ on the mark space $\mathbb{M}$:

$$d\pi_r(\omega) = d\nu(x) \prod_{i=1}^{n} (d\mu(m_i))\ .$$

This Lebesgue-Poisson measure $d\pi_r$ does not model any interaction. To model interactions, we define a Gibbs measure ν_β^V with respect to the Lebesgue-Poisson measure. We define an energy function $H(\omega)$ on the configuration space Ω. The Gibbs distribution $h_\beta(\omega)$ on Ω is defined by the density $p_V(\omega) = \frac{d\nu_\beta^V}{d\pi_r}(\omega)$ with respect to the Lebesgue-Poisson:

$$p_V(\omega) = \frac{z^{|\omega|}}{Z_{\beta,K}} \exp\{-\beta H(\omega)\}, \tag{3.8}$$

where β and z are positive parameters, and the normalization constant $Z_{\beta,K}$ is given by:

$$Z_{\beta,K} = \int_\Omega z^{|\omega|} \exp\{-\beta H(\omega)\} d\pi_r(\omega) = 1 + \sum_{n=1}^{N} \frac{z^n}{n!} \int_K \exp -\beta H(\omega) d(\omega)$$

The Gibbs density is defined in term of a certain energy function $H(.)$. The energy function $H(.)$ should formulate prior knowledge of the problem. For a typical counting problem, $H(.)$ can be given by a term which will attract the discs to objects, and a repulsive term which will penalize overlapping. Restricting to pairwise interaction, this energy function $H(\omega)$ is given by:

$$H(\omega) = \sum_{\omega_i \in \omega} H_1(\omega_i) + \sum_{(\omega_i, \omega_j) \subset \omega} H_2(\omega_i, \omega_j).$$

H_1 is usually referred to as the data term, it measures the fitness of a proposed object from the data point of view. To guarantee a sufficient distance between object (non-overlapping), $H_2(\omega_i, \omega_j)$ can be defined by:

$$H_2(\omega_i, \omega_j) = \begin{cases} \infty & \text{if } d(x_i, x_j) \leq C \\ 0 & \text{if } d(x_i, x_j) > C \end{cases} \tag{3.9}$$

3.3.1 Continuous Birth and Death Dynamics

In order to simulate the proposed model for finding the optimal configuration (for an object detection problem), we consider a multiple birth-and-death (MBD) dynamics. Which defines how our configuration will evolve through iterations until finding the optimal one. This evolution is accomplished through a generator L_β on continuous bounded functions f in Ω. This generator is given by [30]:

$$(L_\beta f)(\omega) = \sum_{\omega_i \in \omega} e^{\beta(H(\omega) - H(\omega \setminus \omega_i))}(f(\omega \setminus \omega_i) - f(\omega)) + z \int_{V(\omega)} (f(\omega \cup y) - f(\omega)) d(\omega_j)\,, \tag{3.10}$$

where $V(\omega) = K \setminus D(\omega)$, $D(\omega) = (\cup_{\omega_i \in \omega} \mathbb{B}_x(\epsilon)) \cap K$ and $\mathbb{B}_x(\epsilon)$ is the disc centered at x_i with radius ϵ. In other words, $V(\omega)$ is the free space on K. This operator, has two components, one that "adds objects" to the space and one that "deletes objects" from the current configuration. The first term of the operator (equation 3.10 is the deletion component, which is function of the energy difference $\Delta H(\omega_i)$ based on object removal (ω_i), where $\Delta H(\omega_i) = H(\omega) - H(\omega \setminus \omega_i) = E(\omega_i, \omega \setminus \omega_i)$. The second term of equation 3.10, is the birth component which is uniform on K. The birth intensity on the space $K \times \mathbb{M}$ can be defined as:

$$b(\omega, \omega_i) d\omega_i = z d(\omega_i)$$

and the death intensity on the configuration space is defined by:

$$d(\omega \setminus \omega_i, \omega_i) = e^{\beta E(\omega_i, \omega \setminus \omega_i)}.$$

Under this choice, the detailed balance condition holds:

$$\frac{b(\omega, \omega_i)}{d(\omega \setminus \omega_i, \omega_i)} = \frac{p_V(\omega)}{p_V(\omega \setminus \omega_i)} = z e^{\beta E(\omega_i, \omega \setminus \omega_i)}.$$

3.3.2 Discrete Approximation

Simulation of this continuous process is not feasible in practice, that is why we consider the discrete case of the birth-and-death process. The process is approximated by a Markov chain $T_{\beta,\delta}(n)$ where $n = 0, 1, 2, \ldots$ on the same space $K \times \mathbb{M}$. This Markov chain should allow in an infinitesimal time δ the following transitions:

$$\omega \to \begin{cases} \omega \setminus \omega_a \\ \omega \cup \omega_b \\ \omega \end{cases}$$

where ω_a and ω_b are configurations such that $\omega_a \subset \omega$ and $\omega_b \subset \omega$. Here we only consider the discrete case of the MBD algorithm, summarized in algorithm 3.5. Let δ be the intensity of the process (which will be detailed later); first we initialize the algorithm (step 1 and 2), by setting the starting values for δ and β (inverse temperature) used for the simulated annealing scheme, α_δ and α_β are coefficients to decrease the intensity of the process and the temperature respectively. Then the iterations start in step 4 till step 6, the algorithm keeps iterating until convergence. At iteration n, a configuration ω is transformed into $\omega'' = \omega^1 \cup \omega^2$, where $\omega^1 \subseteq \omega$ and ω^2 is a configuration such that $\omega^1 \cap \omega^2 = \emptyset$. The transition associated with the birth of an object in a small volume $\Delta v \subset K$ is given by [29]:

$$q_\delta(v) = \begin{cases} \Delta v\delta & \text{if } \omega \leftarrow \omega \cup \omega_i \\ 1 - \Delta v\delta & \text{if } \omega \leftarrow \omega \text{ (no birth in } \Delta v) \end{cases}$$

This transition is simulated by generating ω', a realization of a Poisson process of intensity δ. The death transition probability of an object ω_i from the configuration $\omega_{[n]} \cup \omega'$ is given by:

$$p_\delta(\omega_i) = \begin{cases} \frac{\delta\, a_\beta(\omega_i)}{1+\delta_a(\omega_i)} & \text{if } \omega \leftarrow \omega \setminus \omega_i \\ \frac{1}{1+\delta\, a_\beta(\omega_i)} & \text{if } \omega \leftarrow \omega \ (\omega_i \text{ survives}) \end{cases}$$

where

$$a_\beta(\omega_i) = \exp\big(-\beta[U(\omega \setminus \{\omega_i\}) - U(\omega)]\big)$$

This death probability is calculated for every $\omega_i \in \omega$, and the object ω_i is killed (removed) with probability $p_\delta(\omega_i)$.

The transition operator of the process in the discrete case has the following form:

$$(P_{\beta,\delta} f)(\omega) = \sum_{\omega_1 \subseteq \omega} \prod_{\omega \in \omega_1} \frac{1}{1 + a_x\delta} \prod_{x \in \omega \setminus \omega_1} \frac{a_x\delta}{1 + a_x\delta}\, \Xi^{-1}(\omega_1) \sum_{k=0}^{\infty} \int_{V_k(\omega_1)} \frac{(z\delta)^k}{k!} f(\omega_1 \cup y_1 \cup \ldots y_k) dy_1 \ldots dy_k,$$

where $\Xi(\omega')$ is a normalization factor for the conditional measure under a given configuration ω_1.

Algorithm 3.5 Multiple Birth and Death

1: $n \leftarrow 0$, $\omega_{[0]} \leftarrow \emptyset$
2: $\delta = \delta_{[0]}$, $\beta = \beta_{[0]}$
3: **repeat**
4: Birth: generate ω', a realization of a Poisson process of intensity δ
5: $\omega \leftarrow \omega_{[n]} \cup \omega'$
6: Death: For each $\omega_i \in \omega$, calculate the death probability $d(\omega_i) = \frac{\delta a_\beta(\omega_i)}{1+\delta a_\beta(\omega_i)}$, where $a_\beta(\omega_i) = e^{-\beta(U(\omega \setminus \omega_i)-U(\omega))}$
7: **until** Convergence, if not converged, set $\omega_{[n+1]} = \omega$, $\delta_{[n+1]} = \delta_{[n]} \times \alpha_\delta$, $\beta_{[n+1]} = \beta_{[n]} \times \alpha_\beta$, $n \leftarrow n + 1$ and go to "Birth"

Dynamic Process Parameters

The multiple birth-and-death dynamic has a small set of parameters to be initialized. These parameters are the temperature T (or the inverse temperature β), its scheduling parameter α_β, the process intensity δ and its scheduling parameter α_δ. For the simulated annealing, the multiple birth-and-death dynamic is not different from other stochastic models. The process we actualy simulate is given by the density $h_t(\mathbf{x}) \propto h^{\frac{1}{T_t}}(\mathbf{x})$. The temperature interact in the death probability where $d(\omega_i) = \frac{\delta a_\beta(\omega_i)}{1+\delta a_\beta(\omega_i)}$, and $a_\beta(\omega_i) = e^{-\beta(U(\omega \setminus \omega_i)-U(\omega))}$. Compared to the energy change this difference $(U(\omega \setminus \omega_i) - U(\omega))$ at iteration 0, an appropriate initial temperature value T_0 should be selected. The final value of the temperature T_∞ should tends to zero. For temperature scheduling, there exist many possibilities. The logarithmic scheduling is defined as follows:

$$T_t = \frac{D}{\log(t+1)} \, .$$

Using this scheduling, when D is larger that the maximum depth of a local minimum in the energy, it converges to the global minimum. Unfortunately, this scheduling is very slow for practical applications. Usually we recall to faster scheduling such as the geometric scheduling, which is defined by:

$$T_t = T_0 \times \alpha_T^t \, .$$

The second parameter will be the process intensity δ. This parameter δ represents the time-step for the discretization of the continuous process, and at the same time, this infinitesimal small time-step is proportional to the the number of added objects per iteration in average. Scheduling of δ is still an open problem. We apply the same geometric scheduling to the process intensity

$$\delta_t = \delta_0 \times \alpha_\delta^t \, .$$

There exists one condition that relates T to δ that has to be respected [30]:

$$\delta \exp\left(\frac{b}{T}\right) < const \, ,$$

where $b = sup_{\omega\in\Omega} sup_{\omega_i\in\omega}(U(\omega) - U(\omega \setminus \omega_i))$.

Setting those parameters $(T, \delta, \alpha_\beta, \alpha_\delta)$ is somehow a complex task. Improper selection will prevent the algorithm from convergence to the global minimum. The initial temperature T_0 should just be high enough to accept any move (addition or deletion of objects), but the algorithm should not stay for a long time in this phase, avoiding time loss. The temperature then should be decreased with the selected cooling schedule. And if the energy is very peaky, the cooling should be very slow.

Here we present a more complicated example with a *peaky energy*. Given a satellite image of a port presented by figure 3.1(a), the aim is to detect and extract the boats. This energy is peaky because of objects' orientation: there exists a special structure (alignment) in the data. We use ellipses to approximate boats' shapes. In figure 3.1(b), we present the detection result, assuming uniform prior on the orientation of the ellipses $\theta \sim U_{[0,\pi[}$. The detection result is not that good, because of the complexity of finding this global minimum. One possible solution, is to run the algorithm for much longer time. Another solution will be to reduce the complexity of this problem by using a non-uniform prior on the orientation parameter. Another alternative would be to add an extra term to the prior term for alignment between objects. Using a non-uniform prior on the orientation parameter, we get the result shown in figure 3.1(c). Figure 3.1(d) shows the detection result by adding an alignment term to the prior energy [9].

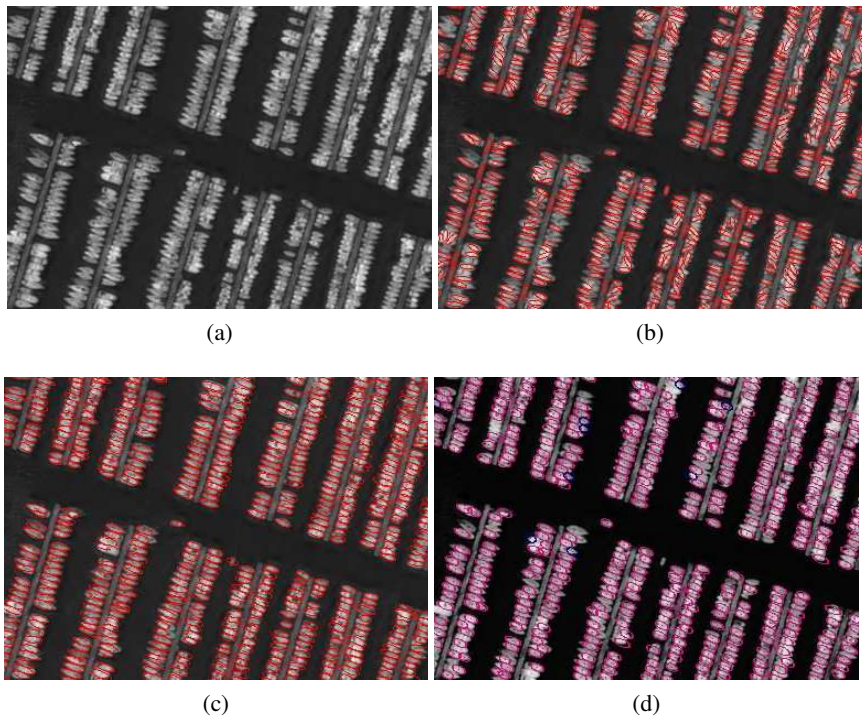
Setting the intensity parameter of the process is less crucial than setting the temperature. If the initial value of the intensity is too small, the algorithm will not find all objects, on the other side, if it is too high, it will slow down the convergence (extra useless computations) but will not deteriorate the final detection result. Also, if α_δ is too small, $\alpha_\delta \in]0, 1[$, then δ will decrease too fast, this will result to low detection.

3.3.3 Methods to speed up MBD

Different strategies can be used to accelerate the convergence speed of the MBD algorithm, we propose here two strategies [28].

Data Driven Birth

The configuration space $K \times \mathbb{M}$ is very large, and the density of objects varies a lot through the whole space. Taking a flamingo counting problem, by examining figure 3.2, we can see that for the space K, the density of object varies a lot. Some regions contain no flamingos at all, some regions contain only very few and other regions are very dense. Using a pre-processing hard segmentation method can harm the final result if the segmentation is erroneous or inaccurate. The alternative we propose is to roughly estimate the intensity



(a) (b)

(c) (d)

Figure 3.1: (a) photograph of vessels in France ©CNES. (b) Detection result when using a uniform prior on boats orientations. (c) Detection result when using a non-uniform prior on the orientations. (d) Result when adding an alignment term to the prior energy.

$\rho(x, y)$ (section 2.1.3).

For the flamingo case, we usually use a circle of fixed radius to scan the whole image. Calculating the fitness of this circle at this position, and this gives a very rough and fast estimate of the density. We call this estimated intensity, the *birth map*. It simply tells the algorithm "where" (in K) to spend more calculation time. We note this birth map by $B(s), s \in K$ defined on every pixel of the image. We use the birth map in the birth step, where birth of object becomes *data driven*. At a pixel s, the higher the value of $B(s)$, the higher the chance to propose an object centered in this pixel s. The values inside the birth map are normalized between 1 and 10, which means that: there is no place were we forbid birth, and the higher the $B(s)$ value the higher the birth probability. The MBD algorithm is then slightly modified when using a birth map, it is summarized in algorithm 3.6.

In this way we use a non-uniform prior on the image space K for data driven birth. The same idea applies to the parameters $\mathbb{M}$. Considering a building detection problem, the authors in [10] propose using a birth map that also encodes other parameters such as rectangle sizes and orientation at each image pixel. Rectangles are used for building detection, with parameters given by length, width and orientation. The birth map can encode information

Figure 3.2: An aerial image presenting a flamingo colony in Camargue, in Fangassie island.

Algorithm 3.6 Multiple Birth and Death

1: $n \leftarrow 0$, $\omega_{[0]} \leftarrow \emptyset$
2: $\delta = \delta_{[0]}$, $\beta = \beta_{[0]}$
3: **repeat**
4: Calculate the birth map: $\forall s \in K, b(s) = \frac{B(s)}{\sum_{s \in K} b(s)}$
5: Birth: generate ω', a realization of a non-uniform Poisson process of intensity δ and approximated intensity by $B(s)$
6: $\omega \leftarrow \omega_{[n]} \cup \omega'$
7: Death: For each $\omega_i \in \omega$, calculate the death probability $d(\omega_i) = \frac{\delta a_\beta(\omega_i)}{1+\delta a_\beta(\omega_i)}$, where $a_\beta(\omega_i) = e^{-\beta(U(\omega \setminus \omega_i) - U(\omega))}$
8: **until** Convergence, if not converged, set $\omega_{[n+1]} = \omega$, $\delta_{[n+1]} = \delta_{[n]} \times \alpha_\delta$, $\beta_{[n+1]} = \beta_{[n]} \times \alpha_\beta$, $n \leftarrow n + 1$

about these parameters, so in the birth map at pixel s, the generated parameters will be sampled from the estimated non-uniform prior over the length, width and orientation.

Sorting Objects

The second modification we propose alters during the death step. In this step, the death probability of each object $\omega_i \in \omega$ is calculated, and some of the objects will be killed $\omega'' = \omega \setminus \omega_1$. Given that ω is an unordered set, imposing an order for the death test will not change the final result after convergence, but will affect $\omega''^{[n]}$. We propose sorting the objects before the death step, starting by the objects with bad data term. Given a set of overlapping objects, starting by the ones with worth data terms, will increase the probability

that these objects will be removed and those with good data term will be kept specially after the penalization reduction coming from the prior term. While this sorting does not affect the final result quality, it was found to have a great impact on the speed of convergence of the MBD algorithm.

3.3.4 Convergence Test

The convergence test for this type of models with a highly non-convex function is harder to verify than for convex models which uses methods such as gradient descent algorithms. Usually, we consider that the algorithm has converged if the energy has not decreased for twenty successive iterations. The number of objects also can be used in a similar way: if it stays constant for n successive iterations, then the algorithm has converged.

3.4 Conclusion

The optimization issue plays a crucial role due to the high dimension of the configuration space. Until very recently, the only existing methods to optimize MPP models were fully stochastic.

Kernels Evolution in Point Process models:

- In a multiple object detection problem, the dimensionality of a set is unknown in advance, as it is the number of objects to be detected. This means that only samplers that can make dimensional jump can be used. This was only possible after the development of a *birth-death* kernel by Geyer et al. [43]. This kernel makes simple and global perturbations.
- MCMC is a flexible sampler, it can incorporate different kernels that are problem specific, but it can not make a dimensional jump. By integrating the *birth-death* kernel into MCMC, Green obtained a new sampler that can also make dimensional jump. This sampler is known as RJMCMC. With this sampler we can make global and local perturbations but only simple.
- Using the previous samplers, we can only make simple perturbations. The problem is that in practice those samplers are very slow. Due to the work of Descombes et al [3], it became possible to make multiple perturbations. They developed a *multiple-birth-death* kernel, and this is global multiple perturbations kernel.

Although those algorithms may be able to optimize very complex models with huge configuration space, they are very slow in practice. They are all full stochastic embedded in a simulated annealing scheme. It is only very recently that a semi-deterministic optimizer was introduced by Gamal et al [38]. This optimizer is known as Multiple Birth and Cut

(MBC).

In the next chapter, we describe the new MBC optimization method. It combines ideas from the (MBD) algorithm and from the Graph-Cut algorithm. This method holds two parts, the stochastic part to explore a very large configuration space, and the deterministic part which makes the optimal selection between an existing configuration and a newly proposed one.

Chapter 4

Multiple Birth and Cut Algorithm

Initially, samplers of similar models, such as Markov random fields (MRF), either stochastic or deterministic, were based only on *standard moves* within the framework of Metropolis Hasting dynamics, where only *one pixel* changes at a time. During the last decade, *multiple moves* methods emerged and most of them are based on graph cut techniques [16].

Point process samplers have also evolved from *simple perturbations* (standard moves) as in birth and death algorithm, where at each iteration, one object is either added to or removed from the current configuration [71, 87]. Such algorithms are extremely slow in image processing. Therefore, the Reverse Jump Markov Chain Monte Carlo algorithm [46] has been widely used for MPP in image processing [83, 74, 41] due to its flexibility, especially when using updating schemes such as Metropolis Hasting [46]. On the birth step, we now add moves such as split, translate, rotate, etc. This algorithm is much faster in practice than the Birth and Death algorithm. The main limitation is, it still treats one or two objects at a time and has a rejection rate. Later, the Multiple Birth and Death algorithm (MBD) was proposed allowing *multiple perturbations* [30].

In this chapter we present our contribution in the development of a new multiple perturbation optimization technique, named Multiple Birth and Cut (MBC) [38]. It combines ideas from MBD and the popular graph cut algorithm. The MBC algorithm's major advantage is the very reduced number of parameters as this algorithm does not involve the simulated annealing scheme and therefore the critical step of defining the cool schedule. We propose an iterative algorithm to explore the configuration space. We choose between current objects and newly proposed ones using binary graph cuts. In the second part of this work, we propose some modifications to this first algorithm in order to increase its speed of convergence. The algorithm starts by proposing a dense configuration of objects from which the best candidates are selected using the *belief propagation* algorithm. Next, this candidate configuration is combined with the current configuration using the binary graph cut.

We also discuss the main characteristics (that we consider to be essential) for the design of an efficient optimizer in the context of highly non-convex functions. Given that the probability density is multimodal, and given the size of the configuration space, an exploration phase is essential at the beginning of the algorithm. Next, the fine details of the density function should be discovered. Inside a third version of the MBC algorithem, we propose a new kernel to efficiently explore the different modes of the density, and new kernels to discover the details of each mode. We study the algorithm theoretically to express convergence speeds and to select its best parameters. We also present a simple and generic method to parallelize the optimization of a specific class of MPP models.

4.1 Graph Cut

In the last few years, a new approach of energy minimization based on graph cuts has emerged in computer vision. Graph cuts efficiently solved the optimization problem of certain energy families by finding either a global or a local minimum with a very high speed of convergence. This technique is based on a special graph construction from the energy function to be minimized. Finding the minimum cut of this graph is equivalent to minimizing the energy. The minimum cut is efficiently calculated using the max flow algorithm. We start by reviewing the max-flow min-cut theory [22].

4.1.1 Review of graph cut

Graph theory has emerged in many applications, and computer vision is not an exception. Many problem can be modeled by a graph, either directed or undirected, and graph algorithms can be applied to find solution of these problems.

A flow network can be represented by a directed graph. Finding the maximum rate at which we can ship material from the source s to the destination t respecting the network capacity, is known as the maximum-flow problem. Connection can be direct of indirect between cities, and they have capacities. Capacities can be *e.g.* the number of cars that can be leased from city A to city B. Once we know that information, one interesting question would be: How many *units* can we send from s to t through the existing network? With this modeling, the *flow* in a network can be anything, water, electric current, or information in a communication network. There exist efficient algorithms for solving this problem, where the most classic is the Ford and Fulkerson method. Another efficient algorithm is known as push-relabel, and has proven to be very fast in practice [22].

Flow networks

Let $G = (V, E)$ be a flow network which consists of a finite set V of vertices, a set $E \subset V^2$ of edges, with a capacity function c. Let each edge $(u, v) \in E$ have a nonnegative *capacity*

$c(u, v) \geq 0$. If E contains an edge (u, v), then $(v, u) \notin E$, and we disallow self-loop. There are two special nodes in a flow network, the source s and the sink t. We assume that any vertex $v \in V$ lies on some path $s \rightsquigarrow v \rightsquigarrow t$. A flow in G is a function $f : V \times V \to \mathbb{R}$ which satisfies the following [22]:

- Capacity constraints: For all $u, v \in V$, we require $0 \leq f(u, v) \leq c(u, v)$
- Flow conservation: For all $u \in V - \{s, t\}$, we require $\sum_{v \in V} f(v, u) = \sum_{v \in V} f(u, v)$

where $f(u, v)$ is a nonnegative quantity that represents the flow from vertex u to vertex v. Let the value $|f|$ of the flow f be defined by:

$$|f| = \sum_{v \in V} f(s, v) - \sum_{v \in V} f(v, s)$$

In a max-flow problem, given a flow network G, we wish to find the maximum flow value.

Ford-Fulkerson method

First, we call it method not algorithm because it encompasses several implementations with different running times. Ford-Fulkerson method depends on three notions: residual network, augmenting path, and cuts. We will explain these notions in the sequel. A coarse representation of the Ford-Fulkerson method can be summarized by algorithm 4.1 [22].

Algorithm 4.1 Ford-Fulkerson method

```
1: Initialize flow f ← 0
2: while There exists an augmenting path p in the residual network G_f do
3:     augment flow f along p
4: end while
5: return f
```

Residual networks:

Given a flow network G and a flow f, the residual network G_f consists of edges with capacities that represents how we can change the flow on edges of G. Let c_f be the residual capacity, defined by $c_f(u, v) = c(u, v) - f(u, v)$. If $c_f(u, v)$ is positive, the (u, v) edge from flow network G appears in the residual network G_f. During the algorithm iteration, it may be required to decrease a flow $f(u, v)$. To accomplish this decreasing, by sending back the flow, an new edge (v, u) will be added to G_f that did not exist in G. This edge can admit a flow in the opposite direction to (u, v), at most canceling the flow on (u, v). More formally, the residual capacity is defined by [22]:

$$c_f(u, v) = \begin{cases} c(u, v) - f(u, v) & \text{if } (u, v) \in E \\ f(v, u) & \text{if } (v, u) \in E \\ 0 & \text{otherwise .} \end{cases}$$

The residual network G_f is similar to the flow network G, with capacities c_f and the exception that it can contain an edge and its reversal.

Augmenting path:

A flow in a residual network provides a roadmap for increasing the flow in the original flow network. Let f be the flow in G, and let f' be the flow in the corresponding residual network G_f. We define $f \uparrow f'$, the augmentation of flow f by f' as:

$$f \uparrow f' = \begin{cases} f(u,v) + f'(u,v) - f(v,u) & \text{if } (u,v) \in E \\ 0 & \text{otherwise .} \end{cases}$$

Given a flow network G, and a flow f, an *augmentation path* p is a path from s to t in the residual network G_f. We can increase the flow in edge (u, v) up to capacity $c_f(u, v)$. The maximum value of this possible flow incrementation in an augmented path p is called the residual capacity of p, and is given by:

$$c_f(p) = min\{c_f(u,v) : (u,v) \text{ is on } p\}.$$

Cuts:

The Ford-Fulkerson method keeps incrementing the flow along augmentation paths until finding the maximum flow. How does the algorithm stops? The max-flow min-cut theorem, tells us that a flow is maximum if and only if its residual network contains no augmenting path.

A cut $c(S, T)$ is a partition of the vertices ($S \cup T = V$ and $S \cap T = \emptyset$), such that $s \in S$ and $t \in T$. If f is a flow in G, the net flow $f(S, T)$ is defined by [22]:

$$f(S,T) = \sum_{u \in S, v \in T} f(u,v) - \sum_{u \in S, v \in T} f(v,u).$$

The capacity of the cut $C(S, T)$ is given by:

$$c(S,T) = \sum_{u \in S, v \in T} c(u,v).$$

A *minimum cut* of a network is a cut whose capacity is minimum over all cuts of the network. The capacity counts only for edges going from S to T, while the flow counts for the flow from S to T minus the inverse flow from T to S.

The value of any flow f is bounded from above by the capacity of any cut of G, here is the proof:

$$\begin{aligned} |f| &= f(S,T) \\ &= \sum_{u\in S, v\in T} f(u,v) - \sum_{u\in S, v\in T} f(v,u) \\ &\leq \sum_{u\in S, v\in T} f(u,v) \\ &\leq \sum_{u\in S, v\in T} c(u,v) \\ &= c(S,T) \end{aligned}$$

The max-flow min-cut theorem states that: If f is a flow in a network $G(V,E)$, with source s and sink t, then the following conditions are equivalent:

1. f is a maximum flow in G
2. The residual network G_f contains no augmentation paths
3. $|f| = c(S,T)$ for some cut (S,T) of G

In other words, the max-flow is equal to the capacity of a minimum cut. The basic Ford-Fulkerson algorithm is summarized in algorithm 4.2 [22].

Algorithm 4.2 Ford-Fulkerson (G, s, t)

```
1:  for each edge (u, v) ∈ G.E do
2:      (u, v).f = 0
3:  end for
4:  while There exists an augmenting path p in the residual network G_f do
5:      c_f(p) = min{c_f(u, v) : (u, v) is on p}
6:      for each edge (u, v) ∈ p do
7:
8:          if (u, v) ∈ E then
9:              (u, v).f = (u, v).f + c_f(p)
10:         else
11:             (v, u).f = (v, u).f − c_f(p)
12:         end if
13:     end for
14: end while
15: return f
```

In steps (1-3) of the algorithm, the flow is initialized to zero. The steps (4-14), the while loop, it continuously finds an augmenting path p in G_f, and augment the flow f along p by the residual capacity $c_f(p)$. Special care is needed for the selection of the used method to find the augmentation path, otherwise the algorithm might be very slow.

