Probabilistic methods for point tracking and biological image analysis

Maël Primet

MAËL PRIMET

PROBABILISTIC METHODS FOR POINT TRACKING AND BIOLOGICAL IMAGE ANALYSIS

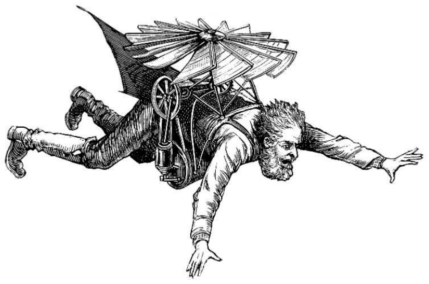
MÉTHODES PROBABILISTES POUR LE SUIVI DE POINTS ET L'ANALYSE D'IMAGES BIOLOGIQUES

THÈSE DE MATHÉMATIQUES
pour obtenir le grade de
Docteur de l'Université Paris Descartes
dans la spécialité Mathématiques appliquées

soutenue le 25 novembre 2011 devant le jury composé de

Patrick Bouthemy	rapporteur	INRIA Rennes
François Fleuret	rapporteur	IDIAP Research Institute
Isabelle Bloch	examinatrice	Télécom Paristech
Julie Delon	examinatrice	Télécom Paristech
Joan Glaunès	examinateur	Université Paris Descartes
Ariel Lindner	examinateur	Université Paris Descartes
Lionel Moisan	directeur de thèse	Université Paris Descartes

Université Paris Descartes
UFR Mathématiques Informatique
École Doctorale Mathématiques Paris-Centre



Probabilistic methods for point tracking and biological image analysis
Maël Primet, 25 novembre 2011.

RÉSUMÉ

Nous nous intéressons dans cette thèse au problème du suivi d'objets, que nous abordons par des méthodes statistiques. La première contribution de cette thèse est la conception d'un algorithme de suivi de bactéries dans une séquence d'image et de reconstruction de leur lignage, travail ayant donné lieu à la réalisation d'une suite logicielle aujourd'hui utilisée dans un laboratoire de recherche en biologie. La deuxième contribution est une étude théorique du problème de la détection de trajectoires dans un nuage de points. Nous définissons un détecteur de trajectoires utilisant le cadre statistique des méthodes *a contrario*, qui ne requiert essentiellement aucun paramètre pour fonctionner. Ce détecteur fournit des résultats remarquables, et permet notamment de retrouver des trajectoires dans des séquences contenant un grand nombre de points de bruit, tout en conservant un taux de fausses détections de trajectoires très faible. Nous étudions ensuite plus spécifiquement le problème de l'affectation de nuages de points entre deux images, problème rencontré notamment pour la détection de trajectoires ou l'appariement d'images stéréographiques. Nous proposons d'abord un modèle théoriquement optimal pour l'affectation de points qui nous permet d'étudier les performances de plusieurs algorithmes classiques dans différentes conditions. Nous formulons ensuite un algorithme sans paramètre en utilisant le cadre *a contrario*, ce qui nous permet ensuite d'obtenir un nouvel algorithme de suivi de trajectoires.

ABSTRACT

The subject of this thesis is the problem of object tracking, that we approached using statistical methods. The first contribution of this work is the conception of a tracking algorithm of bacterial cells in a sequence of image, to recover their lineage; this work has led to the implementation of a software suite that is currently in use in a research laboratory. The second contribution is a theoretical study of the detection of trajectories in a cloud of points. We define a trajectory detector using the a-contrario statistical framework, which requires essentially no parameter to run. This detector yields remarkable results, and is in particular able to detect trajectories in sequences containing a large number of noise points, while keeping a very low number of false detections. We then study more specifically the correspondence problem between two point clouds, a problem often encountered for the detection of trajectories or the matching of stereographic images. We first introduce a theoretically optimal model for the point correspondence problem that makes it possible to study the performances of several classical algorithms in a variety of conditions. We then formulate a parameterless point correspondence algorithm using the a-contrario framework, that enables us to define a new trajectory tracking algorithm.

Quand on ne sait rien,
on peut tout de même trouver des choses,
avec de l'imagination.
— Boris Vian

Remerciements

Il est bien entendu qu'il faut être malade pour faire de la recherche. Cette maladie vitale, c'est l'optimisme. Je suis heureux que Lionel en ait développé un cas fort contagieux.

Je remercie donc Lionel Moisan, mon directeur de thèse, pour son soutien : je dois certainement plus à lui qu'à quiconque, et peut-être même qu'à moi, que ce projet ait pu aboutir. Nos échanges et ses enseignements ont été la source d'une riche stimulation ; pour moi certainement, pour lui je l'espère. Sa recherche systématique de la clarté et de la précision m'a beaucoup aidé, moi qui me satisfait souvent de dessins rigolos et de mots qu'on gribouille au coin d'une enveloppe. Lionel, donc, merci pour ce voyage en Terre des nombres, des symboles, des points, des images ; ce fut bien agréable, et je ne le regretterai pas : l'air est pur à ces altitudes et permet de voir loin. J'ai pour ma part choisi d'aller voir ailleurs, mais tout ce que j'ai trouvé ici, tout ce que j'ai compris ici, je l'emporte avec moi.

Je suis reconnaissant à Patrick Bouthemy et à François Fleuret d'avoir accepté d'être rapporteurs de ma thèse. Leurs commentaires apportent un regard neuf sur mes travaux et suggèrent des prolongements originaux pour en densifier la substance et en élargir la portée.

Je remercie tous les membres de mon jury de thèse, Isabelle Bloch, Julie Delon, Joan Glaunès et Ariel Lindner, de me faire l'honneur d'assister à ma soutenance.

Je remercie mes amis codoctorants du laboratoire de Paris Descartes, mes amis corigolants de Normale Sup', mes amis coétudiants du Corps des Mines.

Je remercie enfin toute ma famille, Florence, Gilbert et Romain, et mes amis Antonin et Aude, Anika, Lucile, Emma, Guigui et Irène, Carlos et Céline, Dude et Claire, Matthieu et Typhaine, Axel, Demichou, Samuel, Denis, Alexandre et Caroline, Bruno et Laetitia, Benoît, Thomas, Benjamin, Jacky, Philippe et Keiko, Oussama. Je veux croire que vous m'avez tous tour à tour porté sur vos épaules pour m'aider à poser cette petite pierre tout en haut d'un mur. Un mur qui doit forcément soutenir quelque chose...

...mais quoi ?

— Maël Primet
rue de la Lune, à Paris,
le 25 novembre 2011

Contents

1 Introduction

Les travaux présentés dans cette thèse ont comme lien l'étude du problème du suivi d'objets – problème fondamental en vision par ordinateur, qui se nourrit aujourd'hui de l'accroissement des capacités de calcul et de la multiplication des capteurs vidéo.

Nous demandons en permanence aux ordinateurs d'effectuer des tâches pour lesquelles il est essentiel d'analyser un mouvement : l'interaction avec les machines se fait couramment au travers de tablettes tactiles qui suivent le déplacement simultané de nos doigts sur leur surface, ou grâce à une caméra capable d'interpréter nos gestes. Nos appareils photographiques savent mettre en correspondance des images pour reconstruire un panorama, et certains logiciels spécialisés savent recréer une scène en trois dimensions à partir de quelques prises de vues faites sous des angles différents. Des prototypes de voitures sont déjà capables de se conduire de manière autonome, en observant leur environnement pour prévenir tout danger ; et couplées à un réseau de caméras qui mesurent la fluidité du trafic en temps réel, elles plannifient leur route pour choisir le trajet le plus rapide. En analysant des scènes complexes et en interprétant nos gestes et nos émotions, les machines rendent possible l'indexation automatique de contenu vidéo par des moteurs de recherche, ou la conception de robots de compagnie capables de nous reconnaître et de nous divertir. Le suivi d'objets est devenu par ailleurs un outil incontournable de la science, qui nous permet de découvrir l'existence de nouvelles particules en reconstruisant les trajectoires des électrons dans des accélérateurs, et de comprendre les mécanismes fondamentaux du vivant grâce à l'observation de cultures de cellules ou des échanges de molécules entre les neurones.

Le nombre de scénarios contemporains ou d'un futur proche dans lesquels apparaît le suivi d'objets est gigantesque. L'ubiquité du problème du suivi est facile à comprendre : dès qu'il est question de l'analyse d'une scène sous plusieurs angles de vue, ou à partir de plusieurs clichés successifs, le besoin de reconnaître un même objet dans les différentes images apparaît.

*

La segmentation et le suivi de cellules

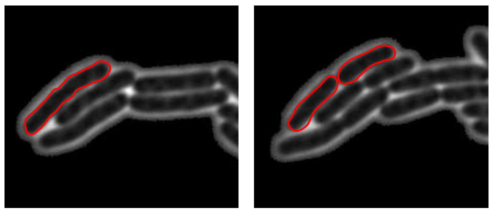
Les applications du suivi d'objets aux images biologiques en particulier ont connu une croissance extraordinaire ces dernières années qui s'explique par le fait que la recherche en biologie se fonde aujourd'hui largement sur l'observation, l'enregistrement et l'analyse systématiques des mécanismes du vivant dans ses moindres détails, et devient ainsi dépendante de la disponibilité de méthodes automatisées capable de traiter des jeux de données de plus en plus grands et de plus en plus complexes. Outre l'intérêt de travailler dans un domaine dont les avancées peuvent avoir un impact considérable sur notre connaissance du vivant et sur l'amélioration de nos vies, les images biologiques présentent une variété et des défis qui en font une cible de choix pour les algorithmes de suivi : que ce soit pour observer les interactions des animaux ou des insectes dans un écosystème, les déformations des organes en mouvement, l'organisation des cellules pendant la morphogénèse, ou les mécanismes de signalisation intracellulaire, des problèmes de suivi d'objets se posent qui possèdent une large palette de formes, de comportements et de densités d'objets, et se présentent sous des modalités variées – données multi-canaux (microscopie et fluorescence) ou tridimensionnelles, par exemple.

Nous nous sommes tout particulièrement intéressés pendant cette thèse, en collaboration avec l'équipe de biologistes du laboratoire TaMaRa de l'INSERM, à la segmentation et au suivi automatiques de cellules dans des séquences d'images pour en extraire les contours précis ainsi que le lignage – c'est-à-dire la relation mère-fille des cellules. Quelques extraits d'une séquence type obtenue grâce à un microscope à contraste de phase sont présentés dans la figure 2 ; les images sont souvent de médiocre qualité, les cellules collées les unes aux autres, et leur mouvement est parfois difficile à prévoir – autant de caractéristiques qui en font un problème complexe à résoudre.