4.1.2 Graph Cut in Computer Vision

The use of graph cuts in computer vision was first introduced in [47]. The authors demonstrated how a Maximum a Posteriori (MAP) estimate of a binary MRF can be exactly calculated using the maximum flow algorithm. They used the Ford-Fulkerson method for max-flow calculation. This work was a break through for the usage of models such as MRF. Previously, MAP estimate of Markov model was only possible either with global optimization methods as simulated annealing, or with local optimization methods as Iterative Conditional Mode (ICM). ICM being a local optimizer, is not of great usage because it requires a "good" initialization. Besides, nothing guarantees the quality of found local minimum. For the simulated annealing method, which is very powerful, and can deal with very complex models, unfortunately it is very slow in practice, which makes its usage constrained and limited in real computer vision applications. It was only after the introduction of the usage of maxflow (min-cut) [47] that the MRF usage has boomed in computer vision application.

After that, graph cut has been extensively used to compute the MAP solution for a large number of applications for discrete pixel labeling. It has been applied for solving many problems: image segmentation using geometric cues [15] or regional cues based on Gaussian mixture models [13], video segmentation [59] taking advantage of the redundancy between video frames (dynamic graph cuts), image restoration [51], stereo vision [79, 58], and many others.

Binary Image Segmentation

Here we describe a simple example of graph cut algorithm for solving an image processing problem. Many computer vision problems can be formulated as energy minimization problems. Energy minimization to solve the pixel labeling problem (*segmentation*) can be represented as follows: given an input set of pixels $\mathcal{P} = \{p_1, \ldots, p_n\}$ and a set of labels $\mathcal{L} = \{l_1, \ldots, l_m\}$, the goal is to find a labeling $f : \mathcal{P} \to \mathcal{L}$ which minimizes some energy function. We are interested in binary labeling, where $\mathcal{L} = \{0, 1\}$. A standard form of the energy function is [60]:

$$E(f) = \sum_{p \in \mathcal{P}} D_p(f_p) + \sum_{p \sim q} V_{p,q}(f_p, f_q) \tag{4.1}$$

where p and q are neighbor pixels. $D_p(f_p)$ is a function based on the observed data, it gives the cost of assigning the label f_p to pixel p. $V_{p,q}(f_p, f_q)$ is the cost of assigning labels (f_p, f_q) to pixels (p, q), where (p, q) are neighbors. $D_p(f_p)$ is always referred to as the *data term* and $V_{p,q}(f_p, f_q)$ as the *smoothness* or *prior term*.

Let us consider the graph shown in figure 4.1. Let $G = (V, E, c)$ be a directed graph which consists of:

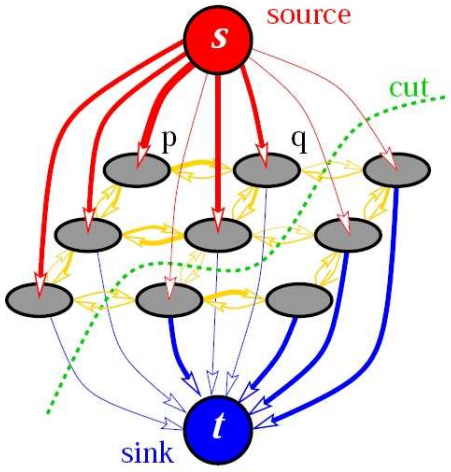


Figure 4.1: A simple 2D segmentation example for a 3×3 image, and the minimal cut indicated by the dotted line [72].

- a finite set V of vertices (or nodes), where each vertex represents a pixel from an image and two special nodes s and t (terminals)
- a set E of edges (links) of two types: *n-links* connecting pixel nodes, and *t-links* connecting pixels to s and t
- a cost assigned to each edge

The *n-links* edges are assigned the prior energy between neighbor pixels $V_{p,q}(f_p, f_q)$, the cost of assigning different labels to neighbor pixels. The *t-links* are assigned the data term energy $D_p(f_p) \in [0, 1]$ to the s terminal, and $1 - D_p(f_p)$ to the t terminal, which represent the cost of assigning label 0 or 1 to pixel p. By applying the max-flow algorithm on this graph G, and finding the minimal cut based on the assigned edge costs, the result corresponds to the MAP estimate of a binary segmentation of this MRF.

4.1.3 Convergence

In [47], the authors demonstrated how the MAP estimate of a *binary MRF* can be exactly calculated using the maximum flow algorithm. For this case (binary case), the algorithm converge to the global minimum of the energy. Later, it has been extended to multiple labels MRF [17, 50]. The multilabel extensions are known as α- expansion and $\alpha - \beta$ swap. α-expansion converge to a better energy minimum than $\alpha - \beta$ swap. Both converges to local minima, but α-expansion guaranteed that the found local minimum is within a known factor

of the global minimum, and this factor is given by:

$$c = \max_{p \sim q} \frac{\max_{\alpha \neq \beta} V(\alpha, \beta)}{\min_{\alpha \neq \beta} V(\alpha, \beta)}$$

Being able to use α-expansion or $\alpha - \beta$ swap depends on the properties of the prior (smoothness) term $V(.,.)$. The properties are:

1. $V(\alpha, \beta) \iff \alpha = \beta$
2. $V(\alpha, \beta) = V(\beta, \alpha) \geq 0$
3. $V(\alpha, \beta) \leq V(\alpha, \gamma) + V(\gamma, \alpha)$

If all conditions are met, then $V(.,.)$ is a metric, and α-expansion can be used, but if only the first two conditions are met, then only $\alpha - \beta$ swap can be used. As long as $V(.,.)$ is a metric, α-expansion should be used.

4.2 Multiple Birth and Cut

Graph cut algorithm usage in computer vision has raised during the last decade for two main reasons:

- Convergence speed of the algorithm to the energy minimum (either local or global).
- Does not require a stochastic relaxation algorithm (simulated annealing).

Although with the help of simulated annealing, we can solve very complex problems that no other method can solve, its usage has many drawbacks. Simulated annealing disadvantages are:

- Whatever the used scheduling (logarithm, geometric, ...), the algorithm is (very) slow.
- Setting the temperature parameter and the critical cooling schedule is a complex task and improper selection will prevent proper convergence.
- Since in practice we cannot use a logarithm scheduling, it is hard to judge the quality of the final result. One can never be sure that the found result corresponds to the global minimum, maybe the temperature decrement was too fast, or maybe the initial temperature was not high enough, *etc*.

4.2.1 Can we use graph cut to optimize a MPP models?

In [65], the authors presented an interesting graph model for a mosaic problem. The main goal of their work was to simulate classic mosaics from digital images. For good visual appearance, mosaics should satisfy constraints such as being non-overlapping. They generate a set of candidate layers containing tiles respecting the constraints and they "stitch" them in an iterative way. In the "stitching process", the selection between tiles of the current layer and a new candidate layer is solved by graph cut algorithm. This work was the inspiration source for our algorithm.

In order to use graph cut to optimize Markov point process models, we have to address the following questions:

- In usual MRF graph representation, each node represents a pixel. What does each node represent in a MPP model?

  → The graph representation is very flexible, while in most of the existing applications, nodes represent pixels, in our application, each node represents an object.

- Graph cut is generally presented to solve a labeling problem: What do these labels represent?

- How many labels do we need?

- What could be the labels in the optimization of an MPP model?

  → Having a set of candidate objects, binary label was the basic intuition. We propose using two labels: one refers to *keeping* an object and the other refers to *removing* an object.

- Do we satisfy the regularity (submodularity) condition?

→ In [60], the authors showed which class of energy function can be minimized by graph cuts. One important result from this paper is the *submodularity condition* (regularity) which must be satisfied, it is a necessary and sufficient condition. This condition represents the labeling homogeneity. For a two-neighbor pixel configuration (i, j) for which we assign labels $\{0, 1\}$, the condition is [60]:

$$E^{i,j}(0,0) + E^{i,j}(1,1) \leq E^{i,j}(0,1) + E^{i,j}(1,0) \tag{4.2}$$

which states that the energy (cost) required for assigning the same label to neighbor pixels should be less than or equal to the energy for assigning them different labels. This term applies to two overlapping objects. This inequality should be satisfied as soon as interaction between object exists. As stated in [47], it is possible to compute the *global minimum* of submodular binary energies.

Figure 4.2: The actual object is shown in green, and two candidate objects i and j with different degrees of fitness to the real object.

Taking the example shown in figure 4.2, in the background we see the real object in green, and we have two candidate objects in blue i and j. It is clear that object i should be kept and object j should be removed. To verify that we meet the regularity condition, we plug the proposed labels keep,remove in equation 4.2, which becomes:

$$E^{i,j}(\text{keep}, \text{keep}) + E^{i,j}(\text{remove}, \text{remove}) \leq E^{i,j}(\text{keep}, \text{remove}) + E^{i,j}(\text{remove}, \text{keep}). \quad (4.3)$$

The interpretation of the regularity condition in this form, it says that the energy required to keep both overlapping objects (i, j) or remove both, is less than or equal to the energy required to keep one and remove the other. Actually, we would prefer the following inequality:

$$E^{i,j}(\text{keep}, \text{remove}) + E^{i,j}(\text{remove}, \text{keep}) \leq E^{i,j}(\text{keep}, \text{keep}) + E^{i,j}(\text{remove}, \text{remove}). \quad (4.4)$$

This means that the energy required to "keep one" and "remove the other", is less than or equal to "keep both" or "remove both". We would like this term to select which candidate object of i and j is the best and keep it, and remove the other one. How to get rid of this problem?

4.2.2 From Supermodular to submodular

Let us consider two configurations ω and ω'. The solution we propose is the following, let us consider objects coming from two different configurations, let $i \in \omega$ and $j \in \omega'$. Let objects belonging to ω and ω' interpret the labels in a different (opposite) way. We consider a binary case, with two labels $\{0, 1\}$, interpreted as follows:

- if object $i \in \omega$ is assigned a label '0', this means this object is 'kept'

- if object $i \in \omega$ is assigned a label '1', this means this object is 'removed' (killed)

and the opposite for object $j \in \omega'$

- if object $j \in \omega'$ is assigned a label '1', this means this object 'kept'
- if object $j \in \omega'$ is assigned a label '0', this means this object is 'removed' (killed)

Using this reversed interpretation, we converted the supermodular terms into submodular terms to meet the submodularity condition. In addition to meeting the regularity condition, the solution corresponds to our goal.

4.2.3 MBC algorithm version 1

We propose an algorithm named Multiple Birth and Cut (MBC) [38, 37] that is described in the sequel, using figure 4.3 and summarized in algorithm 4.3.

Initialization: In step (1) of the algorithm we initialize our unique variable R, which represents the number of objects to be proposed at each iteration. This parameter R can easily be set, we set it to one fifth of the expected population size. Different initializations only affect the speed of convergence but not the final detection results. In step (2), we generate a candidate configuration of non-overlapping objects ω' which we set as initial configuration $\omega_{[0]}$. The set of non-overlapping ellipses ω' is generated as follows: each randomly proposed object ω_i (with position and mark) is rejected if it intersects at least one of the existing ellipses in the current ω', otherwise it is kept [84]. As an example, let the current configuration be $\omega_{[0]} = \{a, b, c\}$ presented in figure 4.3(a) in green. Now the algorithm starts iterating between the *Birth* and the *Cut* steps until convergence.

Birth: In the birth step we propose a new configuration ω', *e.g.*, $\omega' = \{d, e, f, g\}$ of "non-overlapping" ellipses, which are shown in figure 4.3(a) in blue. Note that objects $\{d, e\}$ have an overlapping of less than our defined threshold (10%), so we consider them as non-overlapping, as stated in section (6.6).

Cut:

- Graph construction: In the cut step, a graph is constructed for $\omega = \omega_{[n]} \cup \omega'$ as shown in figure 4.3(b), *each node represents an object* (ω_i), contrary to most graph cut problems where each node represents one pixel[1]. Edge weights are assigned as shown in tables 4.1 and 4.2. For each object $\omega_i \in \omega_{[n]}$, the weight assigned to the edge (t-link) to the source terminal is the data term $u_d(\omega_i)$ and $1 - u_d(\omega_i)$ to the sink,

[1] In a standard graph cut binary image restoration problem, for an image of size N^2, the required graph is of size N^2 (number of nodes). For a MPP problem, for an image of size N^2, the size of the graph is M (number of objects), where $M \ll N$.

while it is the inverse for $\omega_i \in \omega'$, it is $1 - u_d(\omega_i)$ to the source and $u_d(\omega_i)$ to the sink. For the edges between objects (n-links), we assign the prior term: ∞ if they are overlapping neighbors, otherwise it is zero.

- Optimizing: We apply the graph cut algorithm on this graph, to assign labels $\{0, 1\}$. The *key element* to satisfy the *submodularity* condition (equation 4.2), is that the labeling (generated by the graph cut optimization) is differently interpreted for the current configuration $\omega_{[n]}$ and the newly proposed one ω'. Our energy contains indeed *supermodular* terms, which are made *submodular* by inverting label interpretations. Label '1' for $\omega_i \in \omega_{[n]}$ stands for 'keep' this object, label '0' stands for 'kill' (remove) this object whereas for $\omega_i \in \omega'$ label '1' for stands for 'kill' this object and label '0' stands for to 'keep' this object.

  Based on this labeling interpretation, and on the defined interaction cost from table 4.2, the regularity condition (4.2) is satisfied since the left hand side will always be less than the right hand side which is equal to infinity.

Convergence test:
Optimization algorithm convergence for this type of model with a highly non-convex function is harder to verify than for convex models with gradient descent algorithms. Usually, we consider that the algorithm has converged if the energy has not decreased for twenty successive iterations. The number of objects also can be used in a similar way: if it stays constant for n successive iterations, then the algorithm has converged. For convergence to the global minimum of the energy, the energy stabilization is what should be considered. The reason for that is, this algorithm has a very high detection rate, most of the objects or all are detected very fast, so the number of objects stabilize quite fast, while the algorithm still requires more time for locally refining each object.

Algorithm 4.3 Multiple Birth and Cut

1: $n \leftarrow 0$, $R \leftarrow const$
2: generate ω' , $\omega_{[0]} \leftarrow \omega'$
3: **repeat**
4: Birth: generate ω'
5: $\omega \leftarrow \omega_{[n]} \cup \omega'$
6: Cut: $\omega_{[n+1]} \leftarrow \text{Cut}(\omega_{[n]} \cup \omega')$ (optimize with graph cuts)
7: **until** converged

Going back to equation 4.2, and the reversed labeling trick, we can see a strong condition that has to be respected. The condition is, with this representation, we can solve (get rid of) overlapping objects, only if we can separate a given configuration into two sets, where each set, internally, should not contain any overlap. Any newly proposed configuration ω',

(a) (b)

Figure 4.3: (a) Image containing a current configuration $\omega_{[n]}$ in green and a candidate configuration ω' in blue. (b) The special graph constructed for $\omega_{[n]} \cup \omega'$.

Table 4.1: Data Term

Config \ f_s	0	1
$\omega_i \in \omega_{[n]}$	$u_d(\omega_i)$	$1 - u_d(\omega_i)$
$\omega_i \in \omega'$	$1 - u_d(\omega_i)$	$u_d(\omega_i)$

should respect this non-overlapping condition. And of course any resultant configuration $\omega_{[n+1]}$ from the Cut-step, where $\omega_{[n+1]} \leftarrow Cut(\omega \cup \omega_{[n]})$, will respect this condition, and this is necessary, since $\omega_{[n+1]}$ will be used in the next Cut-step.
This condition limits the convergence speed of the MBC algorithm in this form.

4.2.4 Which algorithm to use?

If breadth-first search is used to find the augmentation path in the graph, the total running time of the Ford-Fulkerson algorithm will be $O(E|f|)$. An improvement of this basic algorithm is known as Edmonds-Karps algorithm, and it has a total running time of $O(VE^2)$. There exist another very fast algorithm known as *push-relabel* with a running time $O(V^2E)$.

Table 4.2: Prior Term

(f_s, f_r)	$V_{sr}(f_s, f_r)$
(0,0)	0
(0,1)	∞
(1,0)	0
(1,1)	0

We will use the Ford-Fulkerson method since the number of existing edges E during the MBC algorithm iterations are much smaller than the number of vertices V.

4.2.5 Analysis of the MBC algorithm

By analyzing this version of the MBC algorithm, and comparing it to the MBD algorithm, we found the following cons and pros:

Advantages:

1. It is an efficient algorithm to minimize a highly non-convex function
2. No simulated annealing and other associated parameters
3. The quality of the detection result is perfect (will be shown in the next chapter)

The disadvantages was that the speed of convergence was slower than the speed of the MBD algorithm, and the reasons are the following:

1. Proposed configurations ω' in MBD can be very dense but not in MBC, it has to respect the non-overlapping condition
2. The birth map is not yet integrated in the MBC algorithm, so we can not really make data driven proposal
3. Although the size of the graph is small, we construct a new full graph at each iteration

For clarification, the major speed limitation comes from the non-overlapping condition. Let us consider the following example, figure 4.4(a) represents an image with a circular object in the middle that we aim to detect. In figure 4.4(b), we present a set of candidate objects proposed by the MBD algorithm (5 objects) –in a single iteration– for the detection of this object. Unfortunately, for the MBC algorithm, respecting the non-overlapping condition, to propose the same set of candidate objects, it requires 5 iterations, as demonstrated in figure 4.5, which increases the number of required iterations.

4.2.6 Convergence of the MBC algorithm

The algorithm keeps iterating until convergence. Convergence can be evaluated by monitoring the number of objects or the energy of the configuration: when it becomes stable, we consider that the algorithm has converged.

(a) (b)

Figure 4.4: (a) The real object shown in green. (b) Five candidate objects proposed by the MBD algorithm in on birth-step.

Figure 4.5: Five iterations required by the MBC algorithm for proposing the same set of object that the MBD proposed in one iteration.

4.2.7 Global or local minimum?

Using graph cut, we obtain the global minimum for a configuration $\omega = \omega_{[n]} \cup \omega'$ at each iteration. Let the energy of the configuration ω at the nth iteration be $U^{[n]}(\omega)$, $U^{[n]}(\omega) \leq U^{[n-1]}(\omega)$, it is monotonically decreasing. The non-overlapping prior and the finite size of the image induce that the energy is lower-bounded. Therefore, we have a sufficient condition for our algorithm to converge at least to a local minimum.

In chapter 6, we will discuss more about the reached minimal energy on synthetic images as an empirical proof.

4.3 MBC algorithm version 2: Belief propagation

In this section, we will propose solutions to overcome the speed limitation that originates from the non-overlapping condition [37, 36].

Considering again the example demonstrated by figure 4.4, it is true that we can not give all these candidates objects to graph cut, but we can give graph cut the *best candidate* of those objects. Then graph cut combines this selected candidate with the existing ones and give an optimal solution.

We propose enhancing the generation phase by inserting a *selection phase* in the birth step, which allows adding more *relevant objects* in the birth step, thus reducing the number of iterations (convergence time). In the sequel we explain the modified MBC algorithm, which is summarized in algorithm 4.4.

4.3.1 Generation Phase:

In the birth step, the new algorithm generates a dense configuration Γ. This configuration has a special organization, where $\Gamma = \{X_1, X_2, \ldots, X_n\}$ and $X_i = \{\omega_i^1, \omega_i^2, \ldots, \omega_i^l\}$. Each X_i encodes l candidates from which only one will be kept, see figure 4.7(b). The aim of this organization is, instead of proposing a single object ω_i to detect a given object $\mathbf{o}_j$, we propose many objects at a similar location represented by X_i at each iteration and then select the most relevant object in X_i during the selection phase. The generation of Γ elements takes advantage of the *birth map* to speed up the process.

In figure 4.6(a), we present an example of a birth map. In the generation phase, we localize location in the birth map of high probability, and we define a circle for propositions, as shown in figure 4.6(b). Next, for each selected location (circle), we generate candidate objects thus creating X_i at each location, and by repeating this for the n selected location, we have generated Γ.

Algorithm 4.4 Multiple Birth and Cut

1: $n \leftarrow 0$, $R \leftarrow const$
2: generate ω' , $\omega_{[0]} \leftarrow \omega'$
3: **repeat**
4: Birth: generate Γ
5: $\omega' \leftarrow$ Select_from(Γ)
6: Cut: $\omega_{[n+1]} \leftarrow$ Cut($\omega_{[n]} \cup \omega'$)
7: $n \leftarrow n + 1$
8: **until** converged

4.3.2 Selection Phase

Now the question is raises of how to select the best candidate inside each X_i. If all the X_i were independent, then the selection of every $\omega_i^j \in X_i$ could simply be calculated based on the data term $u_d(\omega_i^j)$. However, if we consider a dense configuration of objects during the birth step, the independence hypothesis does not hold. It is still possible to only consider the data term for selection, and by the end remove objects that overlap.

Imposing the independence between every X_i, will limit the possible density of the final selected ω'. Although ω' should not contain overlapping objects, some objects of X_i and X_j can overlap before the *selection*. If the configuration Γ is dense, with cycles in the dependencies, *loopy belief* propagation algorithm can be used for the selection. Unfortunately, there is no guarantee on the quality of the obtained minimum, it is not a global optimization method.

We propose the optimal selection of ω' from an almost very dense configuration Γ. The idea is to generate Γ such that the interaction graph between sets X_i remains controlled. If with the interactions, we get a chain(s), we can make an optimal selection. The problem is that the chain form is a very strong constraint. If the interaction graph between sets X_i remains a tree (with no loops), the global optimum ω' can then be inferred rapidly on this tree using *belief propagation* [73].

4.3.3 Belief Propagation

Belief propagation is a particular case of dynamic programming, more precisely, it is a variation of Dynamic Time Warping suitable to trees instead of chains, often formulated with message passing. The core of the algorithm relies on the tree structure of the interactions between variables, *i.e.* if ω_1 is a leaf, it interacts with only one variable, ω_2 and therefore:

$$\inf_{\omega_1,\omega_2,\ldots,\omega_n} \left[\sum_i u_d(\omega_i) + \sum_{i\sim j} u_p(\omega_i,\omega_j) \right] = \inf_{\omega_2,\omega_3,\ldots,\omega_n} \left[\sum_{i>1} v_d(\omega_i) + \sum_{i\sim j>1} u_p(\omega_i,\omega_j) \right] \quad (4.5)$$

where $v_d = u_d$ except for $v_d(\omega_2) = u_d(\omega_2) + \inf_{\omega_1}\{u_d(\omega_1) + u_p(\omega_1,\omega_2)\}$. This optimization over ω_1 given ω_2 is easy to perform and rewrites the problem into a similar one but with one fewer variable. Repeating this trick n times solves the problem, with linear complexity in the number of variables.
Once a configuration Γ is generated, we apply belief propagation to select the best candidate inside each X_i, which gives the global optimum ω' of this configuration Γ. While generating Γ, the algorithm keeps track of the created neighborhood to verify that it always remains a *tree*.

The belief propagation algorithm consists of simple local message passing from leaves up to root (an arbitrary node), and given these messages, we calculate the Maximum a

Posteriori (MAP) for this Γ. The messages are given by:

$$m_{j\to i}^{max}(\omega^i) = \max_{\omega^j}\left(u_d(\omega^j)u_p(\omega^i,\omega^j)\prod_{k\in c(j)} m_{kj}^{max}(\omega^i)\right),$$

where $m_{j\to i}$ represents the message sent from node j to node i in the tree, $c(j)$ the children (neighbors) of node j, $u_d(.)$ the data term, and $u_p(.,.)$ the prior term, so the cost of overlapping between objects should be relative to the overlapping area between objects. Using this message, we compute the max value using any node i.

The generation and selection phase schedules are presented on figure 4.7. On figure 4.7(a), we present the current configuration $\omega_{[n]} = \{a, b, c\}$. In figure 4.7(b), the algorithm generates a dense configuration $\Gamma = \{X_1, X_2, X_3, X_4\}$. We apply the belief propagation on Γ to select only one (the best) from each X_i candidates ($\{\omega_i^0, \omega_i^1, \ldots, \omega_i^l\}$) as on figure 4.7(c) by $\omega' = \{d, e, f, g\}$. On figure 4.7(d), we present the combination of the current configuration $\omega_{[n]}$ and the *newly proposed* and *selected* ω' by $\omega = \omega_{[n]} \cup \omega'$ on which the graph is constructed for the Cut step. In figure 4.7(e) we present the tree structure (forest) for a much larger configuration, showing each X_i as a node, and the existing connections between them representing the neighborhood of each object (no loops), where belief propagation is computed on such a tree

4.4 Energy comparison

In this section, we demonstrate that after this modification of the data term (4.6), we still minimize the same energy using graph cut at each iteration.

For the graph cut algorithm, edge weights have to be non-negative, so we normalize the data term to become $\mathcal{Q}_d(d_B) \in [0, 1]$. For each ω_i, the data term becomes:

$$u_d^{GC}(\omega_i) = \frac{1 + u_d(\omega_i)}{2}\,. \tag{4.6}$$

Let U^{CG} be the energy given by the graph cut, with $\omega = \{\omega_{[n]} \cup \omega'\}$ where $\omega_{[n]} = \{\omega_1, \ldots, \omega_p\}$ and $\omega' = \{\omega_{p+1}, \ldots, \omega_q\}$. The energy of the whole graph is the sum of the data term edges and prior term edges:

$$U^{GC}(\omega) = U_d^{GC}(\omega) + U_p^{GC}(\omega)$$

where the data term is given by:

$$\begin{aligned} U_d^{GC}(\omega) = &\sum_{i=1}^{p}\left[\left(\frac{1 + u_d(\omega_i)}{2}\right)\delta_{f(\omega_i)=0} + \left(\frac{1 - u_d(\omega_i)}{2}\right)\delta_{f(\omega_i)=1}\right] \\ &+ \sum_{i=p+1}^{q}\left[\left(\frac{1 - u_d(\omega_i)}{2}\right)\delta_{f(\omega_i)=0} + \left(\frac{1 + u_d(\omega_i)}{2}\right)\delta_{f(\omega_i)=1}\right] \end{aligned}$$

and the prior term is given by:

$$U_p^{GC}(\omega) = \sum_{i=1}^{p} \sum_{j=p+1}^{q} u_p(\omega_i, \omega_j)\delta_{f(\omega_i)=0}\delta_{f(\omega_j)=1} \ .$$

$u_p(\omega_i, \omega_j)$ is defined as in table 4.2, then $U_p^{GC}(\omega) = U_p(\omega)$. The graph cut energy for the data term is given by:

$$\begin{aligned} U_d^{GC}(\omega) &= \sum_M \left(\frac{1 + u_d(\omega_i)}{2}\right) + \sum_D \left(\frac{1 - u_d(\omega_i)}{2}\right) \\ &= \sum_M u_d(\omega_i) + \sum_{M \cup D} \left(\frac{1 - u_d(\omega_i)}{2}\right) \\ &= U_d(\omega) + \sum_{M \cup D} \left(\frac{1 - u_d(\omega_i)}{2}\right) \\ &= U_d(\omega) + \mathcal{K}(\omega) \end{aligned}$$

where after optimization M is the set of objects that we keep and D is the set of objects that we kill. Thus minimizing $U_d^{GC}(\omega)$ is equivalent to minimizing $U_d(\omega)$ plus a constant $\mathcal{K}(\omega)$, function of the configuration ω but not of M. It becomes:

$$\underset{\omega}{\operatorname{argmin}}\ U_{GC}(\omega) = \underset{H}{\operatorname{argmin}}\ U(\omega)$$

where $H = \{u \in \Omega | u \subset \omega\}$.

4.5 MBC algorithm version 3

This work has been submitted to ECCV 2012.

What makes a good optimizer?

In this section we will discuss what we consider to be the main characteristics of optimizers [2, 3] for non-convex functions in the context of MPP models. We will discuss those characteristics to identify the limitations of the existing optimizers, and present a third version of the proposed MBC algorithm that overcome those limitations.

1. Global versus local proposals: Local proposal[2] means exploring the *fine details of the modes*, while global proposal means exploring the *different modes*. For a multimodal

[2]Proposal, also named move or perturbation, means making a modification to the current configuration $\omega_{[n]}$ (at iteration n)

distribution, and with only local proposals, based on the initialization (starting point), only one mode will be discovered, while using only global proposals will not capture the details the modes. For either simulation or optimization, given that the considered density is multimodal, both these properties are required.

2. Simple versus multiple perturbations: The perturbations are simple or multiple based on how the configuration changes between two iterations. If only *one object* is modified (changed, removed or added), then it is a simple perturbation. If *many objects* are modified per iteration, then it is considered as a multiple perturbation.