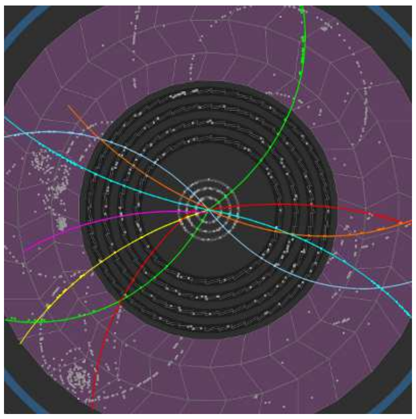
Le nombre de publications sur la segmentation et le suivi de cellules augmente à un rythme effréné, et celles-ci couvrent tout le spectre des méthodes de suivi. Un certain nombre de solutions logicielles sont d'ailleurs déjà disponibles – mais elles sont malheureusement souvent spécifiques à un seul type de cellules et de conditions d'acquisition des images ; le lecteur trouvera une revue très complète des méthodes générales pour le suivi dans Yilmaz, Javed, and Shah, 2006, et une revue récente et détaillée des méthodes dédiées au suivi des cellules et des particules sub-cellulaires dans Meijering *et al.*, 2009 ; le lecteur pourra également consulter Hand *et al.*, 2009 pour sa comparaison de différentes solutions logicielles.

Les méthodes de suivi d'objets comportent deux étapes, qui peuvent suivant les cas être effectuées l'une après l'autre ou être combinées :

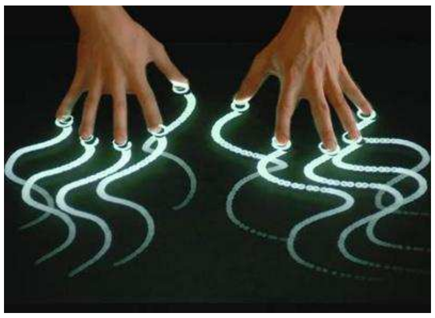
- *la détection* des objets dans les images – sous une forme appropriée à l'application visée (voir figure 3) et à laquelle il est possible d'adjoindre des descripteurs pour aider à identifier les objets (couleur ou texture, par exemple),
- et *le suivi* des objets à proprement parler, consistant à associer entre les images toutes les détections qui correspondent au même objet.


(1) Suivre des cellules qui se divisent

(2) Reconnaître des objets dans deux images pour reconstruire une scène en trois dimensions


(3) Reconstruire les trajectoires des particules dans un accélérateur

(4) Suivre des marqueurs sur un costume pour l'animation de personnages


(5) Suivre plusieurs doigts simultanément pour l'interaction tactile

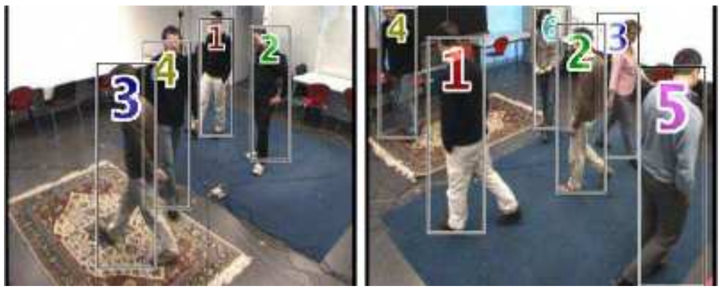

(6) Suivre des individus sous plusieurs angles de vue

Figure 1: **Exemples d'applications du suivi d'objets.** (1) Suivi de cellules se déplaçant, croissant et se divisant dans une colonie, (2) reconstruction tridimensionnelle d'une scène après appariement des objets entre deux images, (3) détection de trajectoires dans un accélérateur de particules (*ATLAS Experiment*, © *2011 CERN*), (4) suivi de marqueurs sur un costume pour l'animation de personnages de cinéma, (5) suivi simultané de doigts sur une surface pour l'interaction tactile, (6) suivi d'individus sous plusieurs angles de vue (*image from* Fleuret et al., *2008*).

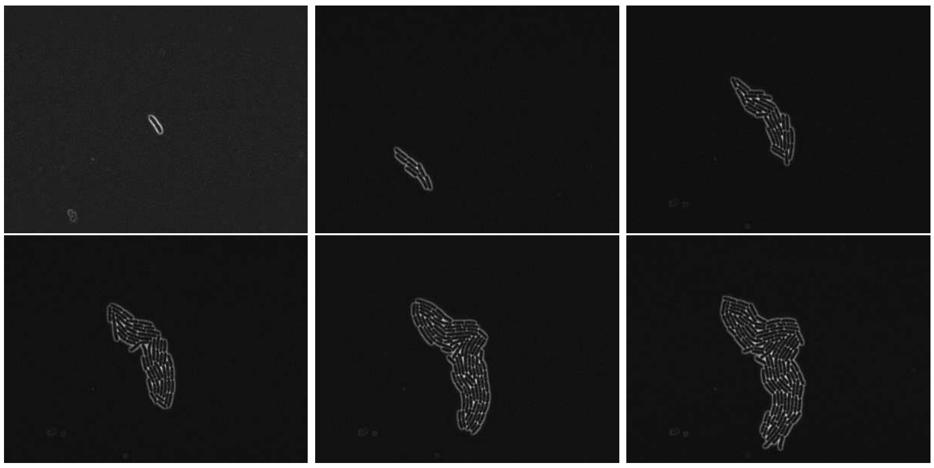
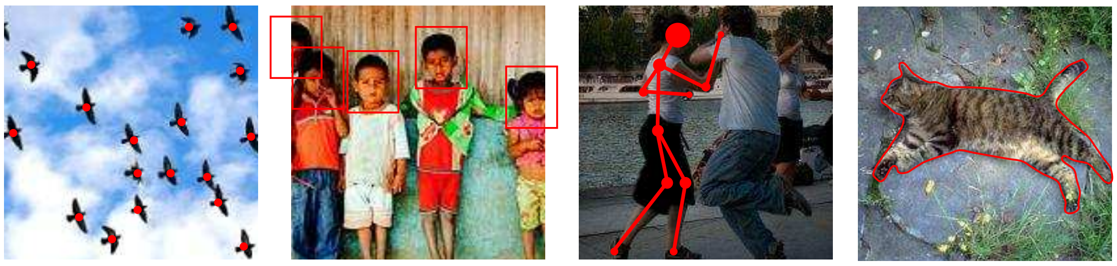


Figure 2: Croissance d'une colonie bactérienne. Quelques images acquises pendant la croissance d'une colonie bactérienne (les intervalles de temps entre les images ci-dessus ne sont pas constants). De nombreux problèmes adviennent lors de l'analyse de ces images et la rendent difficile : les images soient souvent d'une qualité médiocre, les conditions d'illumination changent entre les images et à l'intérieur des images, et les bactéries croissent, se meuvent et se divisent rapidement, tout en restant collées les unes aux autres.



(1) Un oiseau vu de loin est représenté par un point, (2) un visage est localisé par une forme simple, (3) un squelette articulé permet d'analyser un geste, (4) et un contour polygonal fournit une forme précise.

Figure 3: Exemples de représentations d'objets. Selon l'application, la représentation géométrique des objets doit être adaptée : avec des points au barycentre des oiseaux migrateurs, l'ethnologue pourra suivre le trajet des fuyants volatiles ; grâce à de simples formes géométriques comme des rectangles ou des ellipses un appareil photographique localisera les visages de ses sujets et s'assurera de leur sourire avant de les immortaliser ; un squelette articulé permettra à un réalisateur de capturer le mouvement d'un danseur pour une scène de cinéma ; et en suivant ses formes de près grâce à des contours actifs ou des ensembles de niveaux, le robot de compagnie du futur « *Nestor-2000* » pourra caresser le chat de la maison sans lui hérisser le poil. *Miaou. Crédits photographiques : (1) Nate Chute,* Post Register, *(2) Selvin Kurian.*

Les premières méthodes de segmentation et de suivi de cellules dont nous avons connaissance se restreignaient pour des raisons de coût de calcul à des décisions simples et locales, qui fonctionnent bien lorsque les cellules sont isolées et que leur mouvement est de faible amplitude. Dans Liu and Warme, 1977 par exemple, les auteurs partent d'une segmentation statique de chaque image en isolation, et reconstituent la trajectoire des cellules d'une image à l'autre par une méthode locale gloutonne de type plus proche voisin.

Mais, comme c'est le cas de nos séquences, dès que la qualité des images se dégrade, ou que la densité des cellules augmente trop, les ambiguïtés de segmentation et de suivi se multiplient – les cellules se touchent et il devient difficile de les séparer en ne regardant qu'une image, ou leur mouvement est trop rapide pour qu'un simple raisonnement sur la distance permette de retrouver les correspondances entre les détections.

Il devient alors nécessaire de combiner la segmentation et le suivi des cellules, et d'utiliser des méthodes globales pour lever les ambiguïtés : idéalement, nous souhaiterions définir un modèle probabiliste complet de la séquence, incluant à la fois l'apparence des cellules, leur mouvement et leur lignage ; et chercher parmi toutes les explications possibles des images en terme de ce modèle celle qui est la plus probable. Un tel modèle reste malheureusement très théorique, et un grand nombre de simplifications doivent être envisagées pour conserver des temps de calcul raisonnables en pratique.

Les approches récentes essaient donc toutes d'une façon ou d'une autre d'intégrer des contraintes globales pour améliorer la fiabilité de la segmentation et du suivi, tout en restant praticables sur des données réelles. Afin de lever les ambiguïtés de segmentation, Li *et al.*, 2008 proposent de combiner la segmentation et le suivi des cellules. Ils peuvent ainsi suivre une colonie très dense en s'aidant, pour détecter les cellules dans une image, des cellules détectées dans les images précédentes et de leur vitesse. En faisant évoluer simultanément les ensembles de niveau définissant les contours des bactéries qui se touchent grâce à une énergie qui comporte un terme de répulsion, ils évitent de fusionner à tort deux cellules distinctes en un seul contour – ce qui ne manquerait d'arriver si la segmentation des cellules se faisait isolément. Une alternative est proposée dans Padfield, Rittscher, and Roysam, 2008, qui reformule le problème complet comme une segmentation directe dans le volume spatio-temporel tridimensionnel obtenu en empilant les images. Afin de lever les ambiguïtés de suivi, il est fréquent d'introduire des contraintes temporelles ou spatiales globales – dans Delgado-Gonzalo *et al.*, 2010 par exemple, les auteurs forcent les cellules voisines qui se touchent à avoir un mouvement cohérent, et Smith and Lepetit, 2008 intègrent la dépendance entre la forme et le mouvement des objets dans leur modèle pour améliorer la précision du suivi dans le cas de cellules qui s'allongent dans la direction de leur déplacement.

*

Un problème subsiste : chaque séquence à analyser contient plusieurs centaines d'images, une cellule se divisant en moyenne toutes les dix images, ce qui implique qu'elle contient globalement plusieurs dizaines de milliers de traces de bactéries à segmenter et à suivre dans le temps pour analyser les films complètement, et ces opérations doivent être répétées sur des dizaines de séquences, rendant nécessaire l'utilisation d'outils d'analyse *complètement automatisés* et *efficaces*. Ceci est loin d'être un cas isolé dans la recherche, et la tendance actuelle est à l'examen d'énormes ensembles de données pour en extraire les motifs et les régularités qui permettront de conjecturer le fonctionnement des mécanismes biologiques par des analyses statistiques automatisées.