Holding those characteristics is essential to make a fast optimizer. It requires global and local perturbation kernels, and also to be able to make multiple perturbations (simple perturbation is implicite).

In this version of the MBC algorithm, we will propose an more efficient *multiple-birth-death* kernel that the one proposed in [36]. The proposed kernel makes more efficient global perturbations, and explore more efficiently the configuration space, while being very simple. We next show how local perturbation kernels can be integrated in the MBC algorithm to make it more efficient.

4.5.1 New selection method

We start by analyzing the behavior of the selection method based on belief propagation. We consider a 1D problem as illustrated in figure 4.8(a). Let $\omega_{[n]}$ be the current configuration, presented in blue in figure 4.8. Assume that the two empty spots should contain objects. Over each of the two empty spots a packet of objects is proposed. The packets are presented in red in the figure. The aim is to select the best candidate object inside each packet X_i. There exist some interactions (presented by the gray dotted lines) between some objects of different packets. Each object has a data term (cost) and a prior term (interaction cost). Let the set of proposed packets be defined by $\Gamma = \{X_0, X_1, \ldots, X_n\}$. The belief propagation algorithm makes the best selection from Γ (global optimal), based on the data terms and the prior terms. The algorithm ends by selecting the two candidate objects $\{\omega_1^1, \omega_2^1\}$, shown in green in figure 4.8(b).

While the MAP estimate using belief propagation gave the optimal selection from Γ, those selected objects were not necessarily optimal from the point of view of the current configuration $\omega_{[n]}$. As figure 4.8(c) illustrates, what was optimal for the current configuration $\omega_{[n]}$ was $\{\omega_1^2, \omega_2^3\}$ and not $\{\omega_1^1, \omega_2^1\}$. The difference is, $\{\omega_1^2, \omega_2^3\}$ is locally optimal, while $\{\omega_1^1, \omega_2^1\}$ is optimal over Γ.

The aim is to minimize the total energy, if we consider only the data terms, $u_d(\omega_1^1) + u_d(\omega_2^1) < u_d(\omega_1^2) + u_d(\omega_2^3)$, so ω_1^1 and ω_2^1 are *prefect* candidates, while ω_1^2 and ω_2^3 are *good* candidates. If we consider the total energy, prior term and data term, $u_d(\omega_1^1) + u_d(\omega_2^1) +$

$u_p(\omega_1^1, \omega_2^1) > u_d(\omega_1^2) + u_d(\omega_2^3)$.

We propose to make the selection only locally and *re-proposing* new objects instead of the rejected ones that did not respect the *non-overlapping* condition. It costs much more to get a perfect candidate than to get a good one. So keeping one of the two perfect candidates and re-proposing instead of the other is more efficient. In real applications, the percentage of rejected objects ranges between 5% and 10%. We keep iterating the algorithm until re-proposing this small percentage. The main reasons that makes this birth kernel gives a faster algorithm are:

1. It takes more time to propose an excellent candidate than a good one.

2. Let the size of Γ be $m \times n$, where m is the number of packets and n is the number of objects per packet. The complexity of belief propagation is $O(mn^2)$, while for local selection it is $O(mn)$.

3. Proposed configurations can be very dense, and they can form loops.

4. The belief propagation algorithm require extra computational cost to ensure that the proposed packets do not form any loop.

4.5.2 Local Perturbations Kernels

The existing multiple perturbation optimizers hold a single kernel, a *birth* kernel that proposes new objects. What is missed here is making local proposals.

Consider object ω_i to be the best candidate we obtained from the birth kernel to detect object $\mathbf{o}_i$. Object $\mathbf{o}_i$ has a position x_i and a set of markes m_i. If we consider the ellipse model in $\mathbb{R}^2$, the set of parameters of ω_i is $\Theta = \{x_i, y_i, a_i, b_i, \rho_i\}$. Given that object ω_i is a good candidate for the detection of object $\mathbf{o}_i$, this means that $\Theta_{\mathbf{o}_i}$ is close to Θ_{ω_i}. It is much easier to go from $\Theta_{\mathbf{o}_i}$ to Θ_{ω_i} (or closer) than to propose a new object that becomes a better candidate for object $\mathbf{o}_i$.

Let us consider four perturbation kernels, a schematic representation is shown if figure 4.9. One for perturbing the object position, a second to perturb the major axis, a third for the minor axis and a fourth one for the angle.

From the current configuration $\omega_{[n]}$ we randomly select a subset ω'' of objects. On each object $\omega_i \in \omega''$ we apply a perturbation kernel, $\omega'' \xrightarrow{perturbations} \omega'$. More than one kernel can be applied to each object. Kernels are selected randomly. Giving different weights to the kernels may be beneficial. This type of perturbation is usually referred to as *local random walk*. Those kernels where originally proposed in the context of RJMCMC sampler [75]. The new MBC algorithm is summarized in algorithm 4.5.

Algorithm 4.5 New Multiple Birth and Cut

$n \leftarrow 0$, $R \leftarrow const$
generate ω' , $\omega_{[0]} \leftarrow \omega'$
repeat
  Birth:
  Sample $u \sim \mathcal{U}_{(0,1)}$
  Based on u, select a kernel randomly
  $\omega' \leftarrow$ Apply selected kernel
  $\omega \leftarrow \omega_{[n]} \cup \omega'$
  Cut: $\omega_{[n+1]} \leftarrow \text{Cut}(\omega_{[n]} \cup \omega')$ (optimize with graph cuts)
until converged

Global proposals are mostly needed at the starting of the algorithm. We force for the first few iterations to only use of the *birth* kernel. Thereafter we alternate between all the kernels.

4.6 Algorithm analysis

4.6.1 Theoretical analysis

Notations.

Algorithm 4.5 alternates randomly two kinds of steps: global exploration, with probability p_G, and local exploration, with probability $p_L = 1 - p_G$. A global exploration consists in picking randomly $\alpha_G N$ locations in the image (using the birth map, where N is a rough estimate of the number of objects to be found, and $\alpha_G \in [0, 1]$ a proportion) and in testing n_G random marks at each of these locations. A local exploration consists in picking randomly $\alpha_L N$ already detected objects, and in testing n_L random small variations of their marks (*i.e.* exploring direct neighbors in the discrete space of marks M of angle, axes' lengths and precise location).

Preliminary step.

Before applying Algorithm 4.5, a series of successive global explorations is performed, in order to ensure that most of the image domain K has been covered and that the number of objects detected (even if with wrong marks) is close to N. The average proportion of the image still uncovered after k such steps, *i.e.* the average proportion of regions with area $\frac{|K|}{N}$ not visited yet, is $(1 - \alpha_G)^k$. For $\alpha_G = 60\%$ *e.g.*, $k = 14$ steps are sufficient to ensure that 99.9% of the image has been explored.

Global vs. local explorations.

Now, at step s of Algorithm 4.5, the average number of random tests already performed in the neighborhood (of area $\frac{|K|}{N}$) of any image point x is $v_G s$ with $v_G := \alpha_G p_G n_G$, and the average number of local explorations around any detected object is similarly $v_L s$ with $v_L = \alpha_L p_L n_L$. Thus the ratio of local tests over global tests at any location is $\frac{v_L}{v_G} = \frac{1-p_G}{p_G} \frac{\alpha_L n_L}{\alpha_G n_G}$. However, the probability that a global test is successful decreases with time: by construction it is at most the probability that a random try in M performs better than the previous $v_G s - 1$ ones, which is $\frac{1}{v_G s}$. Consequently, the expected number of local explorations between two big moves in M increases with time and is $\frac{v_L}{{v_G}^2 s}$, which means that local minima are more thoroughly explored with time.

Time complexity and Percolation.

We will now study the time spent at each step, in order to optimize the algorithm *w.r.t.* parameters v and α. The cost of any exploration step, global or local, is of the form $\alpha n N$, plus the cost of applying graph cut to a graph of approx. $2\alpha N$ nodes. The expected complexity of graph cut is much lower than its worst-case complexity, and is usually regarded as between linear and quadratic in the size of the graph, depending on problems. The graph here however is generally not connected and thus the complexity depends on the size of its connected components (the smaller, the lower). Since the graph is made of randomly selected objects with density α, we are precisely interested into *percolation* properties, to obtain the typical number and size of connected components [56]. The critical site-percolation density α_c, above which arbitrarily big components arise, depends on the neighborhood size and on the type of lattice the graph is extracted from. In our application case, the number of neighbors is at most 6, and one can lay an hexagonal lattice on the image, with step size the typical object size. We could also consider the Voronoï diagram associated to the objects already detected, which is relatively similar. In this hexagonal setting, α_c is known to be approx. 0.697... . Since α has to satisfy $\alpha < \alpha_c$, a very cautious choice is to set $\alpha \leqslant 60\%$. For such an α, the phase is *subcritical* and the probability of connected regions is known to decrease exponentially with their area A, as $e^{-A/R}$, where R depends on α (smoothly for α far from α_c). The typical region size is then R, and, by computing $\left(\int_A A^2 e^{-A/R} dA\right) / \left(\int_A e^{-A/R} dA\right)$, we obtain that the average graph cut cost (supposing quadratic complexity) is proportional to R^2. One can bound R by $R_M = R(0.6)$ if we restrict $\alpha \leqslant 60\%$. Then the average total graph cut cost at step s, which involves $2\alpha N$ objects in about $\frac{\alpha N}{R}$ connected components, is bounded by $\alpha N R_M$. Consequently, the complexity of step s is bounded by $C_S = \alpha N(n + R_M)$. Note that this bound does not depend on s. The expected clock time at the end of step s is thus
$t = (p_G C_S(G) + p_L C_S(L))\, s = (v_G + v_L + R_M[p_G \alpha_G + (1 - p_G)\alpha_L])\, N s$
which has to be compared to $v_G s$ and $v_L s$, the average number of explorations in the neighborhood of any image point.

Estimated time and precision when in an attraction basin.

Let us consider the case of a locally simple problem, in the sense that objects are located away enough from each other, in order to prevent any mutual interaction. Under this hypothesis, any suitable data energy will satisfy that each individual potential admits a unique minimum, *i.e.* that the attraction basin of any ground-truth object includes all possible marks located in the neighborhood of this object. We can then estimate the time needed to find the global minimum in the mark space M with the method above. After x random global explorations in an image region of area $\frac{|K|}{N}$, the minimum is at distance $\frac{|M|}{x}$ on average and thus then reachable in $\frac{|M|}{x}$ local explorations. Minimizing $x + \frac{|M|}{x}$ leads to $x = \sqrt{|M|}$. Thus we aim at minimizing t such that $v_G s \geqslant \sqrt{|M|}$ and $v_L s \geqslant \sqrt{|M|}$, w.r.t. all parameters. This leads to minimizing $t = \left[(1 + \frac{R_M}{n_G})(v_G s) + (1 + \frac{R_M}{n_L})(v_L s)\right] N$, which is solved by $v_G = v_L$, and by high values of $n_G = n_L >> R_M$ to reduce the relative cost of graph cuts (we choose $n = 10$ in practice since R_M observed is already low). The corresponding average time to find the optimal mark for an object is $2\sqrt{|M|}N$. More precisely, the probability that the optimal mark is not found for this object after time t is $p_N = \frac{|M|!}{(M-t/N)!\, M^{t/N}} \approx e^{-\frac{1}{|M|}\left(\frac{t}{N}\right)^2}$ using Stirling formula. Hence, the probability to have already found it at time $\sqrt{|M|}N$ is about 0.63, and at time $2\sqrt{|M|}N$ about $e^{-4} \approx 0.98$. Furthermore, the probability p to have found all N objects at time $t = \gamma\sqrt{|M|}N$ is $(1-e^{-\gamma^2})^N \approx e^{-Nexp(-\gamma^2)}$; conversely the time needed to reach a given probability p is $t = \gamma(p)N\sqrt{|M|}$ with $\gamma(p) = \sqrt{-\ln\frac{-\ln p}{N}}$. Furthermore, the expectancy of the proportion of the objects not found yet after time $t = \gamma\sqrt{|M|}N$ is $e^{-\gamma^2}$ (in the case of a spatially uniform birth map). Thus a reasonable total time to spend in the locally convex case to expect **all but one** objects and their optimal marks is then $\sqrt{|M|}N\sqrt{\ln N}$.

Fixing local minima.

The results above remain valid for any problem provided objects are correctly localized from the first steps. However, it may happen, in the case of an object category with very-varied sizes, that two contiguous small objects are proposed instead of a big one. In such a case, one needs a global exploration at the correct location with reasonably good marks. The required quality of the birth depends on the relative energy cost of the wrong couple of small objects w.r.t. the one of the correct detection. Given a lower bound $B > 0$ on this difference (based on misplaced edges *e.g.*), it is sufficient that a global exploration finds a mark with energy less than B + the optimal one. Let p_{GM} be the proportion of such marks within the local mark space M. The average time to find one of them is $\frac{1}{2p_{GM}}N$. With similar techniques as before, the time required to fix, with probability p, simultaneously k such local minima occurring in the image, is $\ln(1 - p^{1/k})\frac{1}{2p_{GM}}N$. Note that another, faster possibility would be to add a merge-and-split kernel dedicated to this kind of minima.

Semi-local minima.

Depending on the energy minimized to detect objects, and in the case of overlapping neighborhoods of objects, bad initial explorations may result in wrong detections half-way between two real objects. Such wrong detections are fixed quickly by Algorithm 4.5, when a global exploration is performed at one of these locations, which happens as frequently as $\frac{2p_G}{\alpha}$. What is more problematic is when such wrong detections form chains of misplaced objects, each one preventing its neighbors from evolving to its right place. If the chain is a tree of diameter d, then the expected time to remove it is only $d\frac{p_G}{\alpha}$. However if the chain comprises a cycle of length c, the only way to break it (if this local minimum is strong) is to wait for a simultaneous global exploration of all c sites, which happens as rarely as $\frac{p_G}{\alpha^c}$. Nevertheless, the appearance of such a cycle is possible only during the very few first steps, and the probability of such an event is as low as $p^c\alpha^c$ where p is the probability of one misplacement. Moreover, whenever a chain is broken, it cannot form again.

4.6.2 Algorithm Parallelization

Since in this paper we consider only the first class of MPP models, happily it can easily be parallelized. Given that the objects we are modeling, for *e.g.* buildings, are independent, the resultant configuration from the optimization algorithm also should contain independent (non-overlapping) objects. Overlapping only exists during the iterations of the algorithm.

Given an input image, it should be *split* into n partitions, where n can be the number of cores in a processor, or number of machines used to solve the problem. Two issues to consider for splitting this image:

1. Each time we divide the image, there should be an overlap equal to the size of the largest existing object.

2. In many real cases, dividing the image into two equal halves at each split may not be the optimal choice for load balance. We propose to first calculate the birthmap, then make the split based on this map.

After splitting, each part is processed independently. The last step of the algorithm will be to *join* the result on the different parts. We propose to make the optimal join using graph cut, as it is done in every iteration in the Cuts step of the MBC algorithm.

To illustrate, consider the small sample of a flamingo colony shown in figure 4.10(a). This image is split vertically into two partitions, and each is optimized independently using the MBC algorithm. In figure 4.10(b), we overlap the detection result of the two partitions, one in pink and the other one in green. The common margin contains two detection result. In this example, nine flamingos are detected by both runs. We propose to construct a graph, and to apply the same idea used in any of the MBC algorithms to select between two sets of configurations. This way of parallelization has the following advantages:

1. The cost of the fusion step is the of cost optimizing **one** small graph.

2. This algorithm should scale linearly.

3. Since no communication is required while optimizing each partition, it can be done on different cores or different machines. The communication occur only twice: once to assign the task, and once to get the result for fusion.

4. If an optimization algorithm other than MBC is needed, it can be used without any change, the fusion part will be the same.

4.6.3 Algorithm summary

In table we make a sort of summarized comparison between optimization method presented in this chapter for Markov marked point process models, precisely for simple interaction class. Simple interactions class is more concerned with the counting problems of independent objects, as opposed to, high interactions class that is concerned with problem such as roads detection where a lot of interaction and dependencies between segments exist. The first two algorithm can handle both classes of problems, while MBD, MBC to date only handle the first class of problem.

The optimization algorithms are ordered in the table by increasing speed of convergence. The first four elements of comparison concern speed of convergence. The fifth and sixth concerns the theoretical proof of global minimum of the energy for an algorithm. Next, we just want to point that the first three algorithms are fully stochastic, while the newly proposed ones are a combination of stochastic and deterministic algorithms. The birth step remains stochastic as for the MBD algorithm, while the cut (death) step became deterministic. Finally we state that the first three algorithms require simulated annealing framework, while the newly proposed ones does not require this.

Sampler \ Criteria	BD	RJMCMC	MBD	MBC 1	MBC 2	MBC 3
Local perturbation		√				√
Global perturbation	√	√	√	√	√	√
Single perturbation	√	√		√	√	√
Parallel perturbation			√	√	√	√
Convergence to global min	√	√	√			
Convergence to local min				√	√	√
Stochastic sampler	√	√	√	√	√	√
Deterministic sampler				√	√	√
Requires simulated annealing	√	√	√			√

4.7 Conclusion

In this chapter we have presented an efficient optimization algorithm to minimize a highly non-convex energy function which was previously solved within a simulated annealing scheme. We avoid the difficult task of setting the temperature and cooling parameters of the simulated annealing, and other parameters.

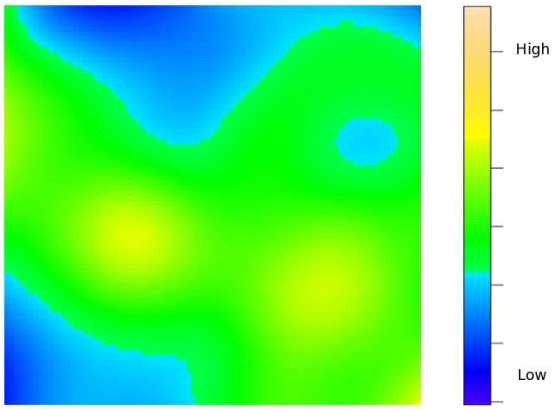
In a first version, we presented how the optimization problem of models such as point process models can be solved differently. The MBC algorithm is inspired from the MBD algorithm and takes advantage of the successful graph cut algorithm. In the first version we overcome many of the limitations of full stochastic algorithms for point process models, while having one major drawback which is the speed.

In a second version of the algorithm, we presented an optimized version using belief propagation. The aim is to optimize the newly proposed configuration at each iteration in order to obtain a relevant proposed configuration. On large scale problems, the second version is faster than the basic MBC algorithm.

While developing a third version, we made a rigorous analysis of the optimization problem and of our proposed algorithms. We discussed the essential characteristics that should be considered to design efficient optimizers. We presented the limitations of existing *birth-and-death* kernels, and, to overcome them, we introduced new kernels, both local and global.

In a theoretical analysis, we demonstrated the importance of both global and local perturbations. This analysis also showed how to select optimally the optimizer's parameters; in particular it quantified the optimal times spent in local and global explorations, in order to obtain the fastest convergence rates.

We also present a simple and generic method to parallelize the optimization of a specific class of MPP models.



(a) (b)

(c)

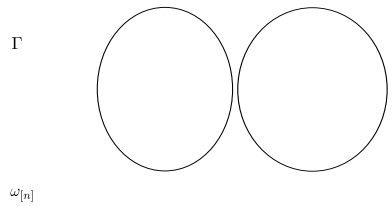
Figure 4.6: (a) This figure shows an example of birth-map; using spatstat. (b) On this birth-map, two positions of high values are localized, and region of proposals is defined. (c) A set of candidate objects is proposed in the defined proposal region.

(a) (b)

(c) (d)

(e)

Figure 4.7: (a) Current configuration $\omega_{[n]}$ in green. (b) Proposed dense configuration Γ. (c) Selected $\omega_{[n]}$ from the candidates of Γ. (d) The configuration $\omega = \omega_{[n]} \cup \omega'$ on which the graph is constructed for the Cut step. (e) A forest of trees of a large configuration from which we select one object per node using belief propagation.



(a) (b)

(c)

Figure 4.8: (a) A synthetic 1D problem. Current configuration $\omega_{[n]}$ shown in blue. Candidate objects shown in red. Interaction terms are presented with dotted gray lines. (b) Best candidates globally are shown in green. (c) Best candidates locally are shown in green.

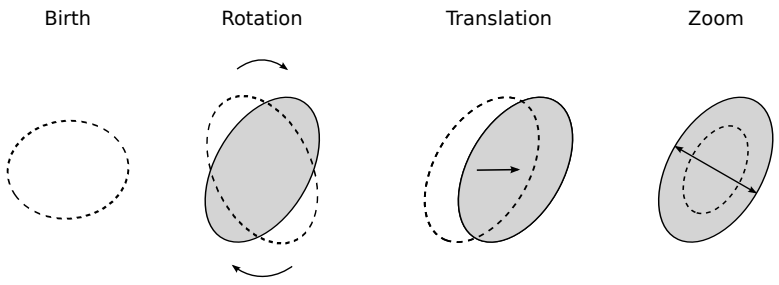
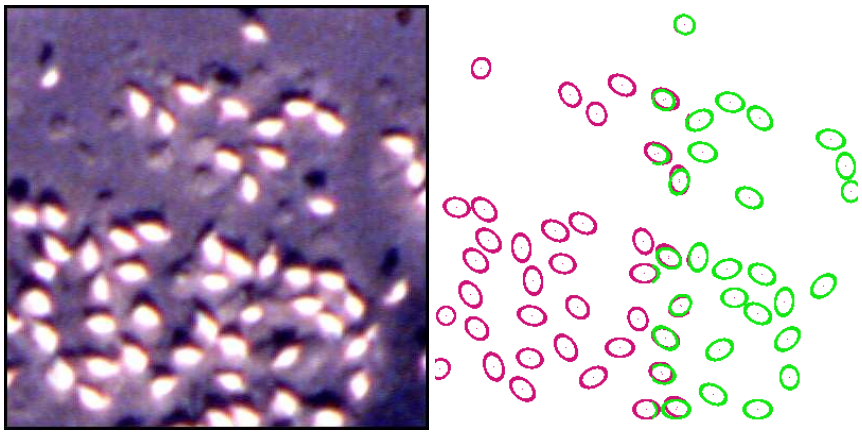


Figure 4.9: (a)



(a) (b)

Figure 4.10: (a) Shows the energy evolution vs. time on the 300 objects configuration. (b) Shows the energy evolution vs. time in log scale.

Chapter 5

3D Point Process

In all previously proposed MPP models, for applications such as building detection, road detection, *etc*, simple geometrical representation was enough given that the input data (images) is a top view perspective of the scene, while in other scenarios a more sophisticated geometrical representation is required.

5.1 Detection by simulation

Marked point process models used for image processing applications, such as the flamingo counting problem [25], is a direct problem, as opposed to an inverse problem. Considering the case of an airport, an example is shown in figure 5.1, where the aim of the application is detecting and counting the number of airplanes. It will not be an easy task to find a simple geometrical form to approximate an airplane shape. One possible solution is to use complex geometrical representation such as active contour models with some prior information [61] about the form to detect these airplanes, or parametric forms using *e.g.* B-splines. We propose the construction of a *3D CAD* [1] *model*. One of the reasons of this method is that any geometrical form can accurately be represented with a CAD model, no need for complex mathematical representation to represent such forms. For the airplane detection example inside an airport, we could simply use a 2D geometrical representation for airplanes, since we have a top view. The second reason for proposing using a 3D CAD model is, other than in the top-view case, a 3D model becomes of great interest to be able to recognize objects from any perspective. By looking at figure 5.3, we see the variability of geometrical form based on the imaging position with respect to the airplane. It is clear that it will be too hard to find a 2D geometrical form or even a set of 2D geometrical forms to cover every possible form.

[1] CAD: Computer Aided Design

Figure 5.1: A top view of Abu Dhabi International airport, internet site: *https://picasaweb.google.com/116589321075544181949/SaudiArabia#5213554461210564226*.

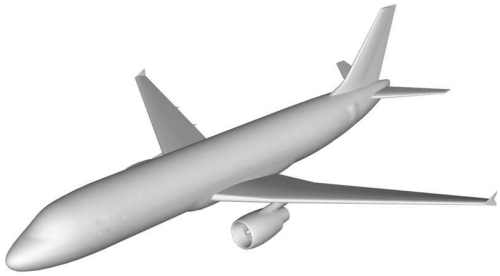


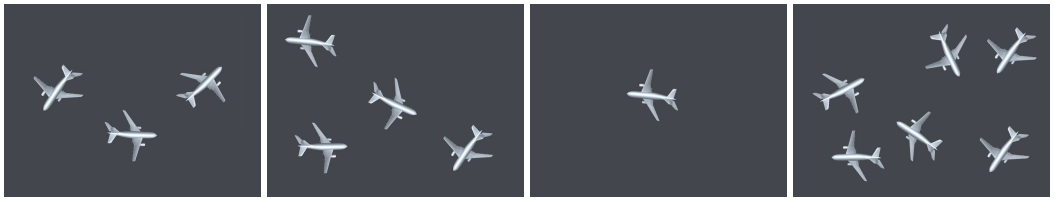
Figure 5.2: A 3D CAD model of the airbus A320 airplane.

The method we propose for object detecting is based on *3D scene simulation* on the GPU using OpenGL [23]. To detect the airplanes such as those in figure 5.1, we propose using a model such as the one in figure 5.2, and by simulating a configuration and projecting the scene onto the image plane, and measuring the similarity as schematically represented by figure 5.4, we can detect the airplanes. We propose using the MBD algorithm for the detection purpose, the algorithm will keep proposing random configurations, with different number of objects and with different parameters. The proposed configurations are projected onto the image plane, and the configurations are modified until convergence. To evaluate a proposed configuration, we measure the similarity between the projected image of the proposed configuration and the real image, by defining a data term and a prior term which penalize objects overlapping.

Figure 5.3: Airbus airplane from different perspectives.

Figure 5.4: Measuring the matching of a proposed object for the data term.

Another example of the multiple object detection problem is crowd counting [91]. Figure 5.6 presents two photos containing a crowd of people, with plenty of occlusions, and also with perspective effects (further people appear smaller). Here our proposed approach becomes even more interesting. We also propose using a 3D CAD model of human for the detection and counting purpose. By simulating the scene, we can propose people with the same number of objects, the same positions, orientation and with the same occlusions between them, taking the perspective effect into account for detecting people in further planes in the image.



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

Figure 5.5: Projected 3D scene of some candidate airplane configurations

(a) (b)

Figure 5.6: (a) A photo showing a crowd of people, with occlusions and perspective effects. (b) Another example of a crowd of people.

Figure 5.7(a) is a zoom on a sample from the crowd shown in figure 5.6(a). The sample is highlighted in red, yellow and green in figure 5.7(b). Most of the body of the first person (highlighted in red) is visible: the face, abdomen, shoulders, and two arms. We can see second person (highlighted in green) from his back, still we can recognize the back of the head, abdomen, shoulders, one of the two arms. Most of the third person is hidden, we can only see a part of the back of his head. The detection of the first two persons is an easy task, we can easily detect the head (face for the first), the arms, the shoulders, while for the third person it is much more challenging, how to detect it?

The detection of the third person (highlighted in yellow), counts on two aspects: first is the geometry of the visible part, second is the coloring. By using the scene simulation idea, if we correctly propose the first two, and then the third one, the visible parts from the projected 3D model by scene simulation will correspond to the visible parts of the person in the input image. Airplanes or people are just examples of detection applications where the usage of a 3D models and scene simulation can be of interest.

(a) (b)

Figure 5.7: (a) Zoom on a sample of a crowd of people. (b) Three highlighted person in red, green and yellow. First person, has a full frontal image, all the whole parts of the body are visible, second person is visible from his back, but still most of the body is visible, while for the third person, only the back of the head is visible.

5.2 3D Point Process

Let us consider a synthetic example. Let $\mathbf{x} = \{A, \ldots, J\}$ be a set of objects that we observe in an input image, as shown in figure 5.8(a). The actual scene is a 3D scene, those objects live in a 3D world, as presented in figure 5.8(b). We assume these objects live on the XY-plane at $z = 0$.

The optimization algorithm while iterating proposes configurations $\mathbf{x}'$ that are simulated using OpenGL, is then projected onto the image plane. The proposed configuration lives in a 3D world as in figure 5.8(b), and are then projected to obtain an image as in figure 5.8(a).

(a) (b)

Figure 5.8: (a) An image of a configuration after projection onto the image plane. (b) The corresponding 3D scene of this configuration.

5.2.1 Configuration Space

Let us consider a space $K \times \mathbb{R}^+$, where K is a closed, connected subset of $\mathbb{R}^2$, referred to as the real plane and $\mathbb{R}^+$ refers to the z-axis (objects height). We consider a perspective transformation, supposed as known, $\mathcal{P}$, such that $\mathcal{I} = \mathcal{P}(K \times \mathbb{R}^+)$. The data consists of $I = (i_s)_{s \in S}$, where S is a discretization of the image plane $\mathcal{I}$ and $i_s \in \Lambda$ is the color of pixel s. We consider configurations of an unknown number of objects in the real plane. We consider two positions for an object $p \in \{0, 1\}$ corresponding to standing up and lying (on the ground). Finally, an object is also characterized by its orientation ϕ with respect to the z-axis (in cylindrical coordinates). Therefore, an object is defined as $(x_i, m_i) \in K \times M$, where x_i represents the object location and m_i its mark, $M = \{0, 1\} \times [0, 2\pi[$. We consider a marked point process with points lying in K and marks in M.

5.2.2 Dependencies

The usual point process models used for image processing, such as buildings, roads, and others, uses 2D point process. These models have one type of dependency, it exists in the prior term between neighbor objects. What about 3D point process as for the configuration $\mathbf{x}$ shown in figure 5.8(b)? Does it have other dependencies? How is the neighborhood defined?

Figure 5.9: A top view perspective of a configuration.

5.2.3 Dependencies in the prior term

We need to think of the configuration of objects as $\mathbf{x}$ in terms of 3D representation, as shown in figure 5.8(b), and also from a top view perspective, as presented in figure 5.9. While in the flamingo counting problem presented in chapter 6, we penalized the overlapping objects in the input image, we can not penalize the overlapping with the presentation of figure 5.8(a). This overlapping is based on the camera position with respect to the scene, and there is no

reason to penalize objects in this view. The correct penalization should be considered in the 3D world, as in figure 5.8(b), or from a top view of the scene as in figure 5.9. The prior term neighborhood should be defined inside this representation. For the example shown in figure 5.9, the neighbor objects are: $C \sim D$, $H \sim F$ and $I \sim J$ (where $\sim$ means neighbor objects). In applications where objects are independent, for counting problems such as: flamingos, trees, people, airplanes, the prior term exists only to penalize any overlapping between object. The same applies to the 3D model for counting, if we used 3D model for people or airplanes or any other object, penalizing the overlapping is enough.

For a given configuration $\mathbf{x} = \{A, B, \ldots, J\}$, we want to know the energy required (cost) to add object J. The total energy $U(\mathbf{x}) = U_d(\mathbf{x}) + \gamma_p U_p(\mathbf{x})$ represents the prior energy and the data energy. We only consider here the prior term, which is decomposed as follows:

$$f(\mathbf{x}) = \exp\left(\sum_{x_i \in \mathbf{x}} V_1(x) + \sum_{\{x_i, x\} \in \mathbf{x}} V_2(x_i, x_j) + \ldots\right) \tag{5.1}$$

In our work, we only consider the second order term $V_2(.,.)$. The total prior energy of the configuration $\mathbf{x}$ is equal to: $f(\mathbf{x}) = V(C, D) + V(F, H)$. By adding object J, the total prior energy becomes $f(\mathbf{x}) = V(C, D) + V(F, H) + V(I, J)$. Neighborhood relation in the prior term is symmetric. We represent this prior term dependency of the configuration $\mathbf{x}$ by a graph, as shown in figure 5.10. Undirected edges are used to represent the undirected dependencies.

Figure 5.10: Dependency graph showing prior term dependencies.

5.2.4 Dependencies in the data term

Contrary to previous 2D point process models used for image processing, now there exist occlusions between objects. So the first question that comes to mind is **Are the data terms independent?**

5.2.5 Directional dependency

From the configuration presented in figure 5.8 and figure 5.9, we only consider $\{A, E\}$.

If we remove A, will the data term $u_d(E)$ change? If we remove E, will the data term $u_d(A)$ change?

For the first case, if we remove object A, considering the pixels belonging to A, some of them will become background and some of them will be added to object E. Will $u_d(E)$ change by gaining some extra pixels in this case or not? And the answer will be YES. For the second case, if object E is removed, will the $u_d(E)$ change? And the answer will be NO, non of E pixels can be affected to A, so $u_d(A)$ will not change. Let us formulate the energy change for the first case. Let ΔE_A be the energy change by the removal of object A, it is defined as follows:

$$\begin{aligned} \Delta E &= E_{\text{after}}(\omega) - E_{\text{before}}(\omega) \\ \Delta E_A &= E(\mathbf{x} \setminus A) - E(\mathbf{x}) \\ \Delta E_A &= E_E^{'} - \left(E_{A|\partial(A)} + E_E\right), \end{aligned}$$

where $E_E^{'}$ is the *new energy* of E after the removal of A, since object E gains some pixels so its data energy changes, and ∂ indicate the neighbors of an object. We refer to object A as a parent of object E, and E is a child of A.

Definition of a parent: An object I is a parent of an object J if and only if, when object I is removed from the configuration, object J gains some of its pixels.

This formulation means that if we want to remove A, not only should we check the data term of A, but also we have to check if the removal of object A enhances or deteriorates the quality of the data term of E (children of object A). The dependency in this case is *directional*, where $u_d(A)$ depends on A and E, while $u_d(E)$ does not depend on A. We represent directed dependency in the data term by directed edges in the graph presented in figure 5.11.

5.2.6 Moralization

Let us concentrate on only the three objects $\{A, B, E\}$. From section 5.2.5 we know that there is a directed dependency between the data term of A and E. The same applies to object B, ΔE_B also depends on object E. Since energy of A depends on E and energy of B depends on E, we conclude that: objects A and B are *dependent*, their data terms are dependent. A is a parent of object E, B is also a parent of E. Since A and B share a common child, then they are *not independent* any more, A and B became *dependent*. We refer to this new dependency by a *moralization* appearing on the dependency graph. The dependency graph

Figure 5.11: Dependency graph showing data term dependencies.

representing the full configuration $\mathbf{x}$ is presented in figure 5.12(a), where the moralization is represented by a dotted line. In figure 5.12(b), we present the full mixed graph representing all the prior and data term dependency. The delta energy associated with the removal of object A is given by:

$$\begin{aligned} \Delta E_A &= E(\mathbf{x} \setminus A) - E(\mathbf{x}) \\ &= E'_{B|\partial(B)} + E'_E - \Big(E_{A|\partial(A)} + E_{B|\partial(B)} + E_E\Big), \end{aligned}$$

(a) (b)

Figure 5.12: (a) Full dependency graph showing both prior and data term interaction. (b) Full dependency graph after moralization.

5.2.7 Markovianity in the dependency

The prior term dependency is markovian as previously presented in the 2D models. What about the data term dependencies, are they markovian or not?

From configuration $\mathbf{x}$, let us only consider three objects $\{b, f, h\}$. In figure 5.13(a) (e) and (i), we show the three possible cases for many dependencies in a chain.

- **First case:**
  The first arrangement of objects $\{b, f, h\}$ is presented if figure 5.13(a). In this case b is parent of f, and f is parent of h. The parent-child dependency between $\{b, f, h\}$ is present by the directed graph in figure 5.13(b). If object f is removed from this configuration as shown in figure 5.13(g), object b and h are independent, as shown in figure 5.13(h).

- **Second case:**
  In figure 5.13(e), b is a parent to f, f is a parent to h, as presented by a directed graph in figure 5.13(f). If f did not exist as show in figure 5.13(c), in this case, b becomes a parent of h, as presented in figure 5.13(d).

- **Third case:**
  In figure 5.13(i), b is a parent to f, b is a parent to h and f is a parent to h, as presented by a directed graph in figure 5.13(j). If object f is removed from this configuration as shown in figure 5.13(k), object b and h are still dependent, as shown in figure 5.13(l).

Conclusion:
A *neighbor* for the data term, is defined by the parent-child relation. If object i is parent of object j, then exist a directional neighborhood relation between i and j, that we note as $i \rightsquigarrow j$. For the three cases presented in figure 5.15, the dependence in the data term exists only if i is parent of object j, only if i is neighbor of object j. If i is parent (neighbor) of j, and j is parent of k, this does not mean that i is parent of k.

We consider an object based data energy, which means that the global data term is a sum of local terms applied on each object in the configuration. Unlike existing approaches using marked point processes for object detection, we cannot consider the data term associated with an object independently of the remaining objects in the configuration. Indeed, the data differ from the real plane where the objects are considered. It is the data of 3D objects living in this plane and we only have a perspective view of the scene, which introduces occlusions and therefore dependencies between the projected objects. Computing the data term requires projecting the scene on the image plane. This can be obtained rapidly using the GPU and OpenGL environment.

We consider a data term associated with the configuration ω given by:

$$U_d(\mathbf{x}) = \sum_{x_i \in \mathbf{x}} U(x_i|c(x_i)) \ , \tag{5.2}$$

where $c(x_i)$ denotes children of x_i, which depend on the camera position with respect to the scene. A child is an object which inherits his parent pixels if the parent was removed from the configuration.

New graphical representation

Here we present a new graphical representation to the point process for the 3D model and its dependencies. We use similar representation to the one used for hidden Markov chain and hidden MRF. In figure 5.14, we present the same configuration $\mathbf{x}$ in a different way. Each object is represented by two nodes, one for the prior term and the second is for the data term, somehow similar to the hidden node and observation node in hidden Markov chain and hidden MRF. In figure 5.14, red hollow nodes represent the object position and mark, the red node filled with violet color represents the data term node for this object. If two objects are neighbors, their two empty red nodes are connected with a green edge. If there exists a parent child dependency between two objects, the parent is connected to his child with a directed edge in green; this edge is between their data term nodes since the dependence is in the data term. If two objects share a common child, then by moralization, they become dependent, and they are connected by a green dotted line (connection between data term nodes).

5.3 Optimization

To optimize the density function to find the minimum that corresponds to the proper object detection, we apply the multiple birth and death algorithm.

5.3.1 Projections

Given that we have dependent data term as shown in equation 5.2, and given that the calculation of the data term is based on a projection of a simulated scene, each death step will require more than one projection (in general). We consider again the example 5.15. For this configuration $\{b, f, h\}$, the calculation of the change of energy with the removal of an object *e.g.* b is given by:

$$\begin{aligned} \Delta E_B &= E(\mathbf{x} \setminus B) - E(\mathbf{x}) \\ &= E'_{B|\partial(B)} + E'_E - \Big(E_{A|\partial(A)} + E_{B|\partial(B)} + E_E\Big), \end{aligned}$$

From this equation we conclude that for the calculation of change of energy, in addition to projection of $\{b, f, h\}$, we need a second projection $\{f, h\}$ (without b). The same for object

f, first we can not handle it at the same time as object b due to the data term dependency, secondly, since b has a child (h), we have to project the configuration without object b to be able to calculate the energy change associated with the removal of b. This leads us to the conclusion that is require a special way to handle all these dependencies during the death step of the MBD algorithm.

Algorithm 5.1 Multiple Birth and Death

1: $n \leftarrow 0$, $\mathbf{x}_{[0]} \leftarrow \emptyset$
2: $\delta = \delta_{[0]}$, $\beta = \beta_{[0]}$
3: **repeat**
4: Birth: generate $\mathbf{x}'$, a realization of a Poisson process of intensity λ
5: $\mathbf{x} \leftarrow \mathbf{x}_{(n)} \cup \mathbf{x}'$
6: Death: For each $\omega_i \in \omega$, calculate the death probability $d(x_i) = \frac{\delta a_\beta(x_i)}{1+\delta a_\beta(x_i)}$
7: **until** Convergence, if not converged, set $\mathbf{x}_{[n+1]} = \mathbf{x}$, $n \rightarrow n+1$, $\delta_{[n+1]} = \delta_{[n]} \times \alpha_\delta$, $\beta_{[n+1]} = \beta_{[n]} \times \alpha_\beta$, and go to "Birth"

5.3.2 Graph Algorithm For Death Step

Here we present our algorithm based on the undirected graph to accomplish the death step taking into consideration all dependencies. The directed graph is transformed to an undirected graph to represent all the dependencies, the directed version of the graph is used only for the parent-child information. Our algorithm is described in algorithm 5.2, with a sketched example in figure 5.16 for the configuration $\mathbf{x}$ of figure 5.8.

The algorithm starts by constructing an undirected graph G representing all the dependencies in ω. Next, the algorithm generates a queue $\mathcal{Q}$ with a topologically sorted version of G, based on a *method*. The selected *method* defines how the graph will be traversed, this affects the speed of convergence. From the many possible ways to sort this graph, we propose to start first by the objects which are closer to the camera. While this method does not give the minimum number of graph partitions, the reason for this is, given a proposed configuration ω, objects that are closer to the camera, partially or fully occlude (hide) objects behind them (children), this can *mislead* the evaluation (death) of their children if the algorithm tests the children first. In step 3 and 4, we create two auxiliary queues, $\mathcal{Q}'$, to hold tested objects, and their nodes color become black, and $\mathcal{Q}''$ to hold blocked nodes and their color become gray. Step 5 to 24, we loop until testing every object $(1, \ldots, \mathcal{N})$ in $\mathcal{Q}$ (in the graph G). In step 6, we start looping on element of $\mathcal{Q}$, if they are not blocked, then we add them to $\mathcal{Q}'$, and block the adjacent nodes. As shown in figure 5.16.(b), $\mathcal{Q} = \{B, C, E, F, A, D, G, H, I, J\}$, B is first added to $\mathcal{Q}'$, its color becomes black, and their adjacent nodes $\{A, E, F\}$ are blocked, then, we also add $\{C, G, I\}$ to $\mathcal{Q}'$, their color also become black, and we also block their adjacent nodes, all blocked nodes color become gray. Now all elements of $\mathcal{Q}$ are either blocked or in the testing queue $\mathcal{Q}'$. The delta energy associated with the proposed removal of node $\in \mathcal{Q}'$ is calculated, then the death probability is calculated (as in algorithm 5.1 step 6), and if

an element is killed, it is removed from $\mathbb{Q}'$. Now the graph has to be updated after the death (removal) of some nodes, the same for $\mathbb{Q}$, and $\mathbb{Q}''$ has to be reset to release blocked nodes. And the algorithm continues as in figure5.16.(d), the algorithm test $\{E, F, J\}$, then E and J are killed. Now A and B are independent, they were only dependent by moralization when they had a common child E. Finally A and G are tested, and both of them are killed. Now a full death step of algorithm 5.1 is accomplished.

Algorithm 5.2 Death using Graph

$G \leftarrow ConstructGraph(\omega)$
$\mathbb{Q} \leftarrow TopoligicalSort(G, method)$
$\mathbb{Q}' \leftarrow \emptyset$ (queue for tested nodes)
$\mathbb{Q}'' \leftarrow \emptyset$ (queue for blocked nodes)
while $Sizeof(\mathbb{Q}') \neq \mathbb{N}$ **do**
  while $\mathbb{Q} \neq \emptyset$ **do**
    $u \leftarrow Dequeue(\mathbb{Q})$
    if $u \in \mathbb{Q}''$ **then**
      continue
    end if
    $\mathbb{Q}' \leftarrow u$ (black color)
    $Breadth\_Block(G, u, \mathbb{Q}'')$ (gray color)
  end while
  $CalcDeltaEnergy(G \setminus \mathbb{Q}', G)$
  for all each $u \in \mathbb{Q}'$
  calc death prob (algorithm. 5.1) **do**
    if u killed **then**
      $\mathbb{Q}' \leftarrow \mathbb{Q}' \setminus u$
    end if
  end for
  Set all nodes $\in \mathbb{Q}'$ to blue
  $G \leftarrow UpdateGraph(\mathbb{Q} \cup \mathbb{Q}')$
  $\mathbb{Q} \leftarrow TopologicalSort(G, method)$, $\mathbb{Q}'' \leftarrow \emptyset$
end while

5.3.3 Camera Parameters

The projected image $I = (i_s)_{s \in S}$ is given by projecting every point from the 3D space world M_p^{3D} to the image space, that is obtained by:

$$i_s = A \times \begin{pmatrix} R & T \\ 0 & 1 \end{pmatrix} \times M_p^{3D} \ . \tag{5.3}$$

Matrix A is the intrinsic camera parameters and $(R\ , T)$ are the extrinsic parameters (rotation and translation). Camera parameters are required for projecting the configuration during the iterations of the optimization algorithm. Based on the image collection we started

from, we found the we have to define with our collaborators (ecologists) an imaging protocol in order to get appropriate type of images for our model.

In the next image collection (that we expected), is based on a defined protocol between us and the ecologist. We have inserted reference points (with known GPS coordinates, distances and angles) inside the real scene to be able to recover the camera parameters, specially the extrinsic parameters. For the semi-synthetic images, we already know all the parameters. For images taken before or without the protocol recommendations, we have to estimate those parameters. For the current time, we manually set those parameters. We plan to integrate an automated method for camera pose recovery [44]. It will be included as one phase of *detection* and one phase of *re-estimating the camera pose* and iterating between them until convergence.

5.4 Conclusion

In this chapter we introduced a novel method for multiple object detection in two sophisticated scenarios, the first is for situations with occlusions and perspective effects, and the second is for complex geometrical forms. We presented a method based on *3D scene simulation* for detection. Instead of approximating shapes with bounding boxes, we use accurate 3D geometrical models. The aim of this model is to present an new way of thinking for handling two very challenging problems: occlusions and perspective effects.

The idea that lies behind this proposed method is the following: We can only detect partially occluded objects, in other words non-representative parts, only if we propose those parts at the same positions, and we can do this by scene simulation.

We presented our modifications to the 2D model to handle the 3D case. New types of dependencies appeared between the data term energy of different objects, between parents and children. We also showed the added dependency by moralization that appears between parents who share a child. We showed how all those presented in the model can be presented with a nice mixed graph.

This model will be validated on the penguin counting problem in chapter 7.

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

(e) (f) (g) (h)

(i) (j) (k) (l)

Figure 5.13: The first column (a,e,i) presents objects B,F and H with different arrangements. The second column (b,f,j) presents the corresponding data term dependency graph of object in column one. The third column (c,g,k) present the same configurations of first column after the removal of object F. The fourth column (d,h,l) presents the corresponding data term dependency graph of object in column three.

Figure 5.14: This is a graphical representation of the dependencies of the configuration **x** introduced in figure 5.8. Each object in the configuration is represented by two nodes. The filled node (in violet) represents the data term, and the hollow node represents the prior term. (1) Data term nodes: Data term dependency is indicated by a blue directed edge from parent to child. Data term dependencies by *moralization* (become dependent) are connected by a green doted undirected edge (between the corresponding filled nodes). (2) Prior term nodes: Prior term dependencies is indicated by an undirected edge (between the corresponding hollow nodes).

(a) (b)

Figure 5.15: (a) A configuration of objects B, F and H. (b) The corresponding data term dependency graph.

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

(e) (f) (g) (h)

Figure 5.16: (a)-(h) An example showing the evolution of a configuration and the representing graph.

Part III

Applications

Chapter 6

Optimization methods comparison on a 2D MPP model

In chapter 3.1 we presented existing optimization methods as Birth and Death, Reverse Jump Monte Carlo Markov Chain, and Multiple Birth and Death. In chapter 4, we presented our proposed optimization algorithms, the MBC and the MBC-Opt algorithm. In this chapter we will compare the proposed algorithms to the fastest existing algorithm for the considered model, the MBD algorithm. The comparison will performed on a 2D Markov point process model using ellipses as geometrical form. The comparison first will be applied on synthetic data for which the solution is known. Then, these algorithms will be validated on a real problem, the flamingo counting problem. For graph-cut, we used the graph-cut code developed by Olga Veksler [60, 18, 16].

6.1 Marked Point Process

In this section we summaries the 2D MPP model presented earlier for seek of completeness. This model will be used for the comparison of the different optimization methods presented earlier in this these.

Let X denote a point process living on $K = [0, I_{max}] \times [0, J_{max}]$. X is a measurable mapping from a probability space $(\Upsilon, \mathcal{A}, \mathcal{P})$ to the set of unordered configurations of points in K. K is a closed, connected subset of $\mathbb{R}^2$. This mapping defines a point process.

In this chapter for instance, each object is modeled by an ellipse. Let $\mathbb{M}$ be the mark space, $\mathbb{M} = [a_{min}, a_{max}] \times [b_{min}, b_{max}] \times [0, \pi[$, where a and b are the length of the major and the minor axes respectively, for which we define a minimum and a maximum value, and $\theta \in [0, \pi[$ is the orientation of the ellipse. The geometry of the shape is represented by the mark m_i associated with each point x_i. Therefore, an object is defined as $\omega_i = (x_i, m_i) \in K \times \mathbb{M}$. We consider a marked point process with points in K and marks in $\mathbb{M}$, the configuration

space is then defined as:

$$\Omega = \bigcup_{n=0}^{\infty} \Omega_n, \quad \Omega_n = \{\{\omega_1, \ldots, \omega_n\}, \omega_i \subset K \times \mathbb{M}\} \,, \tag{6.1}$$

where Ω_n is the subset of configurations containing exactly n objects, and $\omega = \{\omega_i, i = 1, \ldots, n\}$. We define a reference measure as the product of the Poisson measure $\nu(x)$ on Υ and the Lebesgue measures μ on the mark space:

$$d\pi_r(\omega) = d\nu(x) \prod_{i=1}^{n} (d\mu(m_i)) \,.$$

The MPP is then defined by a density with respect to this measure:

$$d\pi(\omega) = f(\omega) d\pi_r(\omega). \tag{6.2}$$

Markov Point Process. Among MPP, Markov (or Gibbs) point process are of particular interest for applications in object detection. The density of the process is then written as the sum of potentials over interacting objects (cliques):

$$f(\omega) = \frac{1}{Z} \exp[-U(\omega)] \tag{6.3}$$

where [5]:

$$U(\omega) = \left(V_0 + \sum_{\omega_i \in \omega} V_1(\omega_i) + \sum_{\{\omega_i, \omega_j\} \in \omega} V_2(\omega_i, \omega_j) + \ldots \right) \tag{6.4}$$

Z is the partition function (normalizing constant), and V_k the potentials of order k. Minimizing the energy $U(\omega)$ corresponds to the target configuration. This energy takes into account the interactions between geometric objects U_p (prior energy) and a data energy U_d to fit the configuration onto the image:

$$U(\omega) = U_d(\omega) + \gamma_p U_p(\omega)$$

where γ_p is the weight assigned to the prior term which can be estimated as in [20].

6.1.1 Prior

The possibility to introduce prior information is a major advantage of the MPP framework. This regularizes the configuration to match the real objects taking into consideration the image defects, due to, *e.g.*, image resolution or noise. In this chapter, we only concider simple interactions, existing mostly in counting problems such as tree counting, flamingo counting. In those applications, the interaction is simply reduced to a non-overlapping term.

Figure 6.1: The overlapping coefficient between two objects

Since our objects (flamingos) should not overlap in reality, we penalize overlapping. Let $\mathcal{A}(\omega_i, \omega_j) \in [0, 1]$ represent the overlapping coefficient between two objects, defined as the normalized area of intersection, as shown in figure 6.1 and proposed by [25]:

$$\mathcal{A}(\omega_i, \omega_j) = \frac{A(\omega_i \cap \omega_j)}{\min\left(A(\omega_i), A(\omega_j)\right)} \tag{6.5}$$

where $A(\omega_i)$ is the area of object ω_i. Let us consider a clique $\{\omega_i, \omega_j\}$, then the prior energy of this local configuration is given by:

$$u_p(\omega) = \begin{cases} 0 & \text{if } \mathcal{A}(\omega_i, \omega_j) < 0.1 \\ \infty & \text{if } \mathcal{A}(\omega_i, \omega_j) \geq 0.1 \end{cases} \tag{6.6}$$

which means that we do not allow a configuration with an overlapping coefficient greater than 10%. The total prior energy of the configuration is then given by:

$$U_p(\omega) = \sum_{\omega_i \sim \omega_j} u_p(\omega_i, \omega_j),$$

where $\sim$ is a symmetric reflexive relation used to determine the neighborhood of an object, and defined by the intersection of ellipses.

6.1.2 Data term

Assuming the independence of the data term of each object, the data term energy of a configuration ω is given by:

$$U_d(\omega) = \sum_{\omega_i \in \omega} u_d(\omega_i) \tag{6.7}$$

Figure 6.2: Ellipse modeling a flamingo and the background around it to measure the relevance of the proposed object

The term $u_d(\omega_i)$ is the output of a local filter, evaluating from the data point of view the relevance of object ω_i. The object contains information on both its location and its shape. The data term can, thus, be interpreted as an adaptive local filter selecting or favoring a specific shape and object depending locally on the data. For the selected flamingo example, as presented in figure 6.2, each flamingo can be modeled as a bright ellipse surrounded by a darker background. For an object $\omega_i = (x_i, m_i)$, with marks $m_i = (a, b, \theta)$, we define the boundary $\mathcal{F}(\omega_i)$ as the subset of K, between the ellipse ω_i border and a concentric one ω_i', with marks $m_i' = (a + \rho, b + \rho, \theta)$. This boundary represents the background and we evaluate the contrast between the ellipse interior and the background. To evaluate the distance $d_B(\omega_i, \mathcal{F}(\omega_i))$, we assume that the interior of the ellipse and its background have Gaussian distributions with parameters (μ_1, σ_1) and (μ_2, σ_2) respectively, which are estimated from the image. We compute a modified Bhattacharya distance between them as follows [25]:

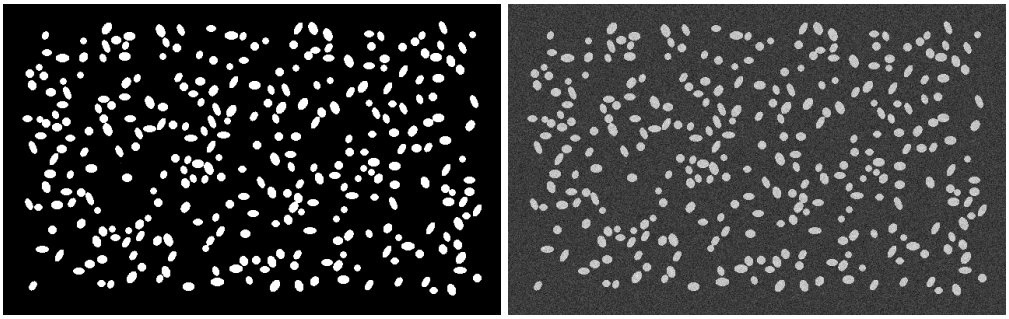
$$d_B(\omega_i, \mathcal{F}(\omega_i)) = \frac{(\mu_1 - \mu_2)^2}{4\sqrt{\sigma_1^2 + \sigma_2^2}} - \frac{1}{2}\log\frac{2\sigma_1\sigma_2}{\sigma_1^2 + \sigma_2^2}.$$

The data energy $u_d(\omega_i)$ associated with object ω_i is then given by:

$$u_d(\omega_i) = \mathcal{Q}_d(d_B(\omega_i), \mathcal{F}(\omega_i)) \tag{6.8}$$

where $\mathcal{Q}_d(d_B) \in [-1, 1]$ is a quality function which gives positive values to small distances (weakly contrasted object) and negative values (well contrasted) otherwise [25]:

$$\mathcal{Q}_d(d_B) = \begin{cases} (1 - \frac{d_B}{d_0}) & \text{if } d_B < d_0 \\ \exp(-\frac{d_B - d_0}{D}) - 1 & \text{if } d_B \geq d_0, \end{cases}$$



(a) (b)

Figure 6.3: (a) A synthetic image with 300 non-overlapping ellipses with dimensions $[0, \pi[= [5, 10] \times [5, 7]$. (b) The same image after a Gaussian noise addition.

where D is a scale parameter calibrated to 100 and d_0 is a threshold that is estimated either for the whole image or for each region, as detailed in [25].

6.2 Results on synthetic data

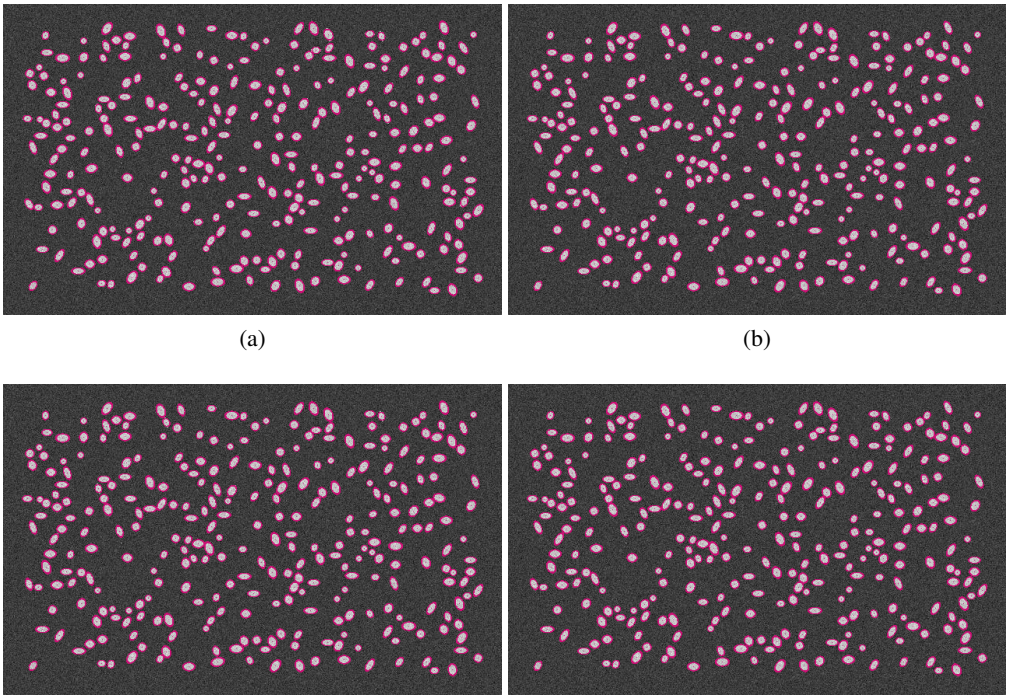
In this section we present a comparison of four optimization algorithms: MBD, MBC1 (the basic one), MBC2 (with Belief Propagation), MBC3 (using the new birth kernel and local perturbations kernels). The aim of this comparison is twofold, first to show how the algorithms scale with the problem size, second to present which type of energy minimum each algorithm can reachs.