Il est alors crucial de s'interroger sur les ingrédients nécessaires à la construction de méthodes d'analyse complètement automatisées. Celles-ci doivent évidemment pouvoir *s'appliquer sans modification, ni paramétrage excessif*, à de larges jeux de données dont la taille fait qu'ils auront souvent une variabilité interne naturelle. Il faut donc construire un algorithme *robuste aux variations des données*, qui s'appuiera sur des constantes physiques intrinsèques aux données étudiées plutôt que sur des paramètres abstraits définis implicitement par une ou plusieurs étapes algorithmiques de traitement. Par exemple, il semble préférable d'étudier une image au travers des propriétés physiques des objets qu'elle contient (la taille d'une cellule par exemple) et de ses propriétés géométriques intrinsèques (ses ensembles de niveaux), plutôt que de s'attacher à traiter directement les intensités des pixels, trop sensibles au bruit et aux variations de conditions de prise de vue.

Nous pensons qu'un algorithme complètement automatisé doit également être suffisamment *simple à comprendre et à modéliser*, pour qu'il soit possible d'étudier théoriquement ses limites et ses performances, et ainsi connaître *a priori* les cas d'utilisation dans lesquels il est possible de l'appliquer avec succès. Il faut donc préférer *une analyse par des étapes simples et séquentielles* – que l'on peut définir et étudier en isolation – à une cathédrale de processus imbriqués s'exécutant en boucle ou en parallèle, chacun perturbant ou complétant les décisions des autres, pour rendre l'analyse des résultats, l'étude théorique des cas d'utilisation et des performances, ainsi que l'adaptation de l'algorithme pour corriger les erreurs le cas échéant, complètement impraticables.

Choisissant d'appliquer ces principes autant que possible, nous avons proposé un algorithme de suivi simple et robuste, et qui ne s'appuie que sur des paramètres physiques des cellules, donc facilement adaptables à de nouvelles conditions d'acquisition des images. Notre analyse s'appuie sur une notion de *risque*, défini à partir d'un modèle probabiliste de l'évolution des cellules, qui quantifie mathématiquement l'ambiguïté de chaque décision de l'algorithme. Celles-ci sont alors prises de manière globale dans la séquence – et non pas séquentiellement de la première à la dernière image – afin d'éliminer les incertitudes sur les choix en commençant par les décisions les plus évidentes d'abord et ainsi graduellement contraindre les cas pour lesquels il y avait une ambiguïté.

Notre travail – réalisé en collaboration avec Alice Demarez – a donné lieu à publication dans l'*International Symposium on Biological Imaging* (2008) et à une réalisation logicielle (appelée *Céleste* – pour Cell Segmentation and Tracking) qui est utilisée avec succès depuis deux ans dans l'équipe, et qui a permis de diminuer significativement le temps nécessaire au traitement d'un film – de plusieurs jours à quelques heures – et d'améliorer la qualité de la segmentation et du suivi des séquences.

*

Détection de trajectoires dans des nuages de points

Afin d'explorer mathématiquement le problème du suivi d'objets et de pouvoir comprendre plus précisément les questions mises en jeu, nous avons choisi de nous restreindre au problème du suivi d'objets détectés comme de simples points sans attribut dans les images, les détections ayant été réalisées au préalable comme une première étape.

Ce modèle simplifié est bien loin d'être une abstraction futile : le suivi de points apparaît naturellement dans un grand nombre d'applications dès lors que les objets sont d'apparence identique ou qu'ils sont vus de loin, et que seuls leur position et leur mouvement sont importants pour les distinguer – par exemple pour l'analyse de détections radar [Reid, 1979], l'étude d'un écosystème de chauves-souris [Betke *et al.*, 2007], ou celle des mécanismes de régulation et de signalisation dans les cellules grâce à des marqueurs fluorescents [Godinez *et al.*, 2011].

Et autant qu'il puisse être simplifié – car il omet bien des difficultés qui peuvent se présenter dans des cas généraux, comme le suivi de l'évolution de formes complexes par exemple, ou des objets qui se divisent ou fusionnent – le problème du suivi de points contient déjà l'essence de la complexité combinatoire des problèmes de suivi d'objets. Supposons pouvoir définir un coût pour chaque trajectoire envisageable – en fonction de son accélération moyenne ou maximale, par exemple – et supposons même que la détection des objets soit parfaite et que chaque objet soit en correspondance unique avec un point de chaque image : le problème de joindre entre elles les détections pour reconstruire l'ensemble de trajectoires optimisant la somme totale des coûts est en réalité terriblement complexe ! Il est précisément NP-difficile puisqu'il permet de résoudre le problème de l'appariement des triplets, qui est lui-même NP-complet [Karp, 1972].

L'algorithme MHT de Reid, 1979 – pour *Multiple Hypothesis Tracker*, l'un des premiers algorithmes de suivi de points et probablement le plus connu – tente justement de résoudre ce problème exactement en explorant l'ensemble *complet* de toutes les explications simultanées des points détectés en terme de trajectoires, avant d'extraire celle qui optimise un modèle probabiliste du mouvement des objets.

Pour réduire la complexité combinatoire, une approche fréquente consiste à limiter l'exploration de l'arbre de recherche en se limitant à optimiser les trajectoires sur un petit nombre d'images. Le cas le plus célèbre étant celui de l'algorithme Hongrois [Kuhn, 1955] qui résout le problème exactement – et efficacement ! – dans le cas de deux images. Cet approche est la base de nombreux algorithmes, et a été par exemple étendue – avec des approximations, qui deviennent alors nécessaires – à un petit nombre d'images dans Veenman, Reinders, and Backer, 2003b.

Une solution orthogonale consiste à restreindre le problème localement « en espace » (voir figure 4), en ne cherchant plus à optimiser toutes les trajectoires simultanément, mais à les détecter de manière gloutonne les unes après les autres ; en d'autre termes, étant donné un ensemble de points détectés dans des images, on se pose la question de détecter *une* trajectoire qui semble réelle, et on itère ce procédé. Fleuret *et al.*, 2008 proposent une telle approche se basant sur un modèle simple d'apparence et de mouvement, qui est résolue efficacement par un algorithme de programmation dynamique.

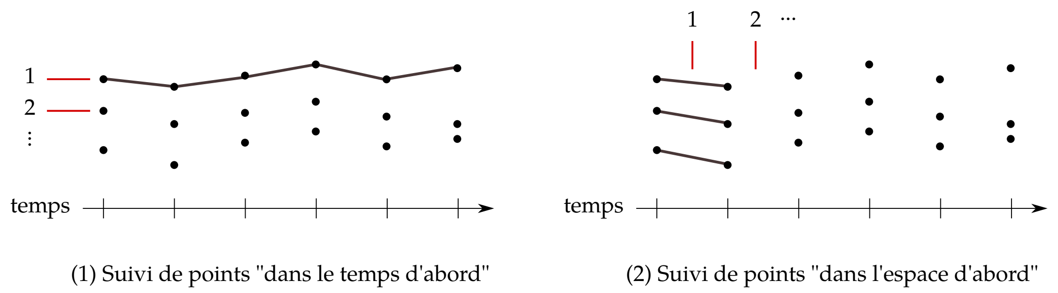

1
2
⋮
temps
(1) Suivi de points "dans le temps d'abord"
1
2 ...
temps
(2) Suivi de points "dans l'espace d'abord"

Figure 4: Illustration des approches en temps d'abord et en espace d'abord. Les images successives sont représentées en une dimension le long de l'axe temporel. (1) L'approche « dans le temps d'abord » extraie les trajectoires les unes après les autres en considérant chaque trajectoire globalement dans le temps, mais en isolation par rapport aux autres, et (2) l'approche « dans l'espace d'abord » considère toutes les trajectoires simultanément, mais les optimise localement dans le temps.

Notons finalement qu'il est fréquent, afin d'optimiser les performances d'un algorithme dans un cas particulier ou de limiter les cas raisonnables d'exploration, d'introduire des modèles de mouvement et d'interaction complexes, des paramètres probabilistes pour modéliser les fausses détections ou les détections manquantes, ou des limites physiques – comme une vitesse ou une accélération maximale par exemple. Ceci est agréable en théorie, car l'algorithme peut être optimisé pour un cas particulier ; mais dans la pratique, la présence de paramètres devient un casse-tête : le réglage est souvent loin d'être intuitif, et doit être souvent modifié entre deux jeux de données, ce qui est contraire aux principes méthodologiques d'analyse automatisée énoncés plus haut.

Nous avons exploré dans cette thèse ces deux approches orthogonales – chercher les trajectoires dans le temps d'abord ou dans l'espace d'abord – en gardant à l'esprit notre démarche de trouver des algorithmes complètement automatisés, et nous chercherons donc à construire des méthodes de suivi sans paramètre.

Pour cela, nous avons construit des algorithmes en utilisant la méthodologie *a contrario*, introduite dans [Desolneux, Moisan, and Morel, 2003] et se basant sur une idée simple : un algorithme de détection ne doit rien détecter dans du bruit. Souvent, ce critère suffit à obtenir un critère de détection et un algorithme efficace associé, dont le seul paramètre a une valeur naturelle et intuitive à choisir.

Nous construisons deux critères *a contrario* pour le suivi de trajectoires « dans le temps d'abord », dans le cas de trajectoires sans trous, et l'autre dans le cas général, qui se traduisent en algorithmes simples et efficaces par programmation dynamique, que nous avons appelé ASTRE (pour « *A-contrario Single TRajectory Extraction* »). Ces algorithmes sont publiés sur le site `http://www.math-info.univ-paris5.fr/~moisan/astre/` avec des données d'exemple, des instructions d'installation et un manuel d'utilisation.

Nous explorons ensuite le problème orthogonal de l'appariement simultané de points dans le cas de deux images ; d'abord en construisant un observateur idéal, WRAP, qui définit une borne optimale pour les algorithmes résolvants ce problème et nous permet d'étudier les performances de quelques approches classiques ; et ensuite grâce à deux nouveaux critères *a contrario*, qui nous permettent de définir un nouvel algorithme de suivi de points sans paramètre.

Par leur simplicité, ces critères mathématiques nous permettent de faire des prédictions sur le comportement des algorithmes, ce qui est une propriété souhaitable des analyses complètement automatisées qui est rarement rencontrée dans la littérature sur le suivi de points à notre connaissance. Nous pouvons par exemple déterminer le nombre d'observations minimales d'un objet pour être capable de le détecter, ce qui dans une application peut avoir une réalité pratique forte.

Enfin, nous verrons que les critères *a contrario* peuvent également agir en tant que filtres, et ainsi améliorer la précision de n'importe quel autre algorithme de suivi, ou même d'automatiser son choix de paramètres.

ORGANISATION DE LA THÈSE

Les chapitres de thèse s'articulent selon le plan suivant :

DANS LE DEUXIÈME CHAPITRE nous présentons le fonctionnement de l'algorithme Céleste pour la segmentation et le suivi de colonies de bactéries, puis nous discutons ses performances et ses limites.