We are testing the three algorithms on three samples containing respectively 300, 1000 and 10000 objects. These samples are generated by the same model introduced in section 6.1. The comparison will present total energy evolution of configurations versus time for each algorithm, and will present the object detection rate. The mark space for the ellipse parameters is $\mathbb{M} = [a_{min}, a_{max}] \times [b_{min}, b_{max}] \times [0, \pi[= [5, 10] \times [5, 7] \times [0, \pi[$

6.2.1 Sample of 300 objects

Figure 6.3 presents a sample of 300 objects. On the left of the figure, ellipses are filled with white color, and background is black colored. On the right part of the figure, the same sample is presented after Gaussian noise addition. The ellipses noise is generated from a Gaussian distribution of mean zero and standard deviation is equal to 2, $\mathcal{N}(0, 2)$, and the background is generated from a Gaussian distribution $\mathcal{N}(20, 2)$.

We use the following set of parameters: for the MBD algorithm, temperature $T = \frac{1}{\beta} = 1/50$, $\Delta\delta = 0.9997$, $\Delta T = 0.999$ and for 20000 iterations. For the MBC1 algorithm: the



(a) (b)

(c) (d)

Figure 6.4: (a) Detection result using MBD algorithm on the synthetic sample of 300 objects. (b) Detection result using MBC1 algorithm. (c) Detection result using MBD2 algorithm. (d) Detection result using MBD3 algorithm.

number of objects proposed per iteration $R = 120$, for 400,000 iterations. For the MBC2 algorithm: $R = 120$, while for the first few iterations, we set $R = 3 \times R$ (for faster convergence), number of candidate per location (we call this parameter packet size) equals 8, and number of iterations is 40,000. For MBC3, we use the same parameters as for MBC2, and for this sample size we only use the new birth kernel, we do not use the local perturbations kernels.

Figure 6.4 shows the visual detection result on the 300 objects sample. Pink ellipse with the detected position and parameter of each object is overlayed on each detected object. On a sample of this size, we can hardly see any visual difference between the detection results of the three algorithms, all of them gives a very good result.

For this sample, figure 6.5(a,b) shows the energy of configuration evolution during iterations with time for the four algorithms. The minimum value of the energy is $E(\omega) = -249.981$, we know this true value since we consider a simulated image with known set of parameters. From the energy curves shown in figure 6.5(a,b), we can conclude that after 130 seconds:

- For this small sample, all the algorithms have no problem converging to the energy minimum.
- MBD, as every algorithm in a simulated annealing framework, will give high energy value since the algorithm accepts both good and bad objects from their energy point of view.
- MBC1 algorithm reaches a lower energy than the MBD algorithm, but in longer time.
- MBC2 is better than MBC1, it reaches the minimum faster.
- MBC3 algorithm has the lowest energy.

Above the 0.7 second, the MBC3 algorithm beats all the other optimized algorithms.

From figure 6.5(c), we conclude that:

- The three algorithms find the correct number of objects in around 100 seconds.
- The MBD algorithm starts with a much higher value for the same reason previously mentioned, with simulated annealing, the algorithm at the beginning accepts both good and bad objects.
- The detection rate variance during all the iterations of MBC3 is lower than MBC2 which is lower than MBC2 which is lower variance than MBD.
- The MBC3 has the higher accuracy for the detection rate, its estimated number of objects is the closest to the correct value during all the iterations. The highest value of detected object is 312 objects in 11 seconds. In 2 seconds it detects 314, an error of 4% (positive false) and in 30 seconds it detects 301 objects which is an error of 0.3%

6.2.2 Sample of 1000 objects

We use the following set of parameters: for the MBD algorithm, temperature $T = \frac{1}{\beta} = 1/50$, $\Delta\delta = 0.9997$, $\Delta T = 0.999$ and for 20000 iterations. For the MBC1 algorithm: the number of objects proposed per iteration $R = 200$, for 250,000 iterations. For the MBC2 algorithm: $R = 200$, while for the first few iterations, we set $R = 3 \times R$, number of candidates per location (packet size) equals 8, and number of iterations is 70000. For MBC3, we use the same parameters as for MBC2, and also for this sample size we only use the new birth kernel, we do not use the local perturbations kernels.

For this sample, figure 6.6(a,b) shows how the energy of configurations during iterations evolve with time for the four algorithms. The true value of the energy is $E(\omega) = -801.672$. From the energy curves shown in figure 6.6(a,b), we can conclude that:

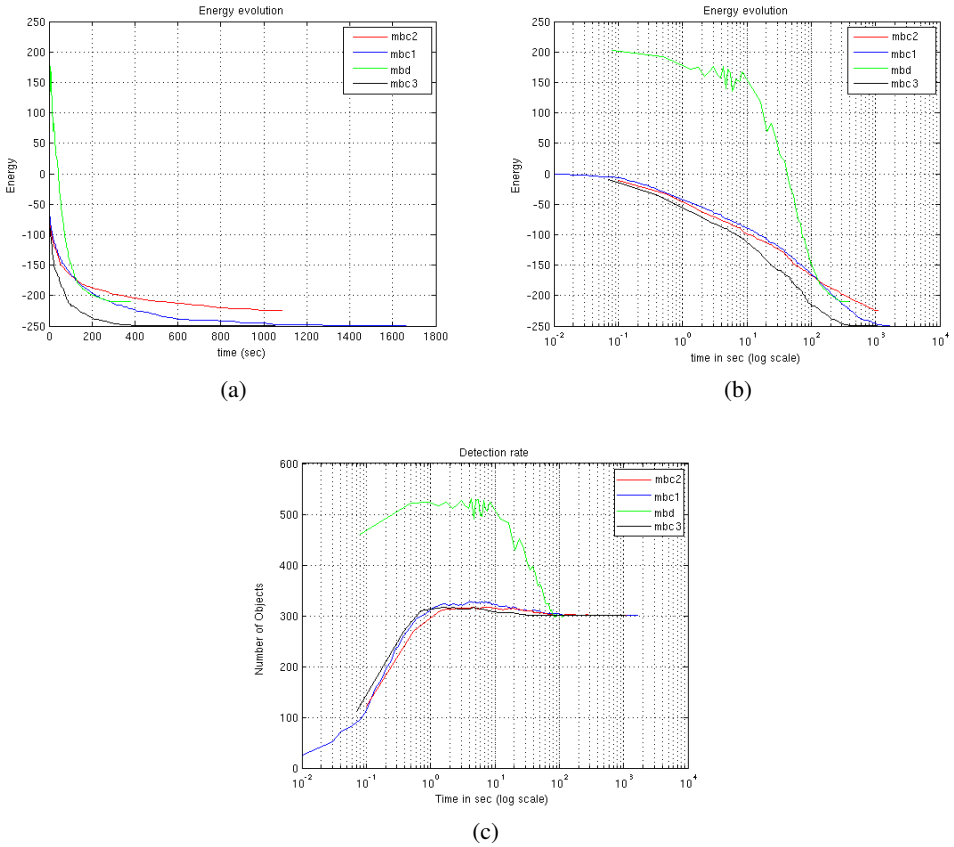


Figure 6.5: (a) Shows the energy evolution versus time on the 300 objects configuration. (b) Shows the energy evolution versus time in log scale. (c) Shows the object detection rate for the same samples.

- From 4 to 40 seconds, both MBC1 and MBC2 algorithm have almost the same energy.
- Above 40 seconds, MBC1 has the lowest energy value.
- Setting the correct MBD parameters becomes hard, we can see that the minimal energy it reachs is far from the true minimum.
- MBC1 algorithm finds the global (correct) minimum faster than MBD and MBC2.
- MBC3 algorithm beats all the other optimized algorithms from the very beginning.

From the detection rate curves shown in figure 6.6(c), we conclude that:

- The MBD, MBC1 and MBC2 find the correct number of objects in around 250 seconds, while MBC3 only requires 200 seconds.

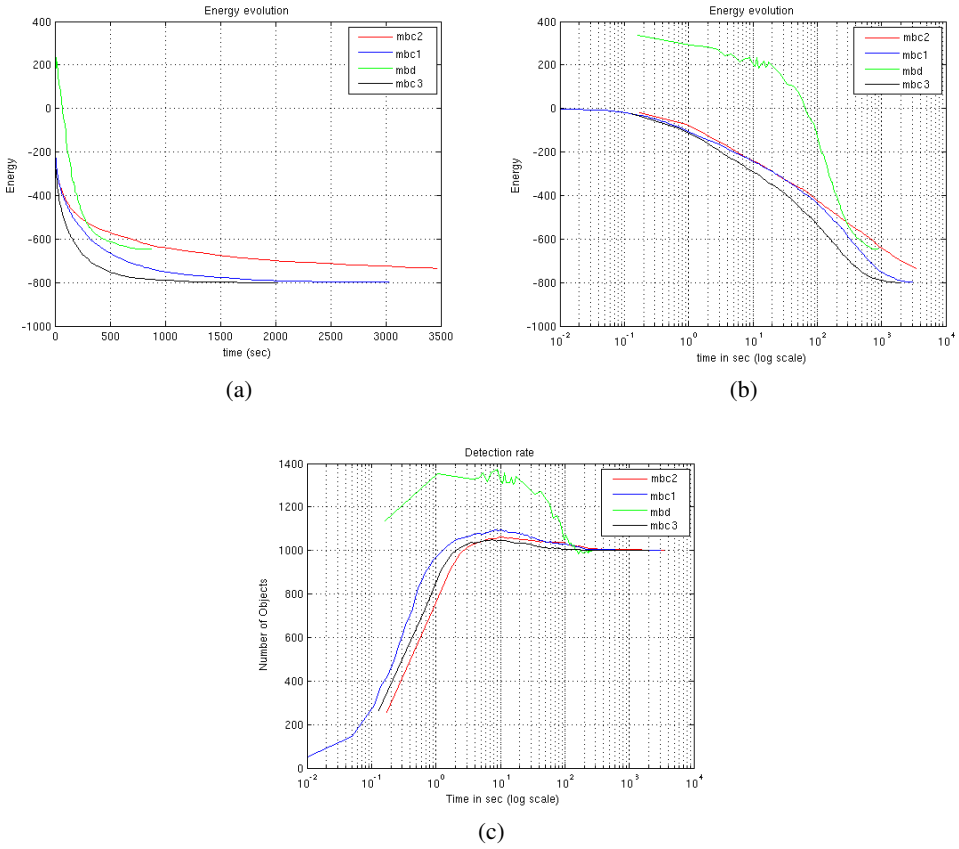


Figure 6.6: (a) Shows the energy evolution versus time on the 1,000 objects configuration. (b) Shows the energy evolution versus time in log scale. (c) Shows the object detection rate for the same samples.

- The MBD algorithm starts with much higher value for the reason previously mentioned.
- From around 10 seconds, MBC1 and MBC2 algorithms have very close detection rate.
- For the detection rate, the MBC3 algorithm has the lowest variance.
- MBC1 has a higher detection rate in the first phase.
- For the MBC2 detection rate, the highest estimated number of objects is 1057 (error of 5.7%) in 9 seconds. The error becomes 1% in 251 seconds.
- For the MBC3 detection rate, the highest estimated number of objects is 1045 (error of 4.3%) in 6 seconds. The error becomes less than 1% in 90 seconds.

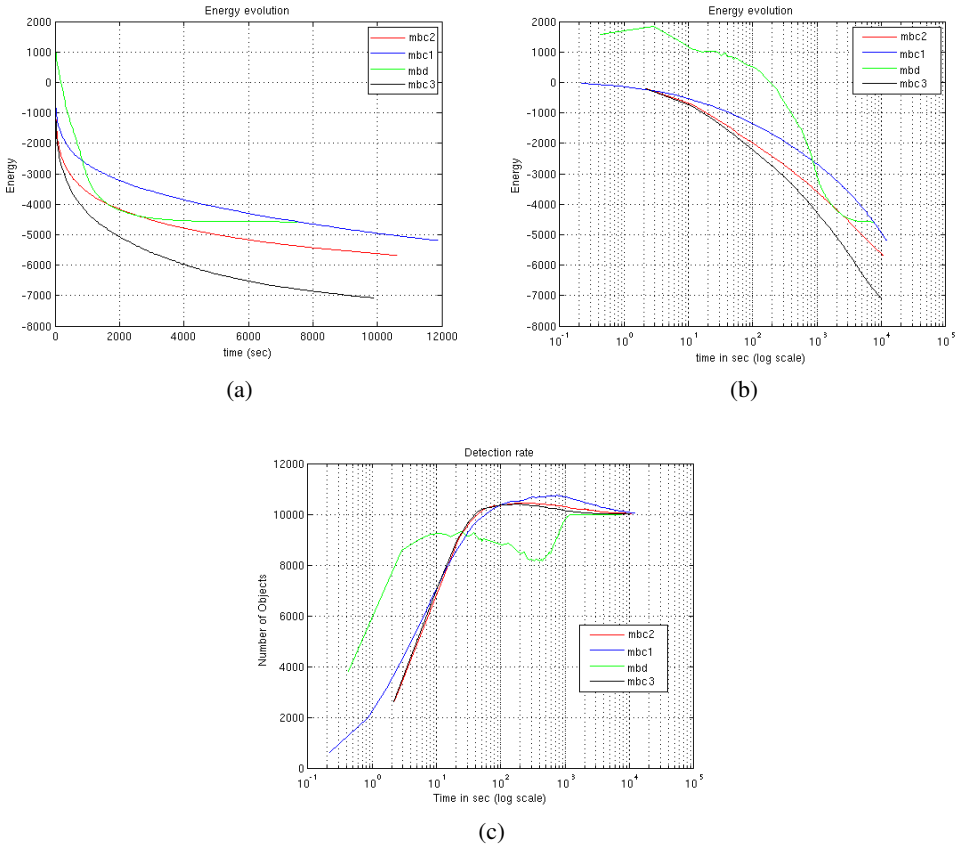


Figure 6.7: (a) Shows the energy evolution versus time on the 10,000 objects configuration. (b) Shows the energy evolution versus time in log scale. (c) Shows the object detection rate for the same samples.

6.2.3 Sample of 10000 objects

We use the following set of parameters: for the MBD algorithm, temperature $T = \frac{1}{\beta} = 1/50$, $\Delta\delta = 0.9997$, $\Delta T = 0.999$ and for 25000 iterations. For the MBC algorithm: the number of objects we propose per iteration $R = 2000$, for 18000 iterations. For the optimized MBC algorithm: $R = 2000$, while for the first few iterations, we used $R = 3 \times R$ (for faster convergence), number of candidate per location (packet size) equals 8, and number of iterations is 10000. For MBC3, we use the same parameters as for MBC2, and also for this sample size we use the new birth kernel and the local perturbations kernels.

For this sample, figure 6.7(a,b) shows how the energy of configurations evolves during iterations with time for the four algorithms. The true value of the energy is $E(\omega) = -7740.95$. From the energy curves shown in figure 6.7(a,b), we can conclude that

- Above 2000 seconds, MBC2 has the lower energy curve than MBD and MBC1.
- MBC3 has the lowest energy curve only after 3 seconds from the start.
- Again, setting the correct MBD parameters becomes hard, we can see that the minimal energy it reaches is far from the true minimum

From the detection rate curve shown in figure 6.7(c), we conclude that:

- The four algorithm find the correct number of objects in around 10000 second.
- The MBD algorithm starts with much higher value for the reason mentioned before.
- MBC3 detection rate has the lowest variance than the other algorithms.
- The MBC2 highest detected value is 10400 objects in 110 seconds, with a positive error of 4%, and then starts decreasing again to the correct value.

6.3 Results on real data

In this section we present results of flamingo detection from aerial images comparing MBC1, MBC2 and MBD algorithms. First we present results on four different colonies. In table 6.1, the testing data is composed of two to three samples from each of the four colonies. We show the percentage of correct detection of flamingos, negative false and positive false. These results are compared to ecologists'[1] counting. Results in table 6.1 show that the newly proposed algorithms outperform the MBD algorithm for the detection. For the detection rate, MBC1 outperforms MBD, and MBC2 outperforms both of them. Both MBC algorithms have lower negative and positive rates for the majority of the samples.

6.4 Conclusions

We have presented an efficient optimization algorithm to minimize a highly non-convex energy function which was previously solved within a simulated annealing scheme. We avoid the difficult task of setting the temperature and cooling parameters of the simulated annealing. We showed the quality of the detection on many test samples of different data-sets. The MBC1 algorithm reaches a lower energy level than the MBD but is slower. We can reach the same level using MBD, but it requires many trials to set the perfect parameter values.

The second version of this algorithm uses belief propagation to optimize the proposed configuration inside each iteration to obtain a relevant proposed configuration. The results show that the MBC2 algorithm is substantially faster than the basic MBC1 algorithm.

[1] Ecologists from La Tour du Valat.

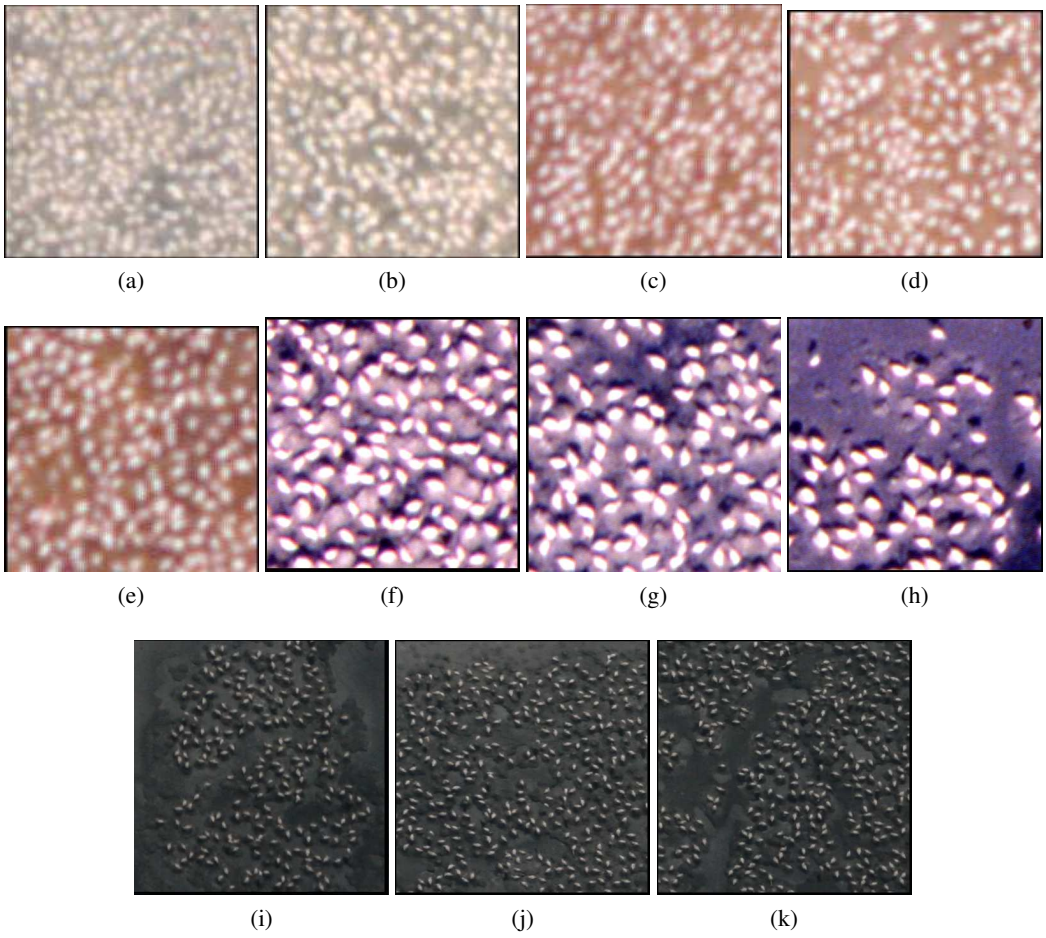


Figure 6.8: (a-k) Samples from 4 different colonies ©Tour du Valat.

We also present the results on the synthetic data using the version of the algorithm (MBC3). We validated the new multiple-birth-death kernel and the integration of the local perturbations kernels. The results demonstrated the efficiency of the MBC3 algorithm on all problem scales, either on the speed of convergence or in the detection rate.

We validated the optimization algorithm in a real application, the flamingo counting problem. Flamingo colonies consist in general of more than one thousand objects, which makes our algorithm much more interesting for a real application. We have demonstrated how our algorithm in the MPP framework can be used to efficiently solve the flamingo counting as one of many possible applications.

Table 6.1: Comparison between MBC1, MBC2 and MBD

Image	**Qualifiers**	**MBC**	**MBC 2**	**MBD**
Fang02 sample 1	Good detection	93	90	87
	Neg. false	0.07	0.08	0.13
	Pos. false	0.16	0.12	0.09
Fang02 sample 2	Good detection	98	98	96
	Neg. false	0.02	0.02	0.04
	Pos. false	0	0.11	2
Fang05 sample 1	Good detection	86	85	82
	Neg. false	0.14	0.15	0.18
	Pos. false	0.1	0.2	0.07
Fang05 sample 2	Good detection	97	97	90
	Neg. false	0.03	0.03	0.1
	Pos. false	0.08	0.07	0.14
Fang05 sample 3	Good detection	94	95	90
	Neg. false	0.1	0.40	0.1
	Pos. false	0.06	0.13	0.14
Tuz04 sample 1	Good detection	100	100	99
	Neg. false	0.0	0.0	0.01
	Pos. false	0.04	0.01	0.01
Tuz04 sample 2	Good detection	98	98	98
	Neg. false	0	0	0
	Pos. false	0.04	0.04	0.04
Tuz04 sample 3	Good detection	100	100	100
	Neg. false	0	0	0
	Pos. false	0.02	0	0
Tuz06 sample 1	Good detection	100	100	100
	Neg. false	0	0	0
	Pos. false	0.01	0	0
Tuz06 sample 2	Good detection	98	100	95
	Neg. false	0	0	0.04
	Pos. false	0.09	0	0.06
Tuz06 sample 3	Good detection	99	99	95
	Neg. false	0.01	0.01	0.04
	Pos. false	0.12	0.12	0.08

Chapter 7

Penguin counting

7.1 Object recognition

Visual recognition of object categories and instances are very challenging problems, finding applications in computer vision, machine learning, and robotics. In the past decade, a huge variety of features and algorithms have been proposed and applied to this problem, resulting in significant progress on object recognition capabilities from monocular images. There is a steady improvement on standard benchmarks such as Caltech-10, Caltech-256 and other. There exist other challenges such as PASCAL Object challenge raised by the computer vision community to keep track of advances in this field.

Multiple object detection from a still image is also a very challenging problem, finding applications in different domains such as counting crowd [70], evaluating a population of trees [33], animals [25], building [55], cars [81] or cells [32]. Our work focus is on large and dense scenes, with main application being the evaluation of a population of penguins (seabirds). Penguins are very sensitive to climate changes, which makes an automated way of monitoring their number a real need for the ecologists.

7.1.1 Imaging and sensors

Solving a multiple object detection problem is a function of the available and feasible data. For some problems, the available data is from a top view perspective of the scene. A typical example is the building detection problem, where data sources could be either satellite or aerial acquisitions. The mostly used imaging sensors for this application (most image processing applications) will be optical, lidar and radar.

From those sources, we can obtain 2D, 2D+ or 3D data that will be used for the detection purpose. In other problems, *e.g.* people counting in a crowd [91], most of the data are from photos taken by hand held cameras. Imaging can be accomplished using standard

monocular camera that holds only color information. We may also have a computed depth map either from stereo camera or using laser device. Infra red cameras can also be used in some situations. The availability or the cost of the image acquisition method, may be the reason for choosing one solution or another for a certain problem.

Descriptors can be in general categorized into two classes, global descriptors and local descriptors. These features had to satisfy certain properties:

- Robustness with respect to affine transformation: scaling, rotation, ...
- Complexity (extraction and comparison): features with high complexity will be of limited usage

More properties for local features:

- Repeatability: The same feature can be found in several images despite geometric and photometric transformations
- Distinctiveness: Each feature has a distinctive description
- Compactness and efficiency: Fewer features than image pixels
- Locality: A feature occupies a relatively small area of the image; robust to clutter and occlusion

7.1.2 Depth Map

Object category detection, from images or depth map has encountered a lot of developments. One of the fields that was boosted due to this advance is the field of image indexing. Depth map can be calculated either using stereo camera (or more cameras), or laser scanners which are becoming more accessible (specially low resolution), or projecting an invisible infra-red structured light pattern and performing stereo triangulation. Once depth information is obtained, we can get accurate geometrical properties of objects, thus we can greatly improve the detection accuracy. Geometrical properties can be used on their own for object detection, but combining this information with color information, will result in a higher detection rate.

To speed-up Viola face detector [89], the depth information was simply used to reduce the image region and possible scales for candidate faces [19]. Basically, depth information will simplify and accelerate the task of detection and recognition by giving shape or geometrical clues about objects. In [14], the authors proposed an interesting method for the extraction of (parametric) primitives (sphere, cylinder, plan,...) from the depth map using the normal vectors as features to extract those primitive shapes, FPFH [80] and using conditional random field (CRF).

Depth map quality: High quality laser scanners give more accurate depth map than what we get from stereo reconstruction. There is also a trade-off when selecting the laser scanners, mostly, high quality scanners are very expensive and slow, and reciprocally, low quality scanners are fast and cheep.

7.1.3 Descriptors and matching for 3D objects

For a recognition task, first we calculate the descriptors of the newly proposed object and then compare it to existing learned object. After the acquisition of the 3D object, what ever the representation [1], we have to assign a representative descriptors. Here we refer to some of the existing approaches for 3D representation and matching (this is not an exhaustive list):

- Shape statistics
- Structural
- Transformation based
- View based

Statistical approach is based on the distribution of some measurements. Histogram is the most used representation for those distributions, it find uses in many computer vision applications. Histograms hold either local or global information. Distributions can model features such as surface curvatures, point features (PFH) [17], or any other. Each object is described by a distribution of such features. The comparison of two 3D objects becomes straightforward, is it a simple comparison of histograms. Some techniques use the correlogram instead of the histogram, since it embeds spatial information, which is lacking in histograms. Statistical approaches can be invariant to translation, rotation and scaling. Global information is more robust to noise, but less discriminative, while local information, is more discriminative, but more sensitive to noise.

The structural approach aims to represent a structural model of 3D objects. This approach describes a higher level of information, e.g. a structural representation of a human body to represent (head, abdomen, arms and legs). This representation can be used for pose and activity estimate for humans, since it hold detailed information about the body parts and the relation between them.

Transformation based approach is based on the transformation of the 3D object from the Euclidean space to another space. The most popular transformation based techniques is known as spin images [54]. The aim of spin images is to capture both local and global

[1] 3D objects can be represented in different forms: Cloud of points, Triangular mesh, Set of parametric surface and Set of voxels.

features, and being view-independent (somehow close to the view based approach). Spin images encode the density of the mesh vertices projected onto an image (of the visible and cluttered parts), for each point of view for each instance.

View based approach is basically projecting 3D data into an image (2D), and do the recognition task based on feature extracted from this 2D image. People usually uses this technique because working with 2D data is simpler than 3D, and because people have more experience and existing methods can easily be adapted to this problem. The number of views can be fixed or dynamic.

7.1.4 Features given 2D data

Features can be categorized into two categories, global and local features.

7.1.5 Global features

Features development from 2D data started earlier in the computer vision community that from 3D. One of the fields that was boosted due to the advances in feature development, especially global features is image indexing. Method for particular instances have been proposed, such as: the Statistical Models of Appearance, developed by Tim Cootes. A detailed report of this work can be found in [21]. We tested some methods of statistical appearance models [21] on photos of individual penguins, but without promising results, and the reasons are:

- In most of the photos, with the different camera position with respect to a penguin, and different illumination conditions, it was not possible to extract a set of *reliable edges*, but only for some positions, and only for penguins close to the camera, where the image holds its high frequency content.
- When we move to process a photo with many penguins, with the occlusion, most of the edges becomes hidden by other penguins.

There also exists a very popular method for face detection [89], but this method requires a specific way of imaging (facing the camera, ...), and it also requires tons of photos for training the model, which makes it not practical for our application.

Here we list some of the simple and well known global descriptors:

- 2D Correlation
- Histogram
- Correlogram

The simplest idea for object detection is using template (patch) with an image, it is known as template matching. It consists of sliding the "patch" against the input image, and measure the similarity between the patch and the image at every position. For similarity measure, two of the most popular matching methods are:

1. Square difference method: This method matches the squared difference, so a perfect match M will be 0 and bad matches will be large. Let I be the input image, T the template, the matching at location (displacement) (x, y) is given by:

$$M_{sqd}(x, y) = \sum_{x',y'} [T(x', y') - I(x + x', y + y')]^2$$

2. Correlation matching methods: This method multiplicatively matches the template against the image, so a perfect match will result in a large value and bad matches will have a small value. The matching M at location (x, y) is given by:

$$M_{corr}(x, y) = \sum_{x',y'} [T(x', y')\dot{I}(x + x', y + y')]^2$$

Both methods have a normalized version which is more useful since in reducing the effect of intensity difference between the input image and the template [78]. The normalization coefficient is given by:

$$Z(x, y) = \sqrt{\sum_{x',y'} T(x', y')^2 . \sum_{x',y'} I(x + x', y + y')^2}$$

Most of global descriptors are based on color information. Two of the most popular in image indexing and object detection features are: histogram [85], [35] and correlogram [49] which model the spatial color distribution and can be considered and as extension of the gray level co-occurrence matrix in color space.

Let $\Lambda_i \in [0, 255]^3$, the histogram value for each color $c \in \Lambda$ for object (or image) $\mathbf{o}_i$ as follows:

$$h_c(\mathbf{o}_i) = \#pixel_c$$

Quantization steps Q is an important parameter to be considered. Once the histograms are calculated, the similarity between two images (image and object) can be measured using the distance between their corresponding histograms. Let I_i and I_j be two images, $h(I)$ be the histogram of image I and let $d(.,.)$ be a distance between two histograms:

$$d_{hist}(\mathbf{o}_i, \mathbf{o}_i^{'}) = \sum_{m \in Q} (|h_m(\mathbf{o}_i) - h_m(\mathbf{o}_i^{'})|)^n \ ,$$

where n determines the norm, as norm L_1 and L_2. The computation of the correlogram to measure the similarity between to images is given by:

$$\alpha^{(k)}_{c_i,c_j} \triangleq |Pr(C(p_1) = c_i, C(p_2) = c_j | dist(p_1, p_2) = k)| \,,$$

where $\alpha^{(k)}_{c_i,c_j}$ is the joint probability $Pr(C(p_1) = c_i, C(p_2) = c_j)$ of having the color of pixel 1 ($C(p_1)$) equal to c_i and the color of pixel 2 ($C(p_2)$) equal c_j at a distance k (distance in norm L_∞). There also exist what is known as the auto-correlogram, which is a reduced version of the correlogram, it makes only the calculations of the joint probability for $c_i = c_j$. Also to measure the similarity between I_i and I_j, we calculate the correlogram or auto-correlogram vector. To measure the similarity between the two images, the distance between their corresponding correlogram or auto-correlogram vectors is calculated as follows:

$$d^k_{corr}(\mathbf{o}_i, \mathbf{o}_i^{'}) = \sum_{m \in M} (|\alpha^k_m(\mathbf{o}_i) - \alpha^k_m(\mathbf{o}_i^{'})|)^n$$

Histogram complexity is $O(n)$, correlogram complexity is $O(n^2k)$ for the dynamic programming version and $O(n^2k^2)$ for naive version. We tested both histogram and correlogram methods, and results will be presented later in this chapter.

7.1.6 Local features

The state-of-the-art object detection algorithms usually consist of two parts: detector and descriptor. First, points of interest are detected in the images with a region around each point of interest, then, an invariant descriptor (feature) is associated with each region. Similarity measure may thus be established by matching the descriptors.

In recent years local image detectors have boomed. Let us list some of the famous local features in invariance order. While all the following are translation invariant, Harris point detector (it combines corner and edge detector) is rotation invariant, it was first introduced in [48]. Features that are rotation and scale invariant are: Harris-Laplace [68], Hessian-Laplace [69], and difference-of-Gaussian (DoG) region detectors [62]. Features that are invariant to affine transforms are: MSER ("maximally stable extremal region") [52] and LLD ("level line descriptor") [34]. For more invariance, comes the popular Scale-invariant feature transform (SIFT) [67]. It is one of the most popular features used in computer vision for object detection, image stitching, video tracking and robotic vision.

SIFT features are invariant to scaling, rotation, translation. It is also partially invariant to illumination change and affine or 3D projection. Gaussian kernels were used in order to achieve scale invariance the scale space [64]. To achieve rotation invariance, key locations where localized at maxima and minima of a DoG in the scale space. An image pyramid with re-sample where used for efficiently calculating these key locations. At each level of the constructed pyramid, an image gradient is calculated, and also orientation. The gradient

is thresholded to achieve robustness to illumination change. The key locations gives canonical orientations to be rotation invariant. For more robustness to illumination and contrast change, the orientation will be the peak in the histogram of local image gradient orientations. Speeded-Up Robust Features (SURF) [7] is considered as a speed up version of the SIFT feature. The usage of *integral image* was the key element for drastically computation time reduction. The selection of which feature to use is problem dependent, not every application requires invariance to every type of deformation.

The complexity of the recognition task comes from the enormous change in visual appearance of objects due to different view point, illumination, occlusions, class variability and others.

7.2 Penguin counting problem

A popular approach in computer vision is to calculate a 3D reconstruction from video data [76] or from many images. An interesting work used two million images uploaded by tourists to Flikr for Rome reconstruction [1]. Also occlusion problems become easier to solve in video data due to object's relative motion [88], the same for multiple cameras [57]. Unfortunately, these approaches are also not feasible in practice for the penguin counting problem.

Photographing penguin colonies is a complex task, where there exists many challenges to deal with. As we can see from figure 7.11, the penguin colony is spread over a very large space. Another complexity comes from objects motion, penguins can move during the photographing. Another reason is, due to the extreme weather conditions in the south pole, taking thousands and thousands of images or a video of the whole colony is not a feasible task. Even if it is possible once, it can not be repeated, and the aim is to develop an easy automated way to monitor the evolution of the number of animal in the colony. This implies that the photographing and counting process should be as simple as possible since it will be done many time per year.

Photographing penguin colonies is only possible using a hand held camera from hills surrounding the colony. Aerial images are not possible (air planes are not allowed), small airplane can not be used most of the year because of the wind speed, and current satellite image resolution nowadays is not enough. In this situation, counting penguins from images requires to handle *occlusions* and *perspective effects*.

7.2.1 Adelie penguins

By end of 2008, we received the first image collection taken by ecologists. This image collection contains images of two types of penguins, Adelie and Emperor penguins. We

propose different solutions for the penguin counting problem based on the selected penguins colony. For Adelie penguins, this case is actually the simplest case. Figure 7.1 shows a panoramic photo of an Adelie penguin colony. This problem is quite actually quite close to a top view scenario, which makes it simple to solve. The simplicity of this specific example comes from:

- The terrain that the penguins are going through is inclined, which makes the camera position close to a top view scenario with respect to the scene. As a result, there are no *occlusion*. The only difference from a top view case is a small perspective effect, on the right of the image (which is actually the top of this rotated image) objects are getting smaller.
- During the climbing, most of the penguin's back are facing the camera, which is black (not too many light reflections in this image), which simplifies the design of the data term function.

Figure 7.1: A panoramic photo of an Adelie penguin colony, taken in 2008 ©DEPE / CNRS.

7.2.2 Proposed solution

We propose using the 2D MPP model introduced in chapter 6, section 6.1, which was developed for the flamingo counting problem [25, 26]. A minor modification should be applied to the input image to be able to use the flamingo model. The contrast of the image should be inverted, so the black color of penguins back becomes bright to match the existing data term. Figure 7.2(a) presents a sample from the large panoramic photo. The detection result of counting on this sample using the Opt-MBC [36] algorithm (with belief propagation) is shown in figure 7.2(b), where detected penguins are highlighted by pink ellipses. We can conclude that the 2D MPP model [25] is quite adapted for this specific scene configuration; only a small adaptation of the data term is required.

(a) (b)

Figure 7.2: A sample of an Adelie penguin colony. The detection result on this sample ©DEPE / CNRS.

7.2.3 Emperor penguins

In figure 7.3(a), we present a panoramic photo of a group of Emperor penguins and in figure 7.3(b) we zoom on a small part of this group. In figure 7.3(c), we present a photo of a single Emperor penguin.

After analyzing the first image collection we received by end of 2008, we found the image quality to be visually good. These images induce many problems:

1. The camera parameters are unknown for these set of images
2. Images are stored using a lossy compression technique
3. The panoramic image is fine for visualising the whole group, but contains errors, as the shadows have a varying orientation by traversing the image from side to side. And usually, shadows are potentially part of data term functions.

7.2.4 Proposed solution

We proposed solving the problem using the 3D point process introduced in chapter 5.

Configuration Space

Let us consider a space $K \times \mathbb{R}^+$, where K is a closed connected subset of $\mathbb{R}^2$, referred to as the real plane and $\mathbb{R}^+$ refers to the z-axis (objects height). We consider a perspective transformation, supposed as known, $\mathcal{P}$, such that $\mathcal{I} = \mathcal{P}(K \times \mathbb{R}^+)$. The data consists of $I = (i_s)_{s \in S}$, where S is a discretization of the image plane $\mathcal{I}$ and $i_s \in \Lambda$ is the color of pixel

(a)

(b)

(c)

Figure 7.3: (a) A panoramic photo is a part of an Emperor penguins colony ©DEPE / CNRS. (b) A zoom on a sample of this colony photo ©DEPE / CNRS. (c) A photo of an individual Emperor adult penguin ©DEPE / CNRS.

s. We consider configurations of an unknown number of objects in the real plane. For our application, consisting of counting a population of penguins, we consider two classes of objects $l \in \{0, 1\}$ corresponding respectively to adults and chicks (babies). 3D color models of each class are designed using standard software [2], and are shown in figure 7.4[3]. The 3D model coloring was applied using *texture mapping*. We consider two positions for the penguins $p \in \{0, 1\}$ corresponding to standing up and lying (on the ground) animals. Finally, a penguin is also characterized by its orientation ϕ with respect to the z-axis (in cylindrical coordinates). Therefore, an object is defined as $\omega_i = (x_i, m_i) \in K \times \mathbb{M}$, where x_i represents the object location and m_i its mark, $\mathbb{M} = \{0, 1\} \times \{0, 1\} \times [0, 2\pi[$.

We consider a marked point process with points lying in K and marks in $\mathbb{M}$. The configuration space is then defined as:

$$\Omega = \bigcup_{n=0}^{\infty} \Omega_n, \quad \Omega_n = \{\{\omega_1, ..., \omega_n\} \subset K \times \mathbb{M}\} , \tag{7.1}$$

[2]Software such as Blender and 3D Max

[3]Rendering using DAZ Studio software, used only to render this photo in maximum quality, but inside our program we only use OpenGL.

where Ω_n is the subset of configurations containing exactly n objects. This configuration space is the space of all sizes, positions and parameters, same idea of the space in [90]. The process is defined on Ω as follows:

$$d\pi(\omega) = h(\omega)dv(x)\prod_n (d\mu(m_i)) \,, \tag{7.2}$$

where $\omega = \{(x_i, m_i), i = 1, ..., n\}$, $x = \{x_i, i = 1, ..., n\}$, $m = \{m_i, i = 1, ..., n\}$, $\nu(.)$ is the measure of the Poisson process of intensity $\lambda(u)$, $u \in K$, $\mu(.)$ is a measure on $\mathbb{M}$ (product measure of the Lebesgue measure on $[0, 2\pi[$ and counting measure on $\{0, 1\} \times \{0, 1\}$) and $h(.)$ is a density. We consider a Gibbs density, written as follows:

$$h(\omega) = \frac{1}{Z} exp\{-U(\omega)\} \,, \tag{7.3}$$

where Z is the partition function (normalizing constant), and $U(.)$ an energy.

7.2.5 Energy

This energy takes into account the interactions between geometric objects (prior energy) and a data energy to fit the configuration to the image (data energy):

$$U(x) = U_d(x) + \gamma_p U_p(x) \tag{7.4}$$

where γ_p is the weight we assign to the prior term. Minimization of this energy corresponds to the correct configuration detection.

Prior Energy

For each 3D object ω_i, we consider its occupancy area in K by an ellipse. For each object ω_i we define an ellipse $C(\omega_i)$ which approximates its occupancy area in K. Both standing and lying objects are well approximated by an ellipse. The overlapping between two objects is approximated by the intersection of the two corresponding ellipses $C(\omega_i)$ and $C(\omega_j)$, which reduces the prior term to the same model as presented [25]. We define the following symmetric relation:

$$\forall \omega_i, \omega_j \in K \times \mathbb{M}, \omega_i \sim \omega_j \Leftrightarrow C(\omega_i) \cap C(\omega_j) \neq \emptyset \,, \tag{7.5}$$

and the overlapping coefficient:

$$\forall \omega_i, \omega_j \in K \times \mathbb{M}, \omega_i \sim \omega_j, O(\omega_i, \omega_j) = \frac{A(C(\omega_i) \cap C(\omega_j))}{\min\left(A(C(\omega_i)), A(C(\omega_j))\right)} \,, \tag{7.6}$$

where $A(.)$ represents the area. To define the prior, we penalize overlapping in the configuration by considering the worst case for each object, that is:

$$U_p(\omega) = \sum_{\omega_i, \omega_j \in \omega} \max_{\omega_j \sim \omega_i} O(\omega_j, \omega_i) \tag{7.7}$$

The reasons for using only the occupancy area, and approximating it with an ellipse are the following:

- Calculating the intersection between the 3D models is a high computational task, and is unnecessary for our model.
- For the prior term, the aim is to penalize objects overlapping, it does not require a very high precision in the calculation of volume or area of intersection, just enough to correctly penalize the overlapping. In the 2D MPP model used for flamingo counting [25], the ellipses' intersection calculation is based on discrete approximation.
- For the prior term, the usage of an ellipse is even more correct than using an ellipsoid or a cylinder, and the reason is, the height of the object should not affect how the overlapping of two objects is penalized, *e.g.* the cost of overlapping between two adults, with $n\%$ occupancy area overlapping, should be the same when the overlapping is between an adult and a baby penguin with the same $n\%$ occupancy area overlapping.

Camera Parameters

The proposed method is based on 3D scene simulation on the GPU using OpenGL. The proposed configurations are projected onto the image plane, and the configurations are modified until convergence. Camera parameters are required for projecting the configuration during the iterations of the optimization algorithm. The projected image $I = (i_s)_{s \in S}$ is given by projecting every point from the 3D space world M_p^{3D} to the image space, that is obtained by:

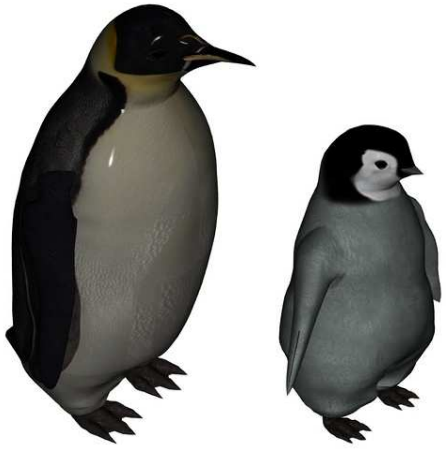
$$i_s = A \times \begin{pmatrix} R & T \\ 0 & 1 \end{pmatrix} \times M_p^{3D} \ . \tag{7.8}$$

Matrix A is the intrinsic camera parameters and $(R\ , T)$ are the extrinsic parameters (rotation and translation). For the semi-synthetic images (section 7.2.6), we already know all the parameters. For images taken before the protocol definition, we have to estimate those parameters. For the current time, we manually set those parameters. We plan to integrate an automated method for camera pose recovery [44]. This method was developed explicitly to recover camera parameters from a random set of points. It will be included as one phase of *detection* and one phase of *re-estimating the camera pose* and iterating between them until convergence.

Data Energy

Computing the data term requires to project the scene on the image plane. We consider a data term associated with the configuration ω given by:

$$U_d(\omega) = \sum_{\omega_i \in \omega} U(\omega_i | c(\omega_i)) \ , \tag{7.9}$$



(a) (b)

Figure 7.4: This figure represents a very fine rendering of our 3D models for adult and chick penguins. The version used has a reduced number of triangles for fast projection

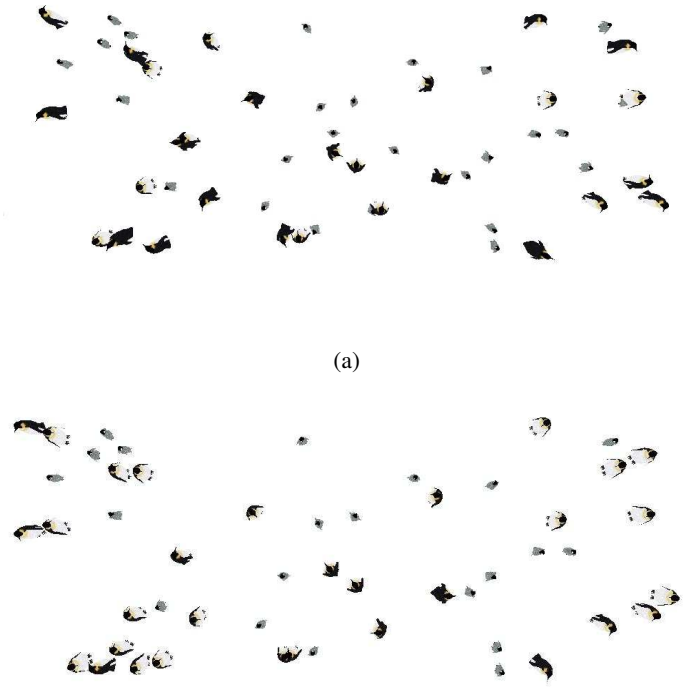
where $c(\omega_i)$ denote children of ω_i, which depend on the camera position with respect to the scene.

Measuring Similarity

Similarity measure for this application is the first major challenge of the counting task. For a given object ω_i, let $\mathbf{o}_i$ be the set of projected pixels and let $\mathbf{y}_i$ be there corresponding set of pixels in the input image. Measuring the similarity is accomplished by measuring the distance $d(.,.)$ between the two objects corresponding to feature vectors $v(\mathbf{o}_i)$ and $v(\mathbf{o}_i)$. We consider at first a global color based approach. We have tested both histogram and correlogram [49] on synthetic images.

Figure 7.5(a) shows a very simple case, a synthetic image representing a top view of a set of penguins, and with simple background (white). In figure 7.5(b), we present the detection result using the proposed 3D MPP model and using correlogram as a feature for similarity measure. The detection results are quite good, where the correlogram gave better results than te histogram. This method gave interesting results for sparse objects and white background, but the detection quality dropped when the background changed from white to icy background as shown in figure 7.7. Even if this method has passed the complexity of using icy background, it was not selective enough for a dense scene under occlusions.

The reason behind the detection quality drop when using a more complex (realistic) background comes from the confusion between the color features of a candidate object and



(a)

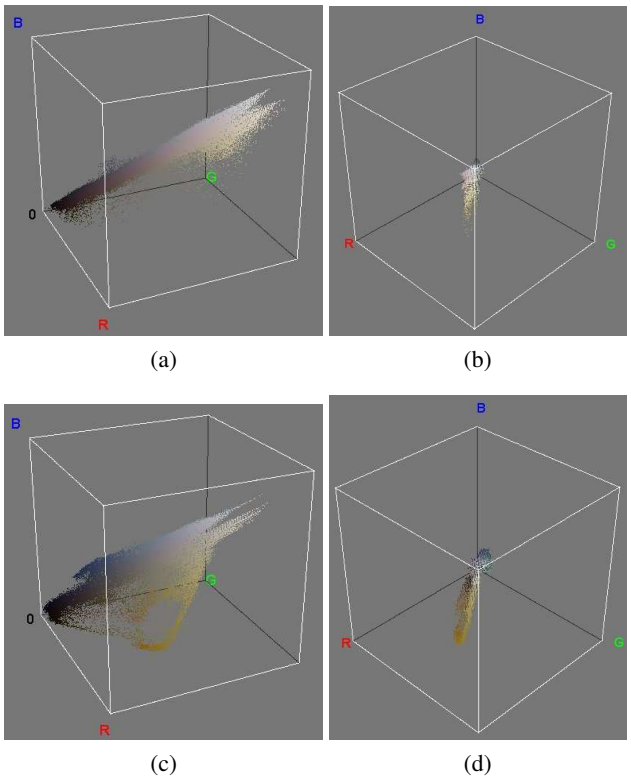
(b)

Figure 7.5: (a) A top view of a simulated penguin configuration composed of adults and babies as input ©Ariana/INRIA. (b) Detection result on this input image ©Ariana/INRIA.

the color feature (in the color space) of any part of the background. This confusion is even higher for baby penguins since their color is very close to the icy background. Let us consider the colony sample from figure 7.3(b), and the single penguin from figure 7.3(c). The RGB color histogram of the colony sample is shown in figure 7.6(a-b), and the color histogram of the single penguin is shown in figure 7.6(c-d). We can comment on the color feature vectors that:

- Colors of a single penguin or the colony sample occupy a very small portion of the space. The variance is very low. Both images occupy almost the same part of the color space.
- Colors of a penguins are not *black* and *white* as it first may look like, it is more or less distributed on the gray axis between $\{0, 0, 0\}$ and $\{255, 255, 255\}$.
- The yellow-orange part on the neck of a Emperor penguin is minor in the color space, it is only prominent on penguins close to the camera, and in many cases it is occluded.

With all these problems, color space can not be easily used for similarity measure function. We also tested some color space reduction as a special quantization for the color space, but it was not of great use.



(a) (b)

(c) (d)

Figure 7.6: (a,b) RGB histogram of a sample of the penguin colony. (c,d) RGB histogram of an individual Emperor penguin.

In a second stage, we aimed at testing the local features method, such as SIFT, and others. We selected SURF for two reasons, it is very close to SIFT method, which is considered as one of the state-of-the-art methods of local feature, and also it has a lower complexity than SIFT.

We created a training database from penguin faces, arms, for adults and babies, and some background parts, where a sample of this training set is shown in figure 7.8. The goal was to extract SURF feature from those images, and train a classifier to be used for the data term function. We extracted the features, and trained a binary decision tree. Unfortunately, this method did not really progress. The main limitation came from the fact that we were able to extract SURF features from objects lying in the first plane of the image, but further objects (far from the camera) due to the perspective effect did not contain enough high frequency components for the features extraction phase. In many samples the number of extracted features where zero or close to zero. The second problem is that usually methods

Figure 7.7: A top view of a simulated penguin configuration composed of adults and babies using a real background (ice) ©Ariana/INRIA.

based on SIFT use a second phase for matching features with some geometric relation between those features, and those geometric information is less obvious for non-rigid objects (penguins).

Proposed solution

We propose interactive k-means segmentation. It allows us to be independent from illumination conditions. We propose an adaptive color space reduction by clustering the data using a k-means algorithm into eight to twelve clusters. The user then labels each cluster among the three classes corresponding to adults, chicks and background to produce an image with only three labels. The advantages of this method are:

- local object detector on the image becomes of very low complexity *O(n)* which is important due to the number of configurations tested during the stochastic dynamics,
- it reduces the parameter space enormously and object detection becomes independent of the orientation ϕ, so that we can ignore this parameter,
- the user interaction is very limited.

Given that each object type has one label, the data term can simply be calculated as follows:

$$U_d(\omega_i) = \sum_{s \in \mathbf{o}_i} \left(\frac{-\lambda \#_{good} + \#_{wrong}}{\lambda \#_{good} + \#_{wrong}} \right)^n \tag{7.10}$$

where $\#_{good}$ is the number of pixels whose class (label) matches, $\#_{wrong}$ is the number of pixels which do not match, λ is a parameter to control the number of correct pixels requires to be a good object (negative energy) which can also be different for adults and chicks. We

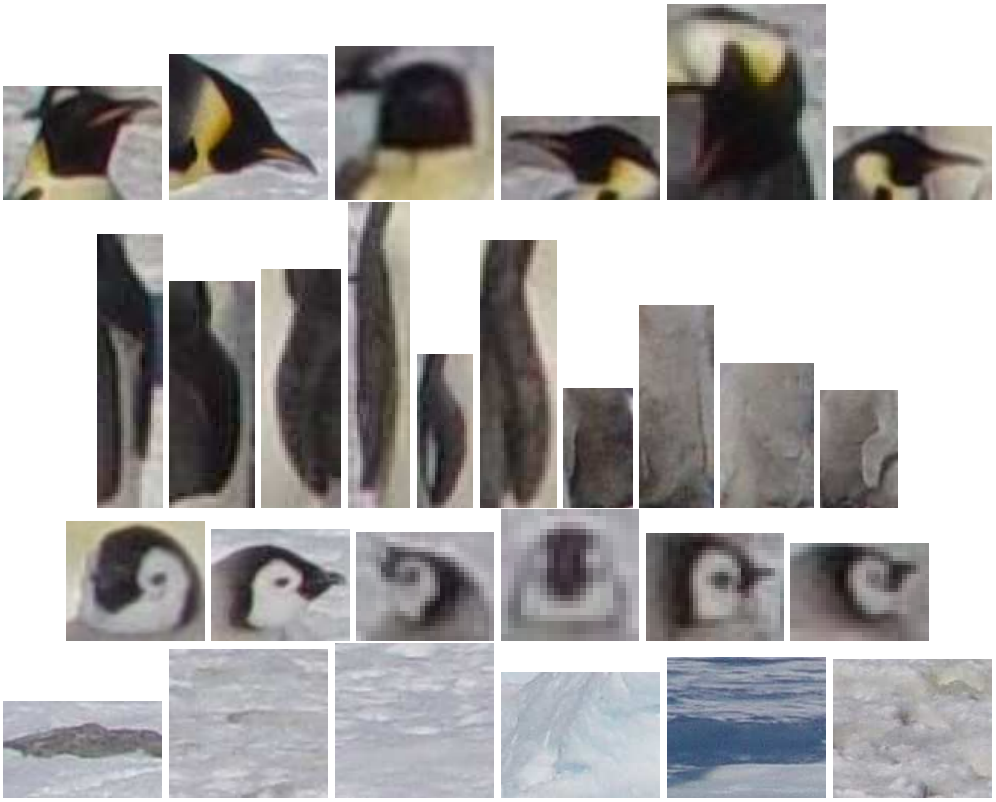


Figure 7.8: A sample of the training set containing heads and arms of adults and babies as well as samples from the background ice ©DEPE / CNRS.

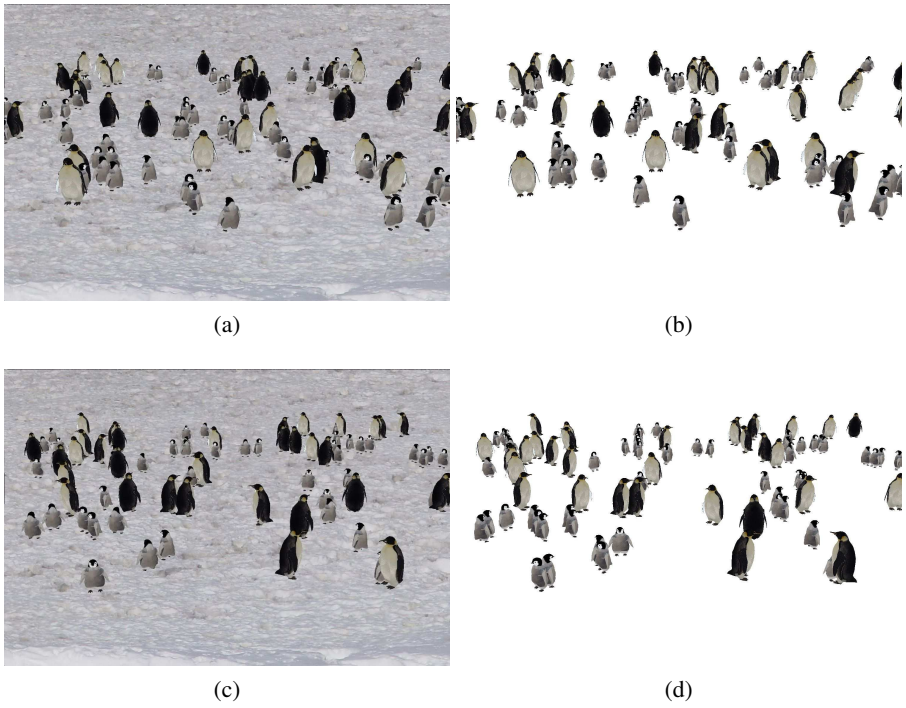
model the effect of partial occlusion (reduction of number of pixels) of $\mathbf{o}_i$, by giving a lower weight to occluded objects based on the ratio of its number of visible pixels to its total number of pixels as follows:

$$U_d(\omega_i) = \frac{\#(visible\ pixels)}{\#pixels\ of\ full\ object} \times U_d(\omega_i)\ . \tag{7.11}$$

While this data term is far from being the most adapted, it allows us to validate some parts of the proposed 3D Markov point process model.