DANS LE TROISIÈME CHAPITRE nous introduisons l'algorithme *a contrario* ASTRE pour la détection de trajectoires sans paramètre dans un nuage de points, et nous analysons son comportement théorique.

DANS LE QUATRIÈME CHAPITRE nous comparons les performances de l'algorithme ASTRE à un algorithme représentant l'état de l'art pour le problème du suivi de points.

DANS LE CINQUIÈME CHAPITRE nous présentons WRAP, l'observateur idéal pour le problème de la correspondance de points, qui nous permet d'apprécier les performances de quelques algorithmes d'affectation classiques, puis nous étudions la possibilité d'utiliser WRAP comme un algorithme à part entière.

DANS LE SIXIÈME CHAPITRE nous définissons un algorithme *a contrario* pour le problème de la correspondance de points, qui nous fournit un algorithme d'affectation sans paramètre, que nous utilisons ensuite pour construire un nouvel algorithme de suivi de points sans paramètre.

DANS L'ANNEXE A nous détaillons deux algorithmes qui résolvent le problème de l'affectation linéaire et de l'affectation goulot, et qui sont utilisés directement ou sous une forme légèrement modifiée pour construire les algorithmes d'affectation présentés dans les chapitres 5 et 6

DANS L'ANNEXE B nous présentons le manuel d'utilisation de la suite logicielle ASTRE pour la détection et le suivi de trajectoires, disponible à l'adresse `http://www.math-info.univ-paris5.fr/~moisan/astre/`.

DANS L'ANNEXE C nous présentons la référence détaillée des programmes de la suite logicielle ASTRE.

CÉLESTE : SEGMENTATION ET SUIVI DE BACTÉRIES

Nous avons cherché à définir une approche de segmentation et de suivi des cellules utilisant les principes méthodologiques que nous avons annoncés plus haut : la robustesse aux variations des données, et la simplicité de l'algorithme,

- nous n'utilisons donc que des constantes physiques et des propriétés géométriques intrinsèques des images pour se passer de paramètres abstraits qu'il faudrait régler sur chaque séquence ou chaque image. Plus précisément, nous faisons les quelques hypothèses simples suivantes : les cellules ont une épaisseur comprise entre deux réels m et M, et leur aire est supérieure à A, et les artefacts d'illumination varient lentement ;
- et nous avons voulu construire un algorithme simple pour que l'on soit capable de comprendre d'où proviennent les erreurs pour pouvoir les corriger le cas échéant : nous choisissons donc une méthode hybride dans laquelle nous segmentons *partiellement* les images de manière statique – les cellules sont divisées en blobs, parfois appelés « superpixels » – avant d'utiliser la redondance temporelle pour simultanément les segmenter en fusionnant les blobs, et les suivre en reconstruisant leur lignage.

Par exemple, le prétraitement des images n'utilise que des constantes sur la taille des cellules, et utilise les lignes de niveaux des images pour renormaliser leurs intensités, ce qui permet ensuite de fixer les paramètres algorithmiques de l'analyse une seule fois pour toutes les images, et de les réutiliser pour toutes les séquences.

Afin de conserver un algorithme simple, l'algorithme fait des choix locaux, en les ordonnant de façon à faire les choix les plus évidents d'abord – l'idée étant que certains choix sont très simples, et d'autres plus complexes ; si l'on commence par faire les choix ne présentant aucun risque, on pourra rapidement contraindre les choix plus ambigus – car on lève graduellement les ambiguïtés en fixant définitivement les choix pour les cellules voisines – qui deviendront alors eux-mêmes des choix évidents.

Pour modéliser cette notion de « choix évident », nous introduisons le risque associé à une transition de la cellule A de l'image k vers la cellule B de l'image $k+1$, défini par

$$\rho_{A\to B} = \max_{X\neq B} \frac{\pi_{A\to X}}{\pi_{A\to B}},$$

où $\pi_{A\to B}$ est la probabilité de la transition, le maximum étant pris sur tous les successeurs potentiels X de A (B excepté), et le risque étant nul par convention s'il n'y a qu'un successeur potentiel. Intuitivement, le risque est faible lorsqu'un choix n'a pas d'alternative crédible, et est élevé lorsqu'il y a un doute sur le choix d'un successeur pour la cellule.

La probabilité d'une transition est obtenue à partir d'un modèle probabiliste simple de l'évolution des cellules dont les paramètres ont été appris sur quelques séquences segmentées manuellement.

L'algorithme Céleste permet de segmenter sans intervention humaine jusqu'à 7 ou 8 générations de cellules. Nous discutons quelques erreurs de segmentations et de suivis classiques et une modification de l'algorithme pour les corriger.

ASTRE : DÉTECTION DE TRAJECTOIRES SANS PARAMÈTRE

La définition d'algorithmes complètement automatisés pose la question de la construction d'algorithmes sans paramètre pour l'analyse du mouvement. Ces algorithmes doivent être capables de définir les limites de l'analyse qui sont intrinsèques aux données en les observant directement, et non en demandant à l'utilisateur de fournir des seuils et autres paramètres.

Nous nous plaçons dans le formalisme des méthodes *a contrario* qui consistent à construire un modèle probabiliste naïf d'une séquence dans laquelle *il n'y a aucune structure* – une séquence constituée simplement de points aléatoires – et de se poser la question : comment construire un algorithme de détection de trajectoires *intéressant* qui ne détecte rien dans cette séquence ?

Nous construisons ainsi deux critères pour la détection de trajectoires, l'un plus simple dans le cas où il n'y a pas de détection manquante dans les données, et l'autre un peu plus coûteux en mémoire et en temps de calcul, dans le cas général.

Ces critères mesurent pour chaque trajectoire la probabilité qu'elle apparaisse par chance *dans une séquence de bruit*, et sont intéressants à plusieurs titres. Les critères combinent en un seul terme le nombre de détections de chaque image, la longueur de la trajectoire et son accélération maximale et s'adaptent ainsi automatiquement à chaque condition d'acquisition des images et à chaque trajectoire individuelle.

En détectant les trajectoires dont le critère associé est inférieur à un seuil donné – le nombre de fausses détections moyen ε que l'on autorise – on obtient un algorithme efficace de détection se basant sur de la programmation dynamique. Bien que le seuil ε sur le critère maximal autorisé d'une trajectoire détecté soit formellement un paramètre, il s'avère en pratique – comment souvent dans les méthodes *a contrario* – qu'il est bien conditionné et robuste, et qu'il possède une valeur naturelle $\varepsilon = 1$ fonctionnant presque optimalement dans la plupart des cas ; nous pouvons donc considérer que les algorithmes de détection de trajectoires obtenus sont sans paramètre.

Nous faisons l'étude théorique du comportement de l'algorithme pour en déduire des limites à ses performances, et donc des conditions requises pour que l'algorithme puisse détecter des trajectoires dans les cas pratiques.

Nous le comparons ensuite à l'état de l'art, pour montrer que notre algorithme est particulièrement robuste lorsqu'on l'utilise sur des séquences contenant un grand nombre de points de bruit, et qu'il se compare très favorablement à l'état de l'art dans le cas général, alors même qu'il est sans paramètre.

Enfin, nous examinons dans quelle mesure ces critères peuvent servir de filtre sur le résultat d'un algorithme de détection de trajectoires quelconque pour éliminer les fausses détections et améliorer la précision de leurs résultats.

WRAP : OPTIMUM POUR LA CORRESPONDANCE DE POINTS

De nombreux algorithmes en vision par ordinateur cherchent à résoudre le problème de la correspondance de points, aussi appelé problème de l'affectation, qui consiste à mettre en relation des détections d'objets qui se correspondent dans deux images légèrement différentes – prises à un instant différent ou sous un angle différent par exemple. Ils utilisent généralement pour cela l'algorithme Hongrois, ou une variante modifiée pour prendre en compte certains caractères pratiques de chaque application – la présence de points de bruit ou de détections manquantes par exemple. Dans le cas d'objets représentés par des points, et sans informations complémentaires sur leur apparence (comme une texture ou une couleur), la plupart des méthodes d'affectation semblent équivalentes. Les algorithmes gloutons les plus simples réussissent aussi bien que les algorithmes globaux plus complexes sur des problèmes simples, et ces derniers ont des résultats aussi peu robustes que les premiers dans les cas difficiles.

Nous avons cherché à définir, sur un modèle simple et assez général de génération des données, un observateur idéal – WRAP, pour *Weighted Recall And Precision* – qui fournit la limite théorique aux performances des algorithmes de correspondance de points, afin de pouvoir les étudier plus précisément. Ce critère fonctionne en détectant les affectations de manière à maximiser l'espérance d'une combinaison linéaire entre le rappel et la précision des résultats, $\mathbb{E}[r + \lambda p]$ pour chaque $\lambda > 0$.

Après avoir présenté quelques algorithmes classiques pour la mise en correspondance de points, notamment l'algorithme par maximum de vraisemblance, par maximum a posteriori et l'algorithme des plus proches voisins, nous étudions le comportement de ces algorithmes et nous le comparons à celui du critère optimal WRAP pour mettre en évidence leur faiblesse principale : les algorithmes classiques ne prennent souvent pas en compte la présence d'ambiguïtés. Nous proposons une modification simple de l'algorithme des plus proches voisins pour améliorer ses performances.