7.2.6 Results

In this section, we present results on two synthetic images and primary results on a real image. Configuration shown in figure 7.10.(a) consists of 33 adults penguins and 44 chicks, and in figure 7.10.(b) we present the results, with 35 detected adults and 50 chicks. Configuration shown in figure 7.10.(c) consists of 32 adult penguins and 35 chicks, and in figure 7.10.(b) we present the results, with 31 detected adult and 57 chick. By looking at these



(a) (b)

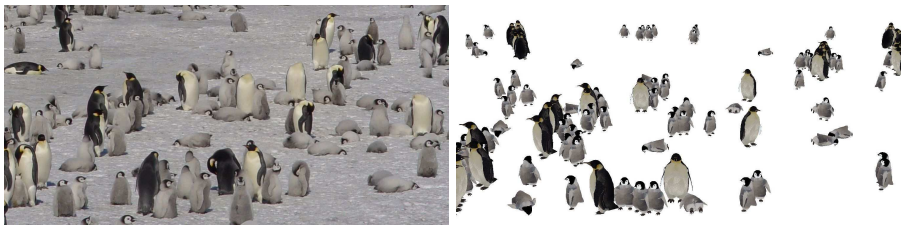
(c) (d)

Figure 7.9: (a,c) Shows inputs for synthetic images, (b,d) are the detection results ©Ariana/INRIA.

results, we can see that we can detect most of the partially occluded objects. This detection is correct when the occlusion is between an adult and a chick, and an over-detection inside the same type. This limitation comes from the currently naive data term used, with just one label for a dense homogeneous group, with no edge (borders) information, it becomes very hard to make exact detection. In the real image, we have a small part of a colony image. For this image we approximate the camera parameters manually. Our model was able to detect partially the configuration, still suffering from the imperfection of the manual approximation of the camera parameters. We also get a *full 3D reconstruction* of the scene as a byproduct of the proposed algorithm.

7.2.7 King penguins

From the photo collection we received by end of 2008, we realized the need to define exactly the imaging process. The first image collection was used mostly for testing different potential data terms. It also served as a reference for the design of the 3D penguin model (baby and adult), both color and geometry. In addition to the proposed model for Emperor penguins, special consideration for the terrain is required.



(a) (b)

Figure 7.10: (a) Shows input for real image ©DEPE / CNRS. (B) Is the detection results ©Ariana/INRIA.

Figure 7.11: Photos of the terrain and the penguin colony taken from the top of a hill surrounding the colony ©DEPE / CNRS.

7.2.8 Proposed solution

For the first image collection, there was only some simple recommendation about imaging process, but due to the scene complexity, such as large space, complex terrain topography, complexity to recover camera parameters, . . . , we collaborated with the on site ecologist to define an *Imaging Protocol*.

Since for the recognition task we simulate the 3D scene with a perspective effect, the topology of the territory is very important. The penguins are living in a region that is not fully flat, and in our model the elevation information is important. North and South poles are not usually covered by space missions for obtaining elevation maps. The Digital Elevation Model (DEM) is not available for the Possession island. The only available data for this island at the Institut Géographique National (IGN) is a map. For Adelie land, which is at the South pole, we where able to get a DEM thanks to a program called SPIRIT. In 2007, CNES actively participated in the 4th International Polar Year (IPY), making available for free their data for the public. For details about this project, please refer to their web site: http://www.spotimage.com/web/en/3163-spirit-dem.php

The protocol goal was to specify the following:

1. Define explicitly a process to be able to recover the camera parameters (inserted reference points with known GPS coordinates, distances and angles)
2. Define an imaging procedure for the terrain for reconstruction
3. Define an imaging procedure for photographing the penguins
4. Define all required hardware for the protocol, starting for proposing camera and laser measurer type, and then defining the number of reference points in the terrain that we mark by sticks (please refere to the protocol), to even the camera settings

We defined in an iterative way with the ecologist an *Imaging Protocol*. This protocol is attached in *Appendix A*. This protocol concerns imaging two elements, the king penguin colony and the territory occupied by the colony of interest. We detailed the description of the required images for both elements.

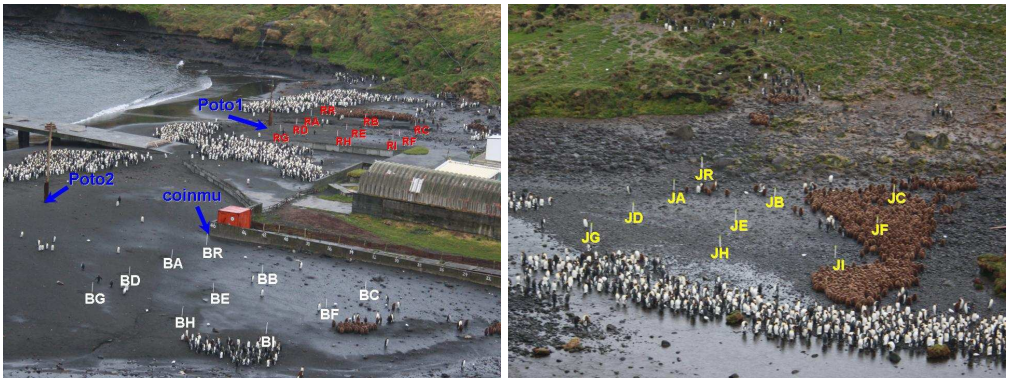
Given the huge space occupied by the king penguin colony, as shown in figure 7.11, the first concern was splitting this large area into smaller areas which can be easily photographed. Figure 7.12 show a proposed division for the terrain by the ecologists. Each proposed region defines a flat region on which penguins live, where each region can somehow be treated independently.

Figure 7.12: A proposed division for the terrain by the ecologists ©DEPE / CNRS.

In figure 7.13, we present two "*reference configuration*"[4]. The purpose of imaging the reference configurations is for the reconstruction of the terrain. In figure 7.16 and 7.17, we present some of the images taken for the penguins based on the defined protocol.

In figure 7.14 and figure 7.15 we present two extractions (print screen) from real data, GPS coordinates and laser measurements that is used for the reconstruction of terrain based on the defined protocol.

[4]Please refer to appendix A for a description of a "reference configuration"



(a) (b)

Figure 7.13: A proposed splitting for the terrain provided by the ecologists. It shows the regions of the same terrain slop ©DEPE / CNRS.

Conclusion on the proposed solution for King penguin counting

Although the existance of a protocol, and the huge effort accomplished by the ecologist to working with us to define the imaging protocol, and working on site under tough weather conditions for applying the protocol for the terrain reconstruction and imaging penguins, this task is very complex.

- Defining a reliable data term for similarity measure it not yet solved, it involves many of the object recognition challenges such as: occlusion which is one of the most complex problem in computer vision, and is very high in our data, with no obvious way to prevent it; form variability of the penguins (twisted head); illumination; ...
- Based on the ecologist feedback, applying the protocol is an exhaustive process, and the inserted sticks (refer to appendix A) disturb the penguins and should not remain fixed
- A big complexity comes from the topography of the place, to take an image with a high incidence angle, the ecologist has to go on hilltops. Unfortunately, the ramps of the hills are very small, which makes it almost impossible to be close to the colony and at high a incidence angle at the same time.

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	date de la mesure	Header	Name	Description	Type	Position	Altitude	Depth	Proximity	Temperature	Display Mode	Color	Symbol
2													
3													
4	8/13/2009	Waypoint	BA		User Waypoint	S46 25.548 E51 51.659	9 m				Symbol & Name	Unknown	Flag
5	8/13/2009	Waypoint	BB		User Waypoint	S46 25.546 E51 51.652	11 m				Symbol & Name	Unknown	Flag
6	8/13/2009	Waypoint	BC		User Waypoint	S46 25.547 E51 51.645	10 m				Symbol & Name	Unknown	Flag
7	8/13/2009	Waypoint	BD		User Waypoint	S46 25.547 E51 51.659	8 m				Symbol & Name	Unknown	Flag
8	8/13/2009	Waypoint	BE		User Waypoint	S46 25.542 E51 51.655	10 m				Symbol & Name	Unknown	Flag
9	8/13/2009	Waypoint	BF		User Waypoint	S46 25.543 E51 51.649	10 m				Symbol & Name	Unknown	Flag
10	8/13/2009	Waypoint	BG		User Waypoint	S46 25.540 E51 51.662	11 m				Symbol & Name	Unknown	Flag
11	8/13/2009	Waypoint	BH		User Waypoint	S46 25.538 E51 51.655	9 m				Symbol & Name	Unknown	Flag
12	8/13/2009	Waypoint	BI		User Waypoint	S46 25.537 E51 51.650	9 m				Symbol & Name	Unknown	Flag
13	8/13/2009	Waypoint	BR		User Waypoint	S46 25.550 E51 51.658	9 m				Symbol & Name	Unknown	Flag
14	8/13/2009	Waypoint	COINMU		User Waypoint	S46 25.553 E51 51.658	8 m				Symbol & Name	Unknown	Flag
15	8/13/2009	Waypoint	JA		User Waypoint	S46 25.518 E51 51.505	14 m				Symbol & Name	Unknown	Flag
16	8/13/2009	Waypoint	JB		User Waypoint	S46 25.516 E51 51.498	13 m				Symbol & Name	Unknown	Flag
17	8/13/2009	Waypoint	JC		User Waypoint	S46 25.517 E51 51.493	9 m				Symbol & Name	Unknown	Flag
18	8/13/2009	Waypoint	JD		User Waypoint	S46 25.513 E51 51.507	10 m				Symbol & Name	Unknown	Flag
19	8/13/2009	Waypoint	JE		User Waypoint	S46 25.512 E51 51.501	12 m				Symbol & Name	Unknown	Flag
20	8/13/2009	Waypoint	JF		User Waypoint	S46 25.513 E51 51.495	10 m				Symbol & Name	Unknown	Flag
21	8/13/2009	Waypoint	JG		User Waypoint	S46 25.510 E51 51.510	10 m				Symbol & Name	Unknown	Flag
22	8/13/2009	Waypoint	JH		User Waypoint	S46 25.509 E51 51.502	10 m				Symbol & Name	Unknown	Flag
23	8/13/2009	Waypoint	JI		User Waypoint	S46 25.508 E51 51.498	12 m				Symbol & Name	Unknown	Flag
24	8/13/2009	Waypoint	JR		User Waypoint	S46 25.521 E51 51.503	14 m				Symbol & Name	Unknown	Flag
25	8/13/2009	Waypoint	POTO1		User Waypoint	S46 25.585 E51 51.662	11 m				Symbol & Name	Unknown	Flag
26	8/13/2009	Waypoint	POTO2		User Waypoint	S46 25.556 E51 51.670	6 m				Symbol & Name	Unknown	Flag
27	8/13/2009	Waypoint	RA		User Waypoint	S46 25.591 E51 51.659	15 m				Symbol & Name	Unknown	Flag
28	8/13/2009	Waypoint	RB		User Waypoint	S46 25.588 E51 51.654	14 m				Symbol & Name	Unknown	Flag
29	8/13/2009	Waypoint	RC		User Waypoint	S46 25.589 E51 51.649	12 m				Symbol & Name	Unknown	Flag
30	8/13/2009	Waypoint	RD		User Waypoint	S46 25.588 E51 51.658	13 m				Symbol & Name	Unknown	Flag
31	8/13/2009	Waypoint	RE		User Waypoint	S46 25.587 E51 51.655	9 m				Symbol & Name	Unknown	Flag
32	8/13/2009	Waypoint	RF		User Waypoint	S46 25.587 E51 51.649	16 m				Symbol & Name	Unknown	Flag
33	8/13/2009	Waypoint	RG		User Waypoint	S46 25.585 E51 51.660	8 m				Symbol & Name	Unknown	Flag
34	8/13/2009	Waypoint	RH		User Waypoint	S46 25.584 E51 51.655	9 m				Symbol & Name	Unknown	Flag
35	8/13/2009	Waypoint	RI		User Waypoint	S46 25.586 E51 51.649	19 m				Symbol & Name	Unknown	Flag

Figure 7.14: A sample of the locations of sticks in each configuration. This data is collected on site by the ecologists using GPS ©DEPE / CNRS.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	de	vers	distance (m)	pente (°)		de	vers	distance (m)	pente (°)		de	vers	distance (m)	pente (°)	
2	RA	RB	6.77	3.0		BA	BB	8.08	-0.1		JA	JB	7.14	1.6	
3	RA	RD	6.69	-0.6		BA	BD	6.70	-1.7		JA	JD	7.08	-1.4	
4	RA	RE	10.03	1.2		BA	BE	9.15	-1.7		JA	JE	9.89	-0.2	
5	RA	RR	6.64	0.8		BA	BR	5.85	1.4		JA	JR	5.60	5.3	
6	RB	RA	6.76	-2.6		BB	BA	8.08	0.1		JB	JA	7.13	-1.4	
7	RB	RC	6.50	1.8		BB	BC	8.57	-0.6		JB	JC	7.63	3.6	
8	RB	RD	9.02	-2.4		BB	BD	10.84	-0.9		JB	JD	9.74	-2.1	
9	RB	RE	7.21	-1.0		BB	BE	6.97	-2.1		JB	JE	6.81	-1.8	
10	RB	RF	10.82	0.6		BB	BF	9.56	-1.6		JB	JF	10.10	0.4	
11	RB	RR	9.94	-1.4		BB	BR	9.64	1.1		JB	JR	9.39	2.1	
12	RC	RB	6.51	-1.7		BC	BB	8.55	0.7		JC	JB	7.62	-3.5	
13	RC	RE	8.79	-2.1		BC	BE	12.52	-0.7		JC	JE	11.84	-3.3	
14	RC	RF	7.38	-0.9		BC	BF	6.72	-1.5		JC	JF	8.00	-3.1	
15	RD	RA	6.70	0.4		BC	BR	16.84	1.0		JC	JR	15.03	-0.2	
16	RD	RB	9.02	2.6		BD	BA	6.75	1.5		JD	JA	7.10	1.6	
17	RD	RE	6.31	2.4		BD	BB	10.85	1.0		JD	JB	9.80	2.4	
18	RD	RG	6.96	1.8		BD	BE	7.39	-0.8		JD	JE	6.80	1.5	
19	RD	RH	10.28	3.1		BD	BG	6.08	-1.4		JD	JG	6.25	-0.3	
20	RE	RA	10.06	-1.1		BD	BH	10.71	-2.4		JD	JH	10.36	0.7	
21	RE	RB	7.24	1.2		BE	BA	9.76	1.7		JE	JA	9.93	0.2	
22	RE	RC	8.79	2.3		BE	BB	6.98	2.3		JE	JB	6.85	1.8	

Figure 7.15: A sample of the distances between sticks in each configuration. This data is collected on site by the ecologists using a laser measuring instrument ©DEPE / CNRS.

(a)

(b)

(c)

Figure 7.16: (a,b,c) Image samples of the Royal penguin colony based on the defined imaging protocol ©DEPE / CNRS.

(a)

(b)

Figure 7.17: (a,b,c) Another image sample of the Royal penguin colony based on the defined imaging protocol ©DEPE / CNRS.

Chapter 8

Conclusion and Recommendations

Markov point process is an interesting and flexible probabilistic framework. It is an object based method, and here lies its strength. It is more powerful and more natural to solve many image processing and computer vision problems on the *object level* rather than on the *pixel level* specially when considering high resolution images.

Markov point process is a superclass of the famous Markov random fields, while being much more flexible. In the prior term, we can insert any type of prior knowledge we have about the problem. The prior varies from favoring alignments between objects to defining minimal distances, based on the considered problem. This prior information is added to the prior term to regularize the solution. Data terms for object detection can range from simple correlation to using an SVM classifier trained on SIFT features for specific object category.

8.1 Optimization

During more than a decade, researchers have been developing point process models for image processing applications focusing on two tracks.

The first track concerned the models development, which is application based. Line point process model have been developed for road network extraction, such as the Candy model, Quality Candy model and IDQ model [63]. A rectangle point process model was developed for building detection. An ellipse point process model has been developed for trees and flamingos counting. We consider that the point process models in image processing can be classified into two categories based on the sophistication of the interactions. Trees and flamingos do not form special spatial structure, the prior term simply guarantees that objects are non-overlapping; and this is what we considered as the *first class* of Marked point process. Road network, and building are considered as being in *second class*, and they embed more sophisticated interactions, such as alignments, or angle between objets.

On the other side, samplers have also evolved. The development of these models was carried on with the development of optimizers, and this is the second track. With the increase of amount of collected data, and the increase in problem size, new optimization methods are required to cope with this demand. Point process samplers had started by the Birth-and-Death algorithm. This sampler was the first allowing a *dimensional jump* between spaces of different sizes, but from an image processing (application) point of view, it is a naive algorithm. Later came the RJMCMC sampler on top of this space dimensional jump idea. Even using only two kernels (birth, death), and even with the required burn in time, RJMCMC is faster than Birth-and-Death, since it has a lower order of complexity than the Birth and Death algorithm.

Both Birth-and-Death and RJMCMC are global-simple perturbation samplers. With the Metropolis Hasting scheme, RJMCMC became able to make also local perturbations, which made it faster and so more attractive to image processing applications. Although their speed gain, RJMCMC remains a slow optimization algorithm, it is a simple perturbation algorithm. This was the driver to develop the first multiple perturbation algorithm, which is the Multiple Birth-and-Death. This a global-multiple perturbation algorithm which has proved to be very efficient for the first class of Marked point process (*e.g.* the flamingo counting problem).

All previously mentioned samplers require a *simulated annealing* scheme, which is a major drawback. Setting the temperature and cooling parameter can also be a complex task. Indeed, certain conditions for those parameters have to be respected for convergence, which makes those algorithms slow.

In this thesis we focused on the first class of problems, objects with simple interactions. We proposed a new optimization method that bypasses many of the problems of previous samplers. The Multiple Birth and Cut algorithm has the following properties:

1. The very reduced number of parameters (no: δ, $\Delta\delta$ and γ (prior term weight))
2. No simulated annealing (β and $\Delta\beta$)
3. The speed of convergence (MBC3) much better than the MBD algorithm, whatever was the problem size.
4. Very rapidly the algorithm gives a very close estimation of the number of objects, far before full convergence. We only get a result from the MBD algorithm when it has almost converged.
5. Can be very easily parallelized, and can be used to parallelize other optimization algorithms.

6. Simpler for researchers unfamiliar with point process theory.

7. Simpler for user such as the ecologists for the flamingo counting problem.

The MBC algorithm is semi-deterministic, it has both a stochastic and a deterministic nature. In the Birth phase, it is fully stochastic to guarantee the exploration of the configuration space, while in the Cut phase, it is deterministic.

We next investigated, where does the speed limitation of this first proposed version of the algorithm comes from. Based on this investigation, we proposed a modification to the birth step. Since the limitation comes from that, during the Cut step, we can not solve the overlappings using graph cut unless both current and proposed configuration respect the non-overlapping condition. Based on that, we proposed to provide the Cut step with an optimized (selected) configuration. We found that when making pertinent proposals, the algorithm became *very fast*, much faster than the MBD algorithm and the first version of the MBC algorithm.

In a third version of the MBC algorithm, we proposed an more efficient *multiple-birth-death* kernel that the one proposed in [36]. The proposed kernel maked more efficient global perturbations, and explore more efficiently the configuration space, while being very simple. We next showed how *local perturbation* kernels can be integrated in the MBC algorithm to make it more efficient.

The MBC optimizer has proven to be simple and modular, while being very efficient. The proposed kernels are very simple and can easily be implemented. Moreover, adding new kernels to adapt the algorithm to specific applications, or even to tune the current algorithm, is possible without a lot of burden. Future work will consider extending this algorithm to solve more complex models, such as Point Process Models with sophisticated interactions.

8.1.1 Future work and perspective

In this work we only considered the first class of Markov marked point process models. The aim was to develop a new algorithm which overcomes the fully stochastic Multiple Birth-and-Death algorithm drawbacks. The reason for considering the flamingo counting problem is twofold, one is to compare the proposed algorithm with the fastest existing one, the second is to analyze how the algorithm scale with the problem size since the number of flamingos in a colony can go over 15,000 birds. We were able to take advantage of a very fast and efficient algorithm which is graph cut. Next we were able to combine in the birth-step another fast algorithm, which is belief propagation, to propose pertinent objects, and thus enhanced the speed of convergence of the proposed algorithm. We concluded with a very efficient multiple-birth-death kernel, and we integrated in a very simple way local

perturbations kernels to boosts the speed of the convergence even more.

Future work should consider the first class of Marked point process models where complex interactions exist. We believe that new optimization algorithms can solve this class of problems, where our algorithm modeling can be a source of inspiration. This is of great importance since this class can only be solved efficiently by RJMCMC algorithm, which is slow in real applications.

8.2 3D Point Process

The second part of this thesis was dedicated to the penguin counting problem using a 3D point process. We can deduce two main conclusions.

8.2.1 3D MPP model

In chapter 5, we presented a novel approach for multiple object detection under occlusions and perspective effects.

We proposed an alternative method for representing objects geometry. Even that for the selected application in chapter 7, using an ellipsoid would have been enough, but the aim was to propose a generic framework that can be easily extended to other cases, such as the airplanes detection example, where using a 3D CAD model can be of great interest. We recommend having a small set of parameters to cover real object variability. In our case, we could easily model the scaling factor in the three axes, but it would be better to have a control on the height and width independently. We could also have added another model representing the adult penguin case with a twisted head. This approach of using a 3D model is much more interesting than the englobing box or ellipse which can hardly define object borders.

The interest of the proposed *3D simulation based approach* is that: is 3D objects dimensions is used to regularize the result. More specifically, in the bounding box (or ellipse) method, only width and height of objects where taken into consideration, but not the depth, it was hard to evaluate the correctness of two proposed objects with occlusion, is the depth difference possible or not. This is the main advantage of the 3D model, the third dimension is an intrinsic part, given the camera position with respect to the scene, it becomes easy to define the minimal depth difference, no pair of objects can intersect in the real world, it is a physical limitation. This non-overlapping is not valid when considering the projection onto the image plan. Using the three dimensions of the object for regularization is more natural and may improve detection results.

We also modeled dependencies between objects in 3D, which is usually neglected by

many multiple object detection under occlusions effects in problems such as crowd counting. Usually methods assume the independence between objects, which we consider as a wrong hypothesis. We discussed the dependencies between objects either concerning the data term or the prior term.

8.2.2 Future work on the 3D model

We proposed a modeling for the dependencies, and showed how to manage these dependencies by a mixed graph. This modeling is *generic*, not specific the MPP models, it can be applied to other frameworks. More investigations should be done to define on appropriate data term, and to recover camera parameters.

The proposed framework for solving the penguin counting problem is interesting but is facing many complexities. Despite the defined protocol, the nature and topology of the occupied region does not facilitate the imaging task, based on the ecologist on site feedback. An alternative method should be proposed.

8.2.3 Future work of the Penguin counting problem

The main origin of complexity for the penguin counting problem comes from the imaging conditions. We propose testing a different imaging approach. We propose taking aerial images using a new kind of unmanned aerial vehicle (UAV) such as quadcopter. Usually areal coverage by airplanes is forbidden, but it may be accepted from such small UAV. Quadcopter or hexa or octa, are from the family of Vertical Take-off and Landing (VTOL). This type of UAV has only gained popularity during the last few years, after developing efficient algorithms to control the n-rotors speed, to achieve a harmony and maneuver capabilities [86]. This UAV usage is used in many application, from fireman help, to shooting movies. The interesting property about this family of UAV is the simplicity of usage and navigation, specially being one of the VTOL family. The price of such an UAV for professional usage varies from 1,500 to 30,000 euros.

We propose using the basic non expensive one, and adding a camera on board. Planing the UAV trip, exact positions (x,y and z) for imaging, can be very easily done using existing open source software, such as ArduPilot Mega [1]. An example of a user interface of such a software used for planning the trip is presented in figure 8.1. We propose using such a UAV and software, to take top views photos of the penguin colony. Wind speed is an issue in the south pole, but we believe that, given the usage simplicity (trip planning is a simple script), collecting images for the colony will be possible. Even if the colony occupies a large terrain with a complex topography. From the obtained aerial images, we can use existing stitching

[1] http://code.google.com/p/arducopter/wiki/ArduCopter

methods to generate an image that covers the whole colony, and next using a 2D MPP model will be a much simpler task to accomplish the counting process.

Figure 8.1: ArduPilot Mega (APM), a sophisticated open source autopilot.

Under the hope for acceptable wind conditions (up to 20km/h is acceptable for certain small UAV), this proposal, would eliminate the following problems:

- Complexity of the images acquisition
- Occlusion in the collected images
- Perspective effects in the collected images

Appendix, and Bibliography

Appendix A

Publications

Journals:

- 2011
  - A. Gamal-Eldin, X. Descombes, G. Charpiat, J. Zerubia. Multiple Birth and Cut Algorithm for Multiple Object Detection. International Journal of Multimedia Processing and Technologies. Accepted in april 2011.

Conferences:

- 2013: Accepted
  - A. Gamal-Eldin, G. Charpiat, X. Descombes, J. Zerubia. An efficient optimizer for simple point process models. IS&T/SPIE Electronic Imaging, California, United States, 3-7 February, 2013.
- 2011
  - A. Gamal-Eldin, X. Descombes, J. Zerubia. A Novel Algorithm for Occlusions and Perspective Effects using 3D Object Process. ICASSP, Prague, Czech Republic, 22-27 May, 2011.
  - A. Gamal-Eldin, X. Descombes, G. Charpiat, J. Zerubia. A Fast Multiple and Cut Algorithm using Belief Propagation. Accepted at ICIP, Brussels, Belgium, 11-14 September, 2011.
  - X. Descombes , A. Gamal-Eldin, F. Plouraboué, C. Fonta, R. Serduc, G. Leduc, T. Weitkamp, Extraction et caracterisation de regions saines et pathologiques a partir de micro-tomographie du systeme vasculaire cerebral, GRESTI, Bordeaux, 5-8 September, 2011.

- 2010
  - A. Gamal-Eldin, X. Descombes, J. Zerubia. AMultiple Birth and Cut Algorithm for Point Process Optimization. SITIS, Kuala Lumpur, Malaysia, 15-18 Decembre, 2010. (**Best paper award**).
  - A. Gamal-Eldin, F. Salzenstien, C. Collet. Hidden Fuzzy Markov Chain Model with K-Discrete Classes. ISSPA, Kuala Lumpur, Malaysia, 10-13 May, 2010.

Appendix B

Imaging Protocol

B.1 Introduction

The aim of this imaging protocol documentation is to define a standard between both the researches at Ariana / INRIA / Sophia-Antipolis and the researches of Tour du Valat.

This protocol concerns imaging two elements, the king penguins and the territory occupied by the colony of interest. We are going to detail the description or the required images for both elements, and we will be waiting for a reply from your side (ecologists on site) about the applicability of our request so we can reach an appropriate description of this protocol.

In this documentation we will start by mentioning the required material, we will define how and where to take images for the territory imaging, we will define the different cases for penguin images and we will state technical details about camera usage.

The required labor and material: Our penguins are living in a region that is not completely horizontal, and in our model the elevation information is important. Since this "Digital Elevation Model" information is not available for Possession Island, we will reconstruct it using the images that you are going to provides us with. The only available data at the IGN [1] is a map.

B.2 Image for the territory elevation

B.2.1 The labor and materials

1. Two people to take the measurements

2. One (ore more) photo camera(s) (whose specifications will be detailed later) with a tripod

[1] Institut National de l'information Géographique et forestière

3. One flat panel, say 0.5 meter x 0.5 meter, made of wood or metal this will be the reflective surface for the laser distance measurer
4. A laser distance measurer
5. A GPS locator
6. Thirty wooden or metallic sticks, length about 1m (meter) to put on the soil in vertical position
7. Three small flags of different colors, these will form, with the sticks, the reference objects

In what follows, I will use the following conventions:

A reference configuration is a set of 10 visible objects that you will put somewhere (to be defined later) in the landscape, as show in figure B.1. A reference configuration is composed by 9 reference objects (in capital letter from A-I) and a pivot (P). The 9 reference objects will be on a 3-by-3 grid, while the pivot will be put close to a corner of the resulting square, aligned with one of the sides that intersects in that corner.

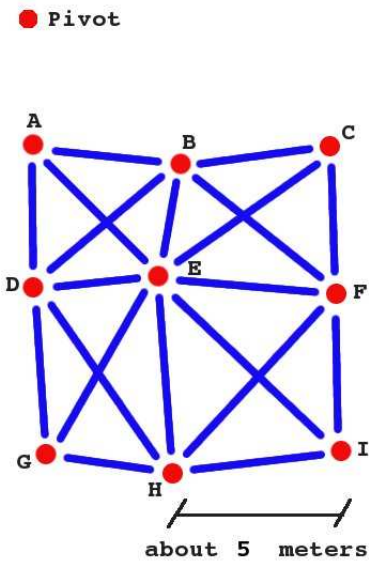


Figure B.1: A graph showing a possible configuration

In this figure we recognize the pivot that will be next to the first point (A) on the grid and aligned with the two points A and D . As clear in this figure, the shape does not have to be a perfect square, the side lengths could vary and the angles does not have to 90. The grid nine points are:

- the places where we want to install a stick in each point
- get the GPS position of each point on the ground[2] not on the top of the stick

Measuring the distance between two points:

The blue lines represent the distances between the points (sticks) that need to be measured. The distance between the points should be from 5 to 10 meters (10m is better) . This distance is the one we need to measure using the laser measure and the reflective surface as show in figure B.1 and B.2.

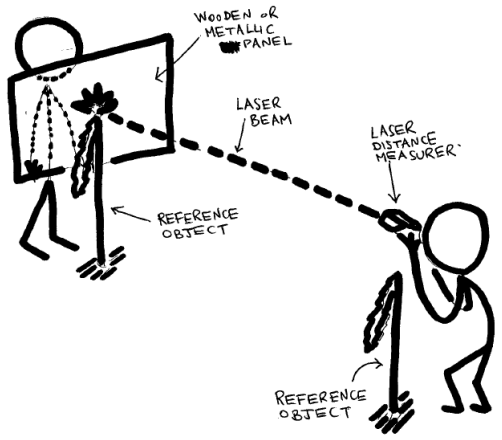


Figure B.2: Measuring distance using the laser measurer

For the laser measurer, you can see as example this web catalog: http://www.professionalequipment.com/laser-measuring/

We need the distance between two points (example stick A and B) at the same height. This mean that at for example 70cm height from the ground. This could be simply done if each time you insert a stick, you add a marker to the position on this stick that is 70cm height from the ground.

For each configuration, we suggest the following form to store the information:

We need the GPS measurements, the distances and the photos. For example:

The Green configuration: (green is the color of the pivot flag in a certain zone, detailed in the next point)

A = [GPS coordinate] B = [GPS coordinate] C = [GPS coordinate] ... AB = [distance in centimeter,meters] AE = [distance] AD = [distance] BE = [distance]

[2] All GPS measurements should be at the ground level.

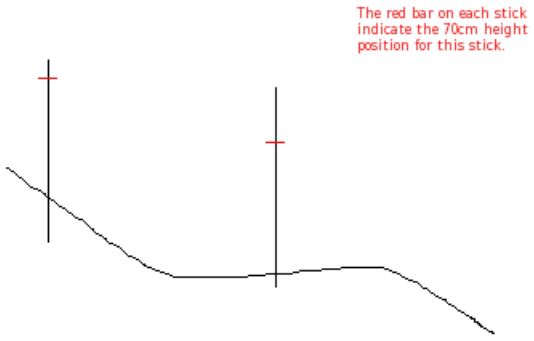


Figure B.3: The height need to be measured at the same level from the ground

B.2.2 The number of required configurations

As we mentioned previously, the elevation information is important for our work. We take as an example of an image that you provided us show in figure B.4.

For the image show in figure (4), we made a rough segmentation of the image into closed red zones. We did not consider all the image because the part that represents very far object in less obvious for us. From what we can see from this image, we think that those zones are almost flat inside each region. We need a configuration inside each zone around its center.

This is an example that we suggest based on this image, but since you are in the field, we leave this task for you.

If we assume using the example presented in figure B.4., then we need:

- six configurations
- the GPS of the four corners of the outer fence (the wall surrounding the station)
- three photos for each configuration, as will be detailed next.

We also want a photo of the complete scene like the one presented in figure B.4.

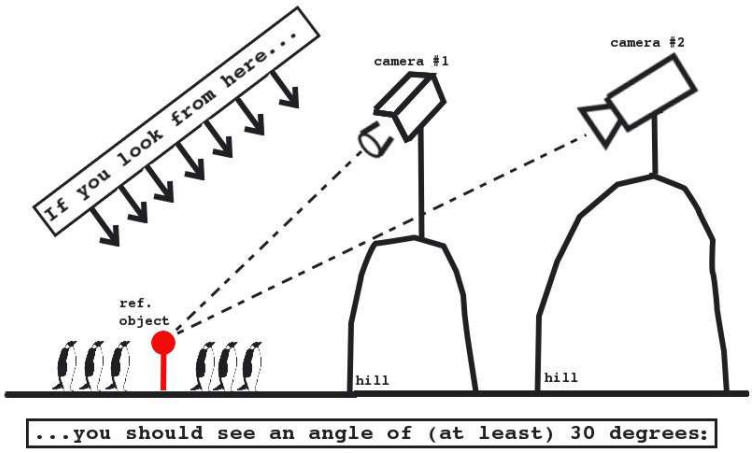
B.2.3 The method defined to take image for each configuration

We assume you centered (as much as possible) you configuration inside each zone. We need the following for the three photos of each configuration:

Figure B.4: Example of possible division to determine the number of configurations

1. Position each camera on the highest possible place, like a hill, as show in figure B.5.
2. Keep the same height (altitude) for the 3 cameras (by using the altitude provided by GPS)
3. Point the camera toward the central object of the configuration, which is point E in figure B.1
4. The angle between the cameras should me almost 90, but if not possible it has to be at least more than 30, as show in figure B.5.
5. The time interval between the image you capture with camera 1,2 and 3 should be as minimum as possible, simultaneous capturing will be perfect.

For all the configurations, since the target of those images is to reconstruct the elevation of the region of interest, the ideal is to take the images where the penguins are not present. Since this is not possible, we want to repeat the imaging of all of the configurations two or three times when the penguins moves a lot, we mean to take image for example at day 1 and day 5 when most of the penguins almost changed or left there places. But this request depends on the possibilities.



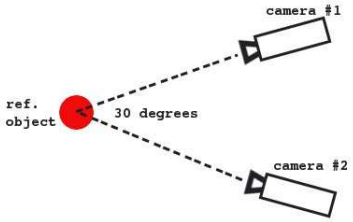


Figure B.5: (a) Imaging from hill and pointing to the central element. (b) Imaging with an angle of 30.

B.3 Imaging for penguins

For the mean time, we need images for simple regions and with a lot of variability as follows:

1. Images for penguins on regions as flat and horizontal as possible
2. Images when the penguins are close together, others when they are far from each other
3. Different illumination conditions, we need images some with shadow and others without (at 12:00)
4. Images dense/sparse with penguins

We need if possible a special imaging for four or five penguins taken apart from multiple view (5 or 6 images with different angle all around it) , or even a video of the penguin

by rotating about it (360).

For imaging the whole colony, there is three possible ways:

1. Take an image from a long distance to cover the whole colony.
2. Take multiple images using the panoramic view provided by the camera.
3. Take multiple images, and we will combine them.

All types are of interest for us.

For case 3 multiple images, and we will combine them, we have some comments:

a) If we can take the whole colony in just two images, we prefer that the two images to be taken simultaneously.

b) For two or more divisions (to covert the whole colony), the images should superimpose (overlap) for a portion of the image.

c) For more than two images, the more the imaging positions are aligned, the better for us.

d) When imaging the whole colony with any technique, (1 image, panorama or multiple), if we assume the colony takes a rectangular shape, take photos fronting the large side

Description of (a) (d) on the following sketch:

For (a,b), as shown in then next figure, if we use 2 or 3 cameras, there is a overlap between the region covered by camera 1 and camera 2, and also between camera 2 and camera 3.

For (c), The alignment of the multiple camera positions is described by **LIGNE A** in the next figure, (knowing that they will not be simultaneous because you have only two cameras). We need to be as much as possible parallel (at the same distance) to the colony, and if possible also to keep almost the same height (We know that this depends so much on the place).

For (d), as shown by the arrows in the above figure B.6, we want the imaging to be perpendicular to the direction of spreading of the colony. In the next two images, we assume that this is all the colony, the first image show a perfect arrangement, where there is no or minimum occlusion between the penguins.

The following image (figure **??**), show the problem of shooting in a direction which is not perpendicular to the direction of spreading of the colony.

We see that in the middle, a lot of penguins overlap. So we want to minimize this effect as much as possible, and we believe that being perpendicular to the direction of spreading, will gives the better results.

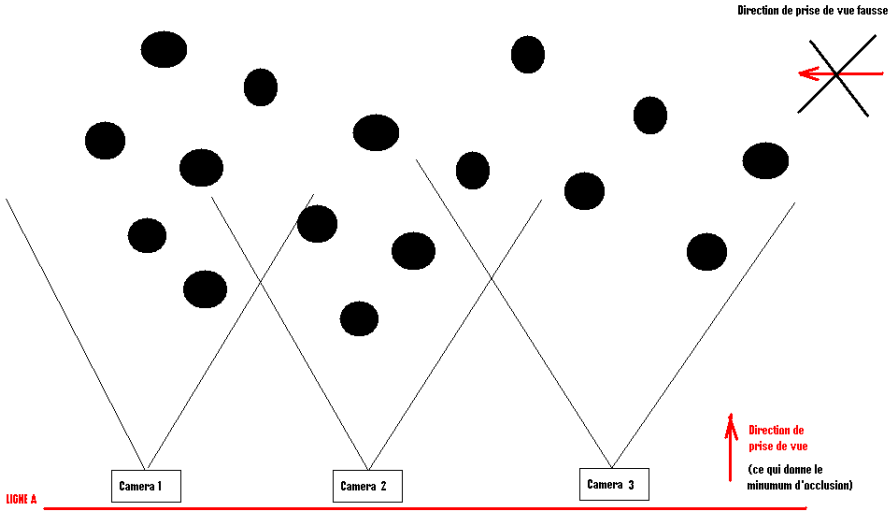


Figure B.6

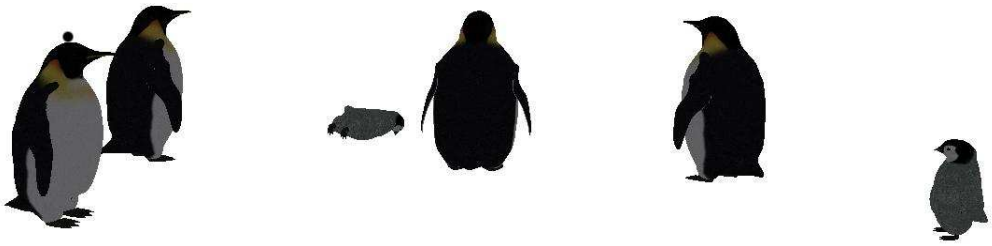


Figure B.7

B.3.1 Technical details about the camera

We have some issues about images you previously provided us, we found that camera is a little bit old and this has a severe effect of the image quality presented in the number of mega-pixels, and other issues.

Here we defined our requirements for the camera and the capturing parameters:

1. Use a new camera with around 10MP (mega pixels)
2. When capturing the image, select the maximum resolution
3. Modify the imaging parameters to save the images in RAW format (format brut), the default format is for example jpeg, it compress images using a lossy technique, and

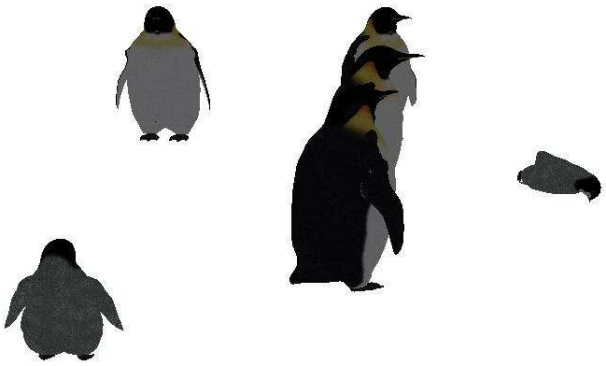


Figure B.8

we need images rich of information.

4. If you have the option to save Exif data, we need it, it contains the capturing parameters

5. While capturing, you can use optical zoom, please disable digital zoom.

We tried to be clear in this documentation, but if you found any point or technical detail not so clear even if it is like camera parameter configuration, we will be more than happy to answer.

Bibliography

[1] S. Agarwal, N. Snavely, I. Simon, S. Seitz, and R. Szeliski. Building rome in a day. *2009 IEEE 12th International Conference on Computer Vision*, pages 72–79, 2009.

[2] C. Andrieu, Nando de Freitas, A. Doucet, and M. I. Jordan. An introduction to MCMC for machine learning, september 2001.

[3] C. Andrieu and A. Doucet. Joint Bayesian model selection and estimation of noisy sinusoids via Reversible Jump MCMC. *IEEE Trans. on Signal Processing*, 47(10):2667 –2676, oct 1999.

[4] A. Baddeley, I. Barany, and R. Schneider. Spatial point processes and their applications. In *Stochastic Geometry*, volume 1892 of *Lecture Notes in Mathematics*, pages 1–75. Springer Berlin, 2007.

[5] A. Baddeley and R. Turner. Modelling spatial point patterns in R. In *Case Studies in Spatial Point Pattern Modelling. Lecture Notes in Statistics 185, 2374*. Springer, 2006.

[6] A. J. Baddeley and M. N. M. Van Lieshout. Stochastic geometry models in high-level vision. *Journal of Applied Statistics*, 20(5-6):231–256, 1993.

[7] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Comput. Vis. Image Underst.*, 110:346–359, June 2008.

[8] A. Bechet, A. Reed, N. Plante, J. F. Giroux, and G. Gauthier. Estimating the size of the greater snow goose population. *Journal of Wildlife Management*, 68(3):639–649, 2004.

[9] S. Ben Hadj, F. Chatelain, X. Descombes, and J. Zerubia. Parameter estimation for a marked point process within a framework of multidimensional shape extraction from remote sensing images. In *Proc. ISPRS Technical Commission III Symposium on Photogrammetry Computer Vision and Image Analysis (PCV)*, Paris, France, September 2010.

[10] C. Benedek, X. Descombes, and J. Zerubia. Building detection in a single remotely sensed image with a point process of rectangles. In *Proc. International Conference on Pattern Recognition (ICPR)*, Istanbul, Turkey, August 2010.

[11] J. Besag. Spatial Interaction and the Statistical Analysis of Lattice Systems. *Journal of the Royal Statistical Society. Series B (Methodological)*, 36(2):192–236, 1974.

[12] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[13] A. Blake, C. Rother, M. Brown, P. Perez, and P. Torr. Interactive image segmentation using an adaptive gmmrf model. In *Proc. European Conference on Computer Vision (ECCV)*, Prague, May 2004.

[14] R. Bogdan, A. Holzbach, N. Blodow, and M. Beetz. Fast Geometric Point Labeling using Conditional Random Fields. In *Proceedings of the 22nd IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, St. Louis, MO, USA, October 11-15 2009.

[15] Y. Boykov. Computing geodesics and minimal surfaces via graph cuts. In *Proc. International Conference on Computer Vision (ICCV)*, pages 26–33, 2003.

[16] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 26(9):1124–1137, September 2004.

[17] Y. Boykov, O. Veksler, and R. Zabih. Markov random fields with efficient approximations. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 648–655, 1998.

[18] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 23(11):1222–1239, November 2002.

[19] W. Burgin, C. Pantofaru, and William D. Smart. Using depth information to improve face detection. In *Proceedings of the 6th international conference on Human-robot interaction*, HRI '11, pages 119–120, New York, NY, USA, 2011. ACM.

[20] F. Chatelain, X. Descombes, and J. Zerubia. Parameter estimation for marked point processes. application to object extraction from remote sensing images. (poster). In *Proc. Energy Minimization Methods in Computer Vision and Pattern Recognition (EMMCVPR)*, Bonn, Germany, August 2009.

[21] T.F. Cootes and C.J.Taylo. Statistical models of appearance for computer vision. Technical report, Imaging Science and Biomedical Engineering, University of Manchester, U.K., march 2004.

[22] T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson. *Introduction to Algorithms*. McGraw-Hill Higher Education, 3rd edition, 2009.

[23] J. Neider D. Shreiner, M. Woo and T. Davis. *OpenGL(R) Programming Guide*. Addison-Wesley Professional, 6 edition, 2007.

[24] Daryl J. Daley and David Vere Jones. *An introduction to the theory of point processes*, volume 2. Springer, 2008.

[25] S. Descamps, X. Descombes, A. Béchet, and J. Zerubia. Automatic flamingo detection using a multiple and death process. In *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Las Vegas, USA, March 2008.

[26] S. Descamps, X. Descombes, A. Béchet, and J. Zerubia. Détection de flamants roses par processus ponctuels marqués pour l'estimation de la taille des populations. *Traitement du Signal*, 26(2):95–108, July 2009.

[27] S. Descamps, M. Gauthier-Clerc, J. Gendner, and Yvon L. Maho. The annual breeding cycle of unbanded king penguins Aptenodytes patagonicus on Possession Island (Crozet). *Avian Science*, 2:1–12, 2002.

[28] X. Descombes. *Stochastic Geometry for Image Analysis*. Wiley-ISTE, 2011.

[29] X. Descombes, R. Minlos, and E. Zhizhina. Object extraction using a stochastic birth-and-death dynamics in continuum. Research Report 6135, INRIA, 2007.

[30] X. Descombes, R. Minlos, and E. Zhizhina. Object extraction using a stochastic birth-and-death dynamics in continuum. *Journal of Mathematical Imaging and Vision*, 33(3):347–359, 2009.

[31] X. Descombes, M. Sigelle, and F. Preteux. GMRF parameter estimation in a non-stationary framework by a renormalization technique: Application to remote sensing imaging. *IEEE Trans. Image Processing*, 8(4):490–503, 1999.

[32] G. Dong and S.T. Acton. Detection of rolling leukocytes by marked point processes. *Journal of Electronic Imaging*, 16(3), 2007.

[33] M. Erikson. Two preprocessing techniques based on grey level and geometric thickness to improve segmentation results. *Pattern Recognition Letters*, 27(3):160 – 166, 2006.

[34] P. Muse F. Cao, Y. Gousseau, P. Muse, and F. Sur. Unsupervised thresholds for shape matching. In *In Proceedings of the IEEE International Conference on Image Processing*, pages 647–650, 2003.

[35] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Qian Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker. Query by image and video content: the qbic system. *Computer*, 28(9):23 –32, sep 1995.

[36] A. Gamal Eldin, X. Descombes, Charpiat G., and J. Zerubia. A fast multiple birth and cut algorithm using belief propagation. In *Proc. IEEE International Conference on Image Processing (ICIP)*, Brussels, Belgium, septembre 2011.

[37] A. Gamal Eldin, X. Descombes, Charpiat G., and J. Zerubia. Multiple birth and cut algorithm for multiple object detection. *Journal of Multimedia Processing and Technologies*, 2011.

[38] A. Gamal Eldin, X. Descombes, and J. Zerubia. Multiple birth and cut algorithm for point process optimization. In *Proc. IEEE SITIS*, Kuala Lumpur, Malaysia, December 2010.

[39] M. G. Gauthier. *Poles en peril*. Buchet Chastel, 2007.

[40] M. Gauthier-Clerc, J P Gendner, C A Ribic, W R Fraser, E J Woehler, S Descamps, C Gilly, C Le Bohec, and Y Le Maho. Long-term effects of flipper bands on penguins. *Proceedings of the Royal Society B: Biological Sciences*, 271 Suppl(September):S423–6, 2004.

[41] W. Ge and R.T. Collins. Marked point processes for crowd counting. In *Proc. Computer Vision and Pattern Recognition*, Miami, USA, july 2009.

[42] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6):721–741, November 1984.

[43] C. J. Geyer and Jesper M. Simulation Procedures and Likelihood Inference for Spatial Point Processes. *Scandinavian Journal of Statistics*, 21(4):359–373, 1994.

[44] G. Gherdovich and X. Descombes. Two dof camera pose estimation with a planar stochastic reference grid. *VISAPP*, 0:762, 2010.

[45] W.R. Gilk, S. Richardson, and David Spiegelhalter. *Markov Chain Monte Carlo in Practice*. Chapman and Hall/CRC, 1995.

[46] P. J. Green. Reversible jump Markov Chain Monte Carlo computation and bayesian model determination. *Biometrika*, 82:711–732, 1995.

[47] D. M. Greig, B. T. Porteous, and A. H. Seheult. Exact maximum a posteriori estimation for binary images. *Journal of the Royal Statistical Society*, 51(2):271–279, 1989.

[48] C. Harris and M. Stephens. *A combined corner and edge detector*, volume 15. Manchester, UK, 1988.

[49] J. Huang, S. Ravi Kumar, M. Mitra, W. Zhu, and R. Zabih. Image indexing using color correlograms. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 0:762, 1997.

[50] H. Ishikawa. Exact optimization for Markov random fields with convex priors. *Proc. IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 25:1333–1336, 2003.

[51] H. Ishikawa and D. Geiger. Mapping image restoration to a graph problem. In *Proc. of IEEE-EURASIP Workshop on Nonlinear Signal and Image Processing*, Antalya, Turkey, June 1999.

[52] M. Urban J. Matas, O. Chum and T. Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. *Image Vision Computing*, pages 761–767, 2004.

[53] Eva B. Vedel Jensen and L. Stougaard Nielsen. A review on inhomogeneous markov point processes. *Lecture Notes-Monograph Series*, 37:pp. 297–318, 2001.

[54] A. E. Johnson and M. Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 433–449, 1999.

[55] Karantzalos K. and Paragios N. Recognition-driven 2d competing priors towards automatic and accurate building detection. *IEEE Transactions on Geoscience and Remote Sensing*, 47(1):133–144, 2009.

[56] H. Kesten. *Percolation Theory for Mathematicians*. Number 2 in Progr. Prob. Statist. Birkhäuser, Mass., 1982.

[57] V. Kettnaker and R. Zabih. Counting people from multiple cameras. *Multimedia Computing and Systems, International Conference on*, 2:267, 1999.

[58] J. Kim, V. Kolmogorov, and R. Zabih. Visual correspondence using energy minimization and mutual information. In *Proc. International Conference on Computer Vision (ICCV)*, 2, pages 1033–1040, October 2003.

[59] P. Kohli and P. H. S. Torr. Dynamic graph cuts for efficient inference in Markov random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 29(12):2079–208, 2007.

[60] V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts? *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 26(2):147–159, February 2004.

[61] M. S. Kulikova, I. H. Jermyn, X. Descombes, E. Zhizhina, and J. Zerubia. Extraction of arbitrarily shaped objects using stochastic multiple birth-and-death dynamics and active contours. In *Proc. IS&T/SPIE Electronic Imaging*, San Jose, USA, January 2010.

[62] David G. L. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, 2004.

[63] C. Lacoste, X. Descombes, and J. Zerubia. A comparative study of point processes for line network extraction in remote sensing. Research Report 4516, Inria, France, July 2002.

[64] T. Lindeberg. Scale-space theory: A basic tool for analysing structures at different scale. *Journal of Applied Statistics*, 21:224–270, 1994.

[65] Y. Liu, O. Veksler, and O. Juan. Simulating classic mosaics with graph cuts. In *Proc. Energy Minimization Methods in Computer Vision and Pattern Recognition (EMMCVPR)*, pages 55–70, 2007.

[66] A. Lorette, X. Descombes, and J. Zerubia. Texture analysis through a markovian modelling and fuzzy classification: Application to urban area extraction from satellite images. *International Journal of Computer Vision*, 36(3):221–236, 2000.

[67] David G. Lowe. Object recognition from local scale-invariant features. In *International Conference on Computer Vision (ICCV)*, pages 1150–1157, Greece, 1999.

[68] K. Mikolajczyk and C. Schmid. Indexing based on scale invariant interest points. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 1, pages 525 –531 vol.1, 2001.

[69] K. Mikolajczyk and C. Schmid. Scale & affine invariant interest point detectors. *Int. J. Comput. Vision*, 60:63–86, October 2004.

[70] K. Mikolajczyk, C. Schmid, and A. Zisserman. Human detection based on a probabilistic assembly of robust part detectors. pages 434–444, 2009.

[71] J. Moller and R. P. Waagepetersen. *Statistical Inference and Simulation for Spatial Point Processes*. Chapman and Hall, 2004.

[72] N. Paragios, Y. Chen, and O. Faugeras. *Handbook of Mathematical Models in Computer Vision*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.

[73] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, 1988.

[74] G. Perrin, X. Descombes, and J. Zerubia. 2D and 3D vegetation resource parameters assessment using marked point processes. In *Proc. International Conference on Pattern Recognition (ICPR)*, Hong-Kong, August 2006.

[75] G. Perrin, X. Descombes, J. Zerubia, and J.G. Boureau. Forest resource assessment using stochastic geometry. In *Proc. Int. Precision Forestry Symposium*, March 2006.

[76] R. Pflugfelder and H. Bischof. Localization and trajectory reconstruction in surveillance cameras with nonoverlapping views. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32:709–721, 2009.

[77] Christian P. Robert and George Casella. *Monte Carlo Statistical Methods*. Springer Verlag, 1999.

[78] J. Lee Rodgers and W. Alan Nicewander. Thirteen ways to look at the correlation coefficient. *The American Statistician*, 42(1):59–66, 1988.

[79] S. Roy and I. J. Cox. A maximum-flow formulation of the n-camera stereo correspondence problem. *IEEE International Conference on Computer Vision (ICCV)*, 0:492, 1998.

[80] R. Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms FPFH for 3D registration. In *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pages 3212 –3217, may 2009.

[81] N. Thongsak S. Tongphu and M. N. Dailey. Rapid detection of many object instances. 2004.

[82] Claire Saraux, Céline Le Bohec, Joel M Durant, Vincent A Viblanc, Michel Gauthier-Clerc, David Beaune, Young-Hyang Park, Nigel G Yoccoz, Nils C Stenseth, and Yvon Le Maho. Reliability of flipper-banded penguins as indicators of climate change. *Nature*, 471(7329):203–206, 2011.

[83] R. Stoica, X. Descombes, and J. Zerubia. A Gibbs point process for road extraction in remotely sensed images. *International Journal of Computer Vision (IJCV)*, 57(2):121–136, 2004.

[84] D. Stoyan and H. Stoyan. *Fractals, random shapes, and point fields*. Wiley, 1994.

[85] M. Stricker, A. Dimai, and E. Dimai. Color indexing with weak spatial constraints. In *Proc. SPIE Storage and Retrieval for Image and Video Databases*, pages 29–40, 1996.

[86] A. Tayebi and S. McGilvray. Attitude stabilization of vtol quadrotor aircraft. *IEEE Transactions on Control Systems Technology*, 14(3):562–571, may 2006.

[87] M. N. M. van Lieshout. *Markov Point Processes and Their Applications*. Imperial College Press, 2000.

[88] P. Viola, Michael J. Jones, and D. Snow. Detecting pedestrians using patterns of motion and appearance. *Computer Vision, IEEE International Conference on*, 2, 2003.

[89] Paul A. Viola and Michael J. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154, 2004.

[90] B. Wu and R. Nevatia. Detection of multiple, partially occluded humans in a single image by bayesian combination of edgelet part detectors. In *Proceedings of the Tenth IEEE International Conference on Computer Vision (ICCV)*, volume 1, pages 90–97, Washington, DC, USA, 2005. IEEE Computer Society.

[91] B. Zhan, D. Monekosso, P. Remagnino, S. Velastin, and L. Xu. Crowd analysis: a survey. *Machine Vision and Applications*, 19:345–357, 2008. 10.1007/s00138-008-0132-4.

Abstract: This thesis work can be presented as two parts:

First part:

We proposed a novel probabilistic approach to handle occlusions and perspective effects (challenging problems in computer vision) for 3D object detection from a 2D image. The proposed method is based on 3D scene simulation on the GPU using OpenGL. Candidates configurations are proposed, simulated on the GPU and projected onto the image plane. Configurations are modified until convergence using an appropriate optimization algorithm.

Second part:

We proposed a new optimization method for Point Process models, which is a interesting framework for solving many challenging problems dealing with high resolution images. Our optimization method which we call "Multiple Births and Cut" (MBC), is the only semi-deterministic optimiser for the point process models. Our proposed algorithm overcomes all previously existing optimisers in terms of: speed, simplicity, reduced and simplified set of parameters, and modularity.