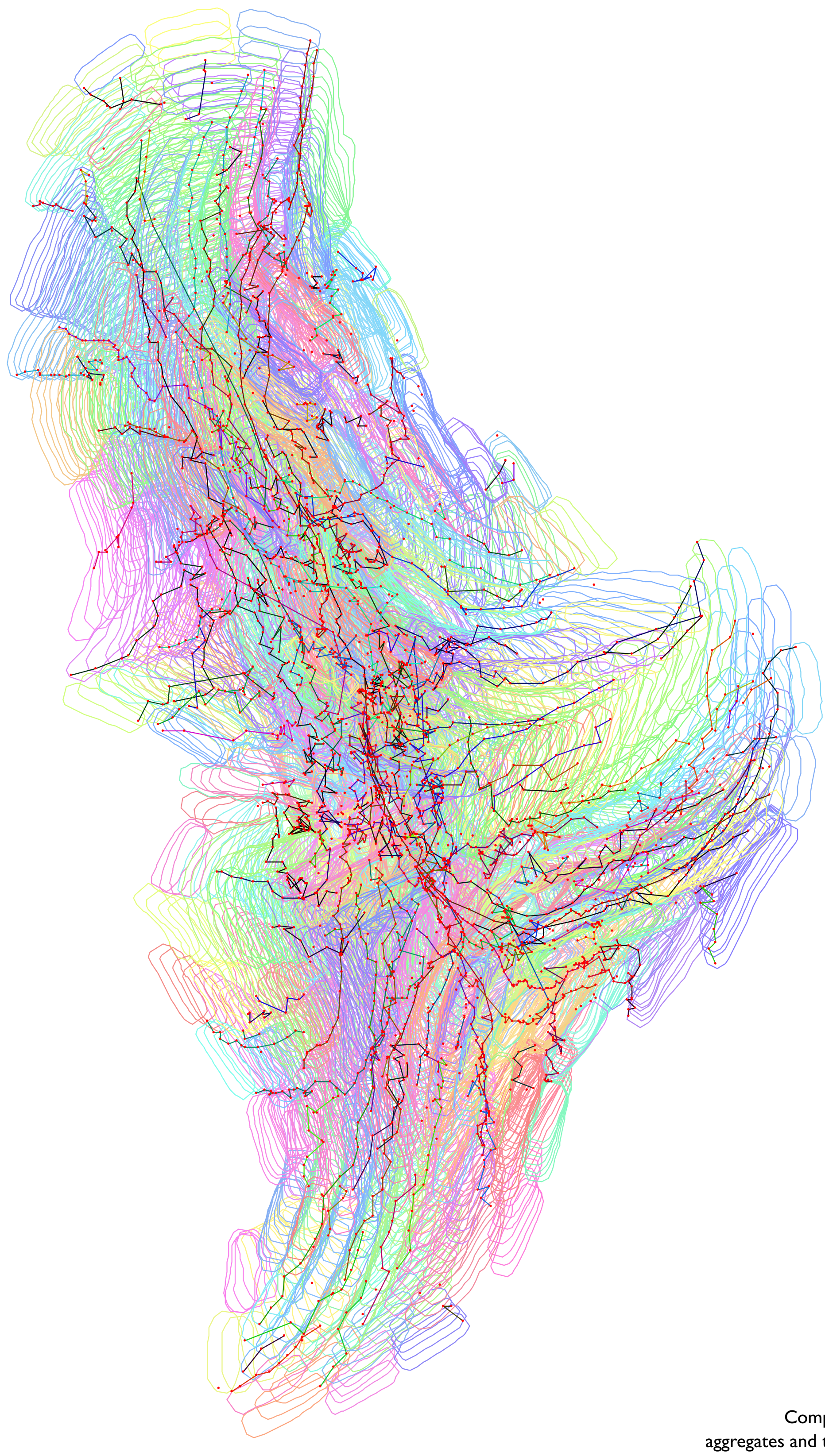
Nous examinons ensuite la possibilité d'utiliser le critère optimal WRAP comme un algorithme (que l'on peut calculer efficacement par une méthode MCMC) et comparons ses performances à l'algorithme par maximum de vraisemblance.

CORRESPONDANCE DE POINTS SANS PARAMÈTRE

En nous replaçant dans le cadre *a contrario*, nous définissons trois algorithmes de détection d'affectations en présence de points manquants ou erronés : un algorithme glouton qui considère chaque couple de points, et deux algorithmes qui considèrent les affectations globalement, en optimisant respectivement leur coût maximal et leur coût total.

Ces critères *a contrario* permettent à nouveau d'effectuer des prédictions théoriques sur le comportement des algorithmes, et peuvent être utilisés avec un paramètre naturel, $\varepsilon = 1$, qui les rend essentiellement sans paramètre.

Nous comparons ces algorithmes sans paramètre avec quelques algorithmes classiques pour le problème de l'affectation, puis nous appliquons ce problème à celui de la détection de trajectoires, en comparant un algorithme de suivi de points très simple qui utilise l'algorithme Hongrois avec celui obtenu en remplaçant simplement cet algorithme par notre version sans paramètre. Nous montrons que ce nouvel algorithme de suivi sans paramètre se compare favorablement à l'algorithme initial pour lequel un paramètre optimal est choisi.



Composition of cells,
aggregates and their trajectories

2 Céleste

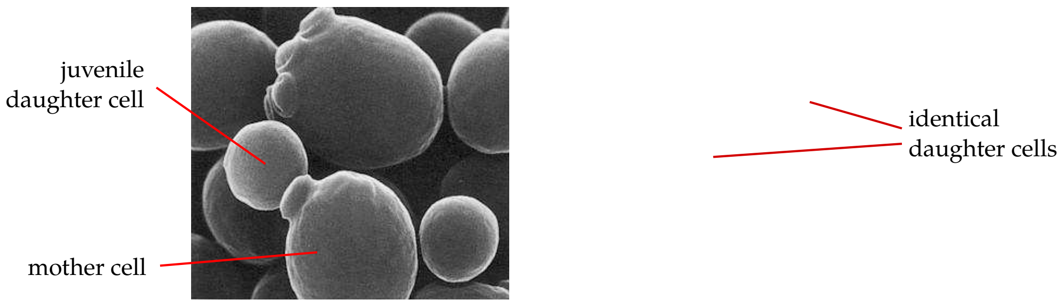
CONTENTS

2.1 INTRODUCTION

VISUALIZING AND QUANTIFYING the mechanisms of biology, at the level of an organism, of a cell or inside a cell, is a current challenge that may bring a new understanding to the functioning of life and lead to major discoveries in numerous scientific fields. Recent technological advances in imaging devices [Stephens and Allan, 2003] and memory storage capacities [Hilbert and López, 2011; Science, 2011] have enabled scientists to observe those mechanisms in detail and collect unprecedented amounts of data on them. As recently discussed in Gough and Yaffe, 2011, there is a need for software dedicated to the analysis of cell images to leverage the increasing computational power at the researchers disposal to process these data automatically, and gain a novel insight on the behavior of the living. The Computer Vision literature on this subject is now flourishing (see Meijering *et al.*, 2009, Bhaskar and Singh, 2007 and Zimmer *et al.*, 2006 for recent and detailed reviews) and many software solutions are readily available (see Hand *et al.*, 2009 for a comparison of several solutions), although they are often tuned for specific families of cells and imaging conditions and an ad-hoc combination of algorithms must still often be crafted manually for each particular application. Such an in-house solution adapted to phase-contrast microscopy images had been developed in the TaMaRa's lab, the biology research unit U1001 from INSERM and Université Paris Descartes, to study the aging mechanism of the bacteria *Escherichia coli*. Their analysis tools had some limitations, and our goal was to provide them with an improved image analysis solution to ease their future research on *E. coli*.

2.1.1 Aging of *Escherichia coli* bacteria

The aging of *E. coli* was until recently still subject to question [Stewart *et al.*, 2005]. Indeed, although many cells like the yeast *Saccharomyces cerevisiae* divide asymmetrically, producing a smaller offspring cell which goes through a juvenile phase, and then differentiates and ages visibly, some other cells, like *E. coli*, grow and divide symmetrically, providing with two apparently identical offspring that do not undergo such a juvenile phase (see Figure 5). This raises the question of whether these bacteria age, or endlessly rejuvenate?



(1) *Saccharomyces cerevisiae* (2) *Escherichia coli*

Figure 5: When a cell divides asymmetrically like the yeast *Saccharomyces cerevisiae* (1), with a daughter cell undergoing a juvenile phase, the aging process is obvious, but when the cell divides symmetrically and yields two seemingly identical daughter cells like the bacteria *Escherichia coli* (2), the question of whether the cells age or rejuvenate naturally arises.

In order to find evidence for the existence of an aging process, the researchers have recorded the growth of bacterial colonies (see Figure 6) and used their own automated image processing software to segment and track the bacteria and extract meaningful characteristics supporting their hypothesis – growth speed or reproduction rates for instance. They where thusly able to prove the existence of an aging mechanism for *E. coli* cells. More precisely, a dividing cell splits in two new cells, both having an "old pole" and a "new pole" – one of the extremities of the daughter cell was also an extremity of the mother cell, and the other was created during the mitosis. We can thus attribute an "age" to each pole: the number of cells in the lineage that it has belonged to (see Figure 7).

It is known that chemical compounds tend to accumulate in the poles as the cells divide, and therefore, old poles tend to contain "cell garbage". The researchers at TaMaRa's lab have found that cells having an old pole exhibit an aging behavior – growing and reproducing less, while cells having new poles seem to have been "rejuvenated".

In order to complete this research, it was crucially important to be able to observe each cell individually with great precision. The movies of colony growth span several hundreds of

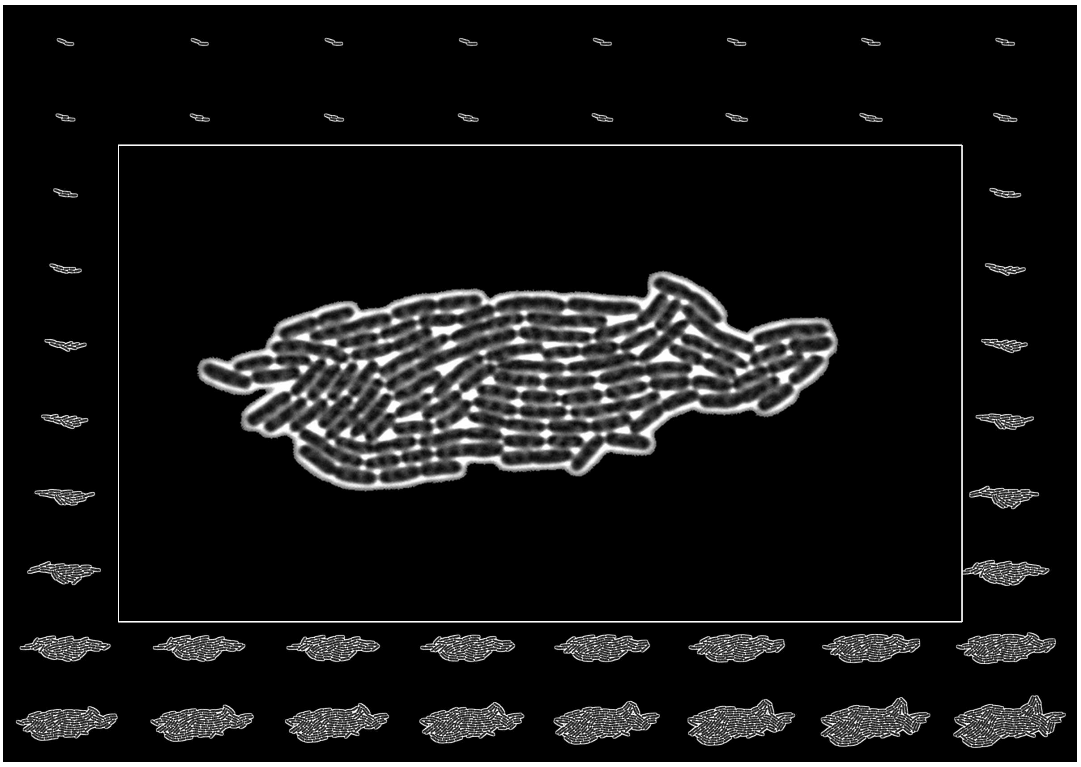


Figure 6: Sample preprocessed images taken from a bacterial colony growth movie, and an inset containing a zoomed image. The initial bacteria regularly divides and grow into a colony of several hundreds of cells, thus resulting in an image sequence consisting of dozens of thousands of cell traces to analyze. The cells are packed together and the border between two touching bacteria is not always discernible, resulting in a challenging problem for computer vision algorithms.

images, and a cell would typically divide every 10 frames, resulting in tens of thousands of bacteria to segment and track in all the combined images, and this had to be repeated for several dozens of movies, in order to obtain statistically significant measurements of differences in cell growth and reproduction rate, for instance. This underlines the importance of automated image analysis tools.

2.1.2 Automated cell movie analysis

The analysis of image sequences of cells is often rendered complex by numerous factors: the poor quality of the images, the density of the cells in the image, the variability of the cell appearance and its inherent combinatorial nature due to cell divisions, to name a few.

Although some image analysis solutions exist, the sheer diversity of cell shapes, imaging conditions and particular application requirements often require to craft a specific solution for each application. For instance, some applications require a real-time analysis of the cells to influence on the growth substrate, and some applications require only a rough idea of the number and position of the cells, while other require a precise outline for each cell. Some applications only request the location and shape of the cells in one isolated image, while others

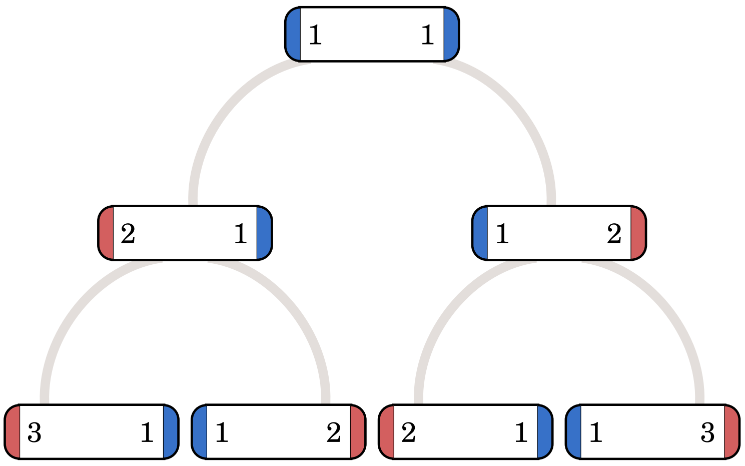


Figure 7: Definition of the age of a bacteria. Two successive divisions of a bacteria. Each division propagates an old pole (red) to the daughter cell, and create a new pole (blue). The poles can thus be attributed an age, and researchers have shown that the behavior of the cells is related to the age of their poles – cells having old poles exhibit an aging behavior, while cells with young poles rejuvenate.

also want to track the cells through time and reconstruct their complete lineage. An algorithm capable of correctly segmenting and tracking any type of cell is today not a reality. Most of the algorithms have been designed for a specific family of cells, in order to use the particularities of each cell, whether a prior on its shape helps decide its contours in an image, or the fact that it can divide help us disambiguate a tracking decision.

There are two main steps in the analysis of biological image sequences:

1. *segmenting the images* to define the spatial boundaries representing the objects to measure in each image,
2. and *tracking the objects* to follow the evolution of the objects through time and observe the variation of their measurements.

Note that in some applications requiring only a rough idea of the position and number of cells, the segmentation step is replaced by a simple detection of the cells (see for instance Debeir *et al.*, 2005).

Before segmenting images, one will often want to correct for uneven shading of the images (see for instance Li and Kanade, 2009; Tomažević, Likar, and Pernuš, 2002) and normalize them in order to obtain homogeneous intensities inside images and across the sequence.

Segmentation

The simplest segmentation algorithm is the intensity thresholding, which separates a brighter foreground from a darker background (or the opposite). This crude segmentation approach is popular because of its simplicity, but will fail in many situations where the cells are to close from each other, or their borders are too dim for instance, leading to over-segmentation (a cell is broken in several parts) or under-segmentation (two cells are merged in one). The Otsu thresholding algorithm [Otsu, 1979] uses a local thresholding value and might be used when the lighting conditions vary inside the images.

A widespread and more robust approach is the *watershed algorithm* [Vincent and Soille, 1991] that consists in segmenting the shapes having a bright contour by seeing the intensities of the pixels as the height levels of a map – the shapes are thus "catchment basins" in this map, and flooding the basins with water until two adjacent basins join, implicitly defining a separating "watershed line". When the shapes do not have a bright contour but are only contrasting with the background, the watershed algorithm might be applied to the gradient image. Usually, a direct application of the watershed algorithm will lead to an over-segmentation of the cells since images are noisy and thus many basins are present inside the cells. However, by defining an isolated marker inside each cell and flooding the shapes only at the location of those markers, one achieves reasonable segmentation results (see Beucher and Meyer, 1992).

Deformable models like active contours [Blake and Isard, 1998; Kass, Witkin, and Terzopoulos, 1988] (or "snakes") and level-sets also stir considerable interest (see for instance Shen *et al.*, 2006). They work by explicitly or implicitly defining a contour that they evolve using an energy functional that depends both on the image and on a prior on the shape. These approaches might be prone to under-segmentation, and might thus be problematic when cells are packed together.

Most of the time, the segmentation algorithms are tuned to take advantage of the particularities of the type of cells. For instance, a cell might have a minimum area, or an elongated shape, etc. (see Fernàndez, Kunt, and Zrÿd, 1995; Xie, Khan, and Shah, 2008; Zhang *et al.*, 2006).

Tracking

Once the shapes of the cells in each frame have been found, one usually wants to track each cell through the sequence, and reconstruct its complete track, or even lineage if it happens that the cells divide or merge.

Ideally, we would observe each possible complete lineage and choose the one that best describes the cells in each frame. Most of the time, however, the combinatorial complexity due to the large number of frames and cells prevents us from building interesting functionals that are to be optimized on the global sequence, and we have to make local tracking decisions.

There exists a wealth of tracking algorithms, that can usually be adapted to the problem at hand to cope with possible divisions, merges, appearances or disappearances of the cells, and that can use the information collected on the cells profitably to track them: their position obviously, but also their shape and appearance if they do not change dramatically.

The simplest algorithms like Liu and Warme, 1977 will statically segment each image in isolation, and reconstruct the trajectories by greedily linking each cell to its nearest neighbor in the following image, where "nearest" can be understood as a spatial and photometric proximity. This approach is best suited when the density and the motion of the cell is limited.

However, in some situations, the segmentation step cannot be achieved in a reliable way on individual images (even the human eye has some difficulties), and even if it could, there might be a complex motion resulting in many tracking ambiguities (see Figure 8).

Recent approaches thus all try in a way or another to include global constraints to ameliorate the robustness of the segmentation and the tracking.

In order to alleviate segmentation ambiguities, Li *et al.*, 2008 propose to combine the segmentation and the tracking of the cells. They are thusly able to segment and track a dense cell colony by using, to detect cells in an image, the cells detected in the previous images and their speed. By simultaneously evolving the level sets defining the neighboring cell contours with an energy containing a repulsion term, they avoid merging together two distinct cells in one contour. An alternative is proposed in Padfield, Rittscher, and Roysam, 2008, where the

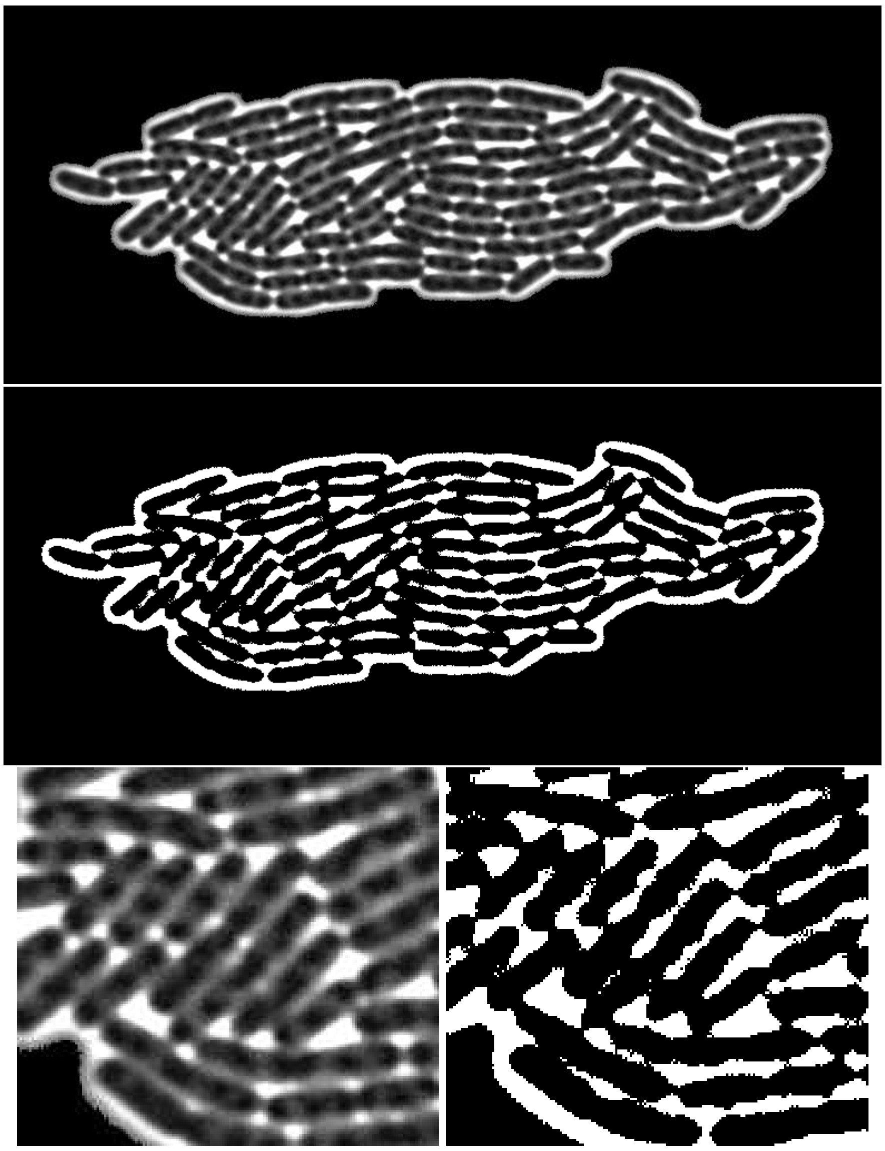


Figure 8: Segmentation ambiguities. The general borders around the cells are easy to detect, but deciding on the exact frontier between two cells that are stuck together by observing one image only is often challenging.

segmentation and tracking problem is reformulated as a direct segmentation problem in the spatio-temporal volume obtained by stacking the images in a pile.

To resolve tracking ambiguities, it is common to introduce global temporal or spatial constraints – for instance, Delgado-Gonzalo *et al.*, 2010 force neighboring cells to move coherently in the same direction, and Smith and Lepetit, 2008 integrates the dependency between the cells shapes and their motion to ameliorate the tracking performances when the cells elongates in the direction of the motion.

One-step and two-step segmentation and tracking

When possible, analyzing image sequences in a two-steps process (segmentation, then tracking) has two main advantages: first, it dramatically reduces the combinatorial complexity of tracking, since it works on the object space (the segmented objects) instead of the pixel space. Second, it splits the overall problem in two, and thus produces intermediate results than can be checked and used to improve each step separately.

The recent approaches discussed above (for instance Li *et al.*, 2008) build a model of the cell motion in order to improve the segmentation performances. Such one-step approaches are efficient to resolve segmentation ambiguities when they are localized, or when an accurate motion model can be built. In the case of a bacterial colony, cells are constantly in contact, steadily grow and divide, and push each other, which results in unpredictable rotations and motion. This makes one-step approaches difficult to use, because they do not yield any intermediate representation between image pixels and the final objects (the cells), so that in general there is no easy way to understand what is wrong (and which parts of the algorithm have to be modified) when mistakes are present in the final lineage.

The Bacterial Home Vision solution

As an illustration of segmentation and tracking algorithms, we briefly describe the in-house software solution *Bacterial Home Vision* (BHV) developed at TaMaRa's lab. In this case, we want an algorithm specialized for *E. coli*, that has no running-time constraints since it does not control a real-time system, but should output very precise cell outlines and record the complete lineage of the cells throughout the sequence.

BHV segments all the images in isolation in a first step, then tracks the bacteria across the sequence in a separate second step. The segmentation consists in detecting local maxima in the images to detect the cell borders (the bacteria are surrounded by a bright halo) and using morphological operations to clean them up. This usually leads to many segmentation errors that had to be corrected manually by the researchers using an image editing software.

The tracking is done sequentially from the first image, and each cell from the current frame is paired to one or two (in case of division). Each cell from the current image is moved to its predicted location (using a simple motion model) in the next image, and is associated to one or two cells (in case of division) depending on the area on which the cell overlap.

2.1.3 Céleste

The BHV automated segmentation suite that the researchers developed for this study revealed very time-consuming to use, sometimes a movie took the researchers up to one week to completely segment and track.

We here describe an image analysis software called Céleste (an homonym of CellST, for Cell Segmentation and Tracking) that was designed during the preparation of this thesis to assist them in further research. Céleste tries to combine the advantages of two-steps approaches (so that intermediate results can be checked) with the simultaneous segmentation and tracking of cells, to disambiguate segmentation ambiguities using the temporal redundancy. The first step is a reliable and efficient over-segmentation process (described in Section 2.2) that produces an intermediate low-dimension image representation, a collection of small shapes called *blobs*. The second step is a segmentation-and-tracking iterative algorithm (section 2.3), in which the segmentation is performed at the blob level (a cell is a union of connected blobs), and the tracking decisions are ordered with a notion of risk we introduce, which permits to improve the robustness of the process by a factor typically equal to 10 (see Figure 9 for an illustration of the steps). Both steps use no intensity-based thresholds or parameters, but geometric cell properties (minimal and maximal width, minimal area, etc.) and simple parametric cell motion priors. We show experiments made on real data and comment the results obtained.

2.2 IMAGE SIMPLIFICATION BY OVER-SEGMENTATION

Since we would like to analyze the image sequence with a high degree of reliability, we have to be very careful in the first step of data processing, as an error at this step will inevitably make the whole algorithm fail. Hence, rather than trying to immediately perform a complete segmentation of each image, we compute an "over-segmentation", that is, a partition of the image domain into a background domain and small regions called *blobs*, with the properties that any cell of the image is a union of connected blobs, and that any blob belong to exactly one cell. Such an over-segmentation seems to achieve a good compromise, because it manages to simplify the images into a small number of "objects" (the blobs) without having to solve ambiguous decisions.

2.2.1 Image renormalization

Data acquisition protocol

For practical reasons, several image sequences are acquired at once, and a moving plate places each colony in turn in front of the microscope camera (phase contrast microscopy). The duration between two consecutive takes of a colony is not constant: as the number of bacteria becomes large, their speed also increases, and a faster acquisition time is required to disambiguate the cell associations. In the researchers experiments, an image was taken every 4 minutes at the beginning of the sequence, and every 2 minutes near the end. We assume that the acquisition frequency is always such that a cell cannot divide more than once between two successive frames.

Images are quantized with a spatial resolution of about $0.064\,\mu m$ per pixel, and gray levels are encoded on 16 bits (sample raw images from a typical movie are shown in Figure 10).

Image denoising

The first step of our image processing is naturally the denoising. In order to obtain sharper borders for the cells, we used a simple total-variation denoising [Rudin and Osher, 1994; Rudin, Osher, and Fatemi, 1992] with adequately chosen parameters (see Figure 11). This proved sufficient for our application, but denoising methods exist that leverage the space-time redudancy of the image sequences to obtain even better performances [Boulanger *et al.*, 2010].

Intensity renormalization

A natural idea to define the blobs is to grow seeds obtained after some gray-level thresholding (the darkest regions of the image are the inside of the cells). Since image illumination may slowly vary inside one image, or may change across time or between experiments, we first apply a gray-level renormalization to obtain homogeneous intensities which is based only on geometrical and physical assumptions:

(A1) There is a minimum width m of a cell

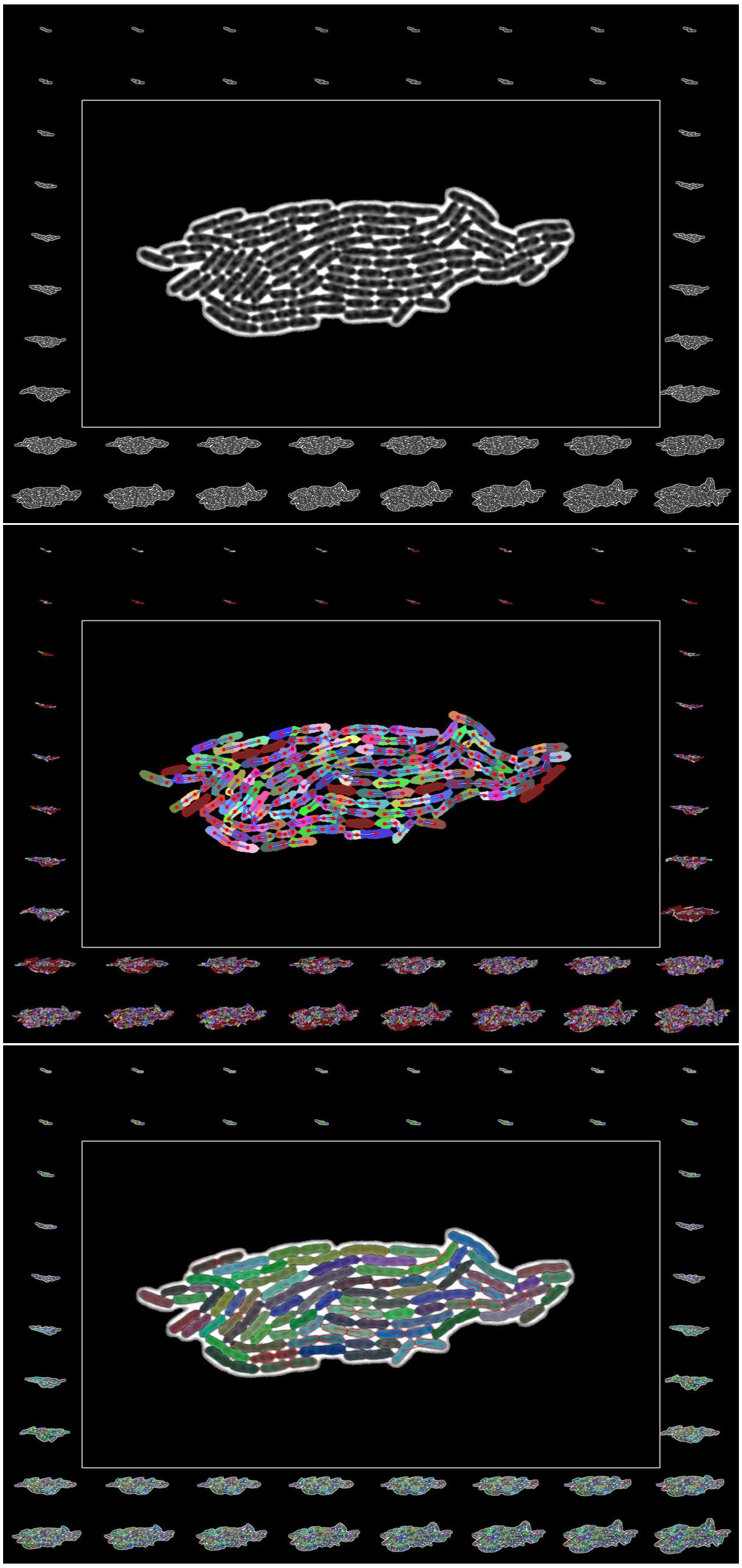


Figure 9: **Overview of the Céleste algorithm.** The images are normalized and cleaned up, then they are simplified by an over-segmentation resulting in blobs connected by a graph (blobs in red have been statically detected as cells), and finally, a dynamic tracking and segmentation phase yields the correct cells and lineage by merging adjacent blobs that belong to the same cell. The links in red in the last image correspond to a cell that just divided.

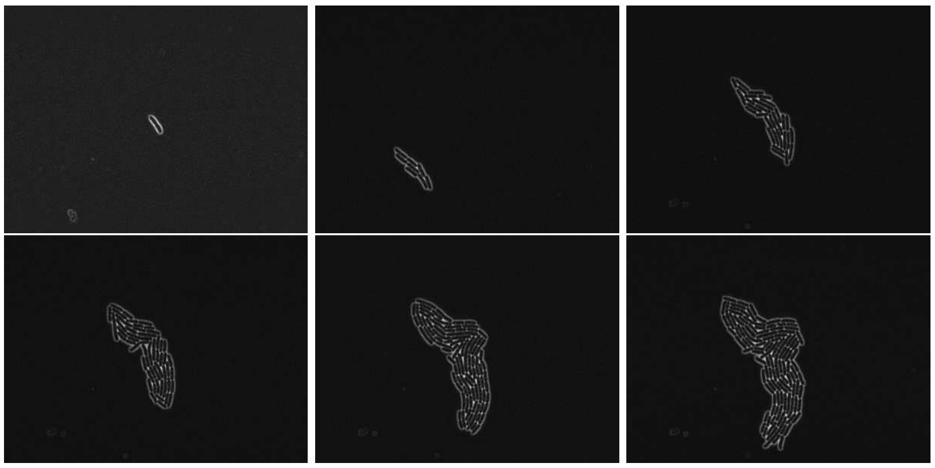


Figure 10: Sample raw images from a bacterial colony growth movie (the elapsed time between the images shown above is not constant). The initial bacteria regularly divides and grow into a colony of several hundreds of cells, resulting in an image sequence consisting of dozens of thousands of cell traces to analyze.

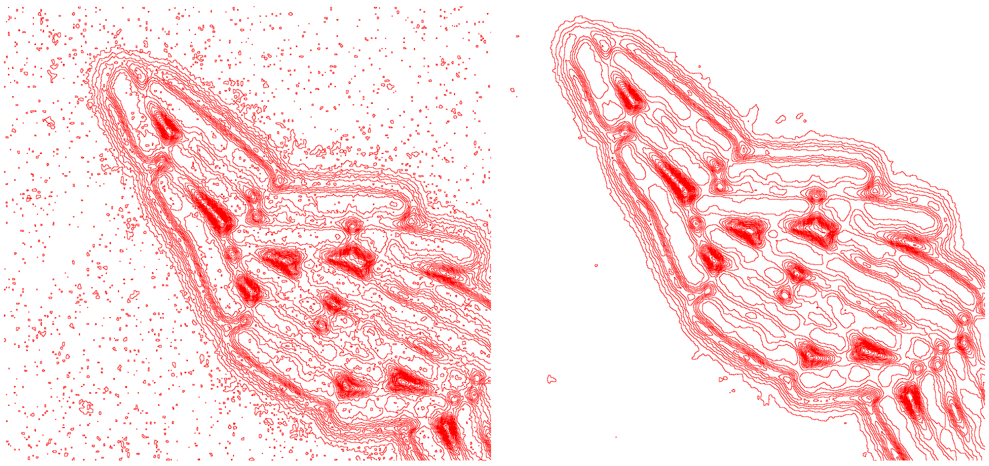


Figure 11: Detail of the level lines of a cell colony (a line is shown at every 50 gray levels) before and after denoising. The cell borders become smoother and the noise is significantly decreased.

(A2) There is a maximum width M of a cell

(A3) The illumination artifacts are slowly-varying.

The idea is to sample the gray levels at various places near the cells in the picture, and estimate the local gray-level thresholds that can be used to define seeds in the cells (low threshold) and extra-cell space (high threshold). Figure 12 illustrates this process by showing the seeds, the high threshold, and renormalization of a sample sequence image.

We first use a Fast Level-Set Transform [Monasse and Guichard, 2000] to obtain what we call *seeds*, that is, connected regions defined by the image lower level-sets with area less than $\pi m^2/4$

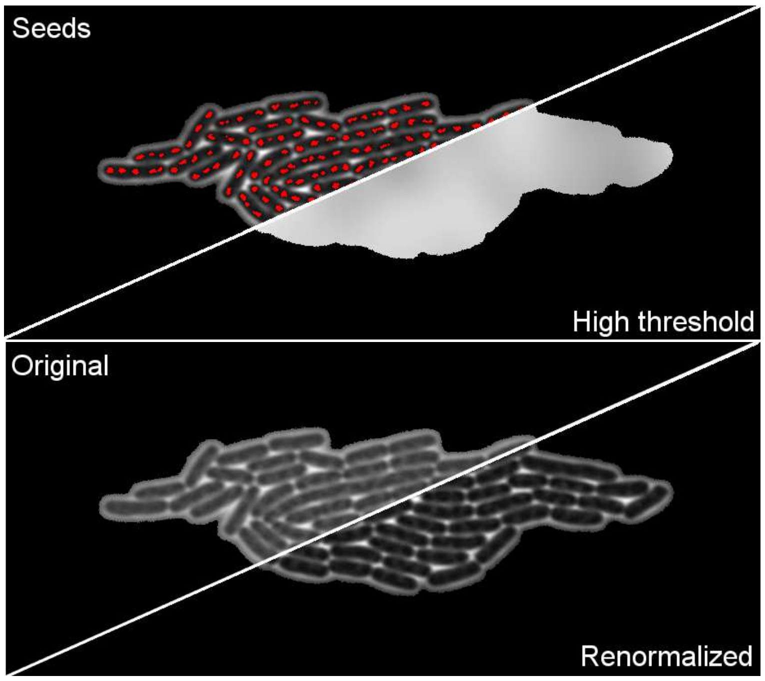


Figure 12: **Grey-level renormalization.** **(Above)** The extracted seeds (in red) and the high-threshold image computed from those seeds that reflect the intensity variations across the image, and **(Below)** the original denoised image, and its renormalization after applying the point-wise affine transform (2.1).

(A1) (discarding the very small lower-level sets as noise). We then compute the maximum of u (the original image) on each seed, and extrapolate these values on the whole image domain by using a Gaussian convolution from known values (the convolution parameters have been chosen once, and give adequate results on all the sequences we have encountered). This process yields a smooth (A3) "low threshold image" ν_-.

To obtain the "high threshold image" in a similar way, we compute for each point x of each seed, the gray value

$$\rho(x) = \min_{y;x\in B(y,M)} \max_{z\in B(y,M)} u(z),$$

where $B(y, M)$ denotes the disc of center y and radius M. From (A2), we deduce that if $x \in B(y, M)$, then the disc $B(y, M)$ necessarily contains a pixel outside the cell, so that the max over z is an upper bound for the optimal high threshold, that we can minimize with respect to y. We then compute the maximum of $\rho(x)$ on each seed, and extrapolate these values on the whole image domain in the same way as before, yielding a "high threshold image" ν_+.

We finally normalize the original image by a point-wise affine transform

$$w(x) = \frac{u(x) - \nu_-(x)}{\nu_+(x) - \nu_-(x)}. \tag{2.1}$$

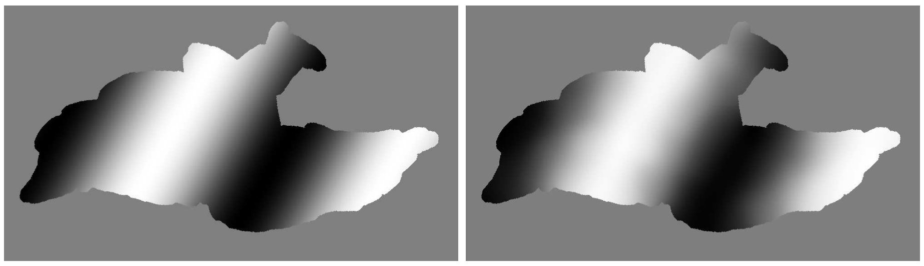


Figure 13: **Grey-level renormalization.** (**Left**) We added a sinusoidal intensity perturbation to an image (amplitude of 150 gray levels, on an image whose amplitude range between the cell inside and outside was approximately 600 gray levels), and (**Right**) the difference between the high-thresholds obtained using the renormalization process on the original image and the perturbed image. We see that we are able to recover approximately the perturbation wave, giving us confidence that our renormalization process is able to eliminate slow-varying intensity changes.

Image recentering

A simple thresholding enables us to separate the colony from the background, and we proceed to recenter the colony on its center of mass. This helps remove the remaining position jitter caused by the fact that when the camera acquires several colonies at a time, it never comes back precisely to the same position (see Figure 14).

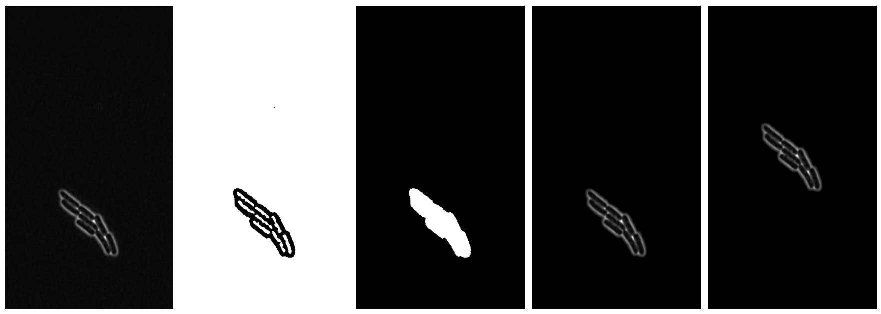


Figure 14: **Image recentering.** Thresholding the denoised and renormalized image yields a connected component, from which we extract a mask representing the cell colony. The background of the image is then cleaned to remove noise, and the colony is centered in the image to attenuate the camera motion jitter.

2.2.2 Non-uniform dilation

Now that the seeds have been found and the image renormalized, we grow the seeds into *blobs* by using a concurrent dilation (ie.. blobs should not penetrate each other) that bears some resemblance with the viscous watershed transform of Vachier and Meyer, 2005.

Rather than seeing the image as a "landscape" whose catchment basins are filled with water as in the watershed transform, we view it as an environment of non-uniform viscosity, which is defined by the local gray level: the blobs grow faster in dark areas, and slower in the bright regions that separate two cells, ensuring that they cannot cross cell borders as they grow (see Figure 15). We thus obtain a correct over-segmentation of the cells, where each cell is exactly defined by a union of neighboring blobs.

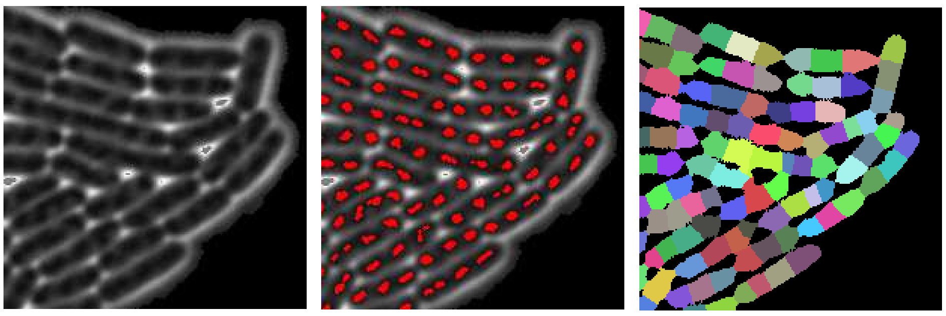

(1) Renormalized image (2) Cleaned-up seeds (3) Blobs

Figure 15: Overview of the non-uniform dilation process. (1) A renormalized colony image where the cells are surrounded by bright halos, (2) the detected seeds that have been cleaned up with a morphological opening, and (3) the corresponding blobs obtained using a non-uniform dilation. Some light over-segmentation errors subsist, particularly in the areas where seeds were not regularly spaced, but the non-uniform dilation mitigates this problem by allowing other seeds belonging to the same cell to expand before seeds from other cells have a chance to cross cell borders.

Let us precise a little bit the dilation process. We assume that we are given N disjoint sets (the seeds) $s_1, ..., s_N$, and a viscosity function $\nu : \Omega \to [1, +\infty[$, where Ω is the image domain. This viscosity function will determine the speed of the dilation at any point in space, the higher the value, the slower the dilation. We define paths on Ω as $\mathcal{C}^1$ functions from $[0, 1]$ to Ω, and the length of a path as

$$\delta(\gamma) = \int_0^1 \nu(\gamma(t)) \, \|\gamma'(t)\| \, dt$$

and the distance $\delta(a, b)$ between two points of Ω as the lower bound of the length of a path connecting the two points, and the distance $\delta(a, X)$ between a point of Ω and a set $X \subseteq \Omega$ as the lower bound of the distance between a and a point of X. We then define the non-uniform dilation of the sets $s_1, ..., s_N$ as the sets (the blobs) $b_1, ..., b_N$ where

$$b_i = \{ x \in \Omega \mid \forall j \neq i, \ \delta(x, s_i) < \delta(x, s_j) \}$$

We can directly translate this definition in the discrete domain, and compute the resulting blobs efficiently using an operation akin to a shortest-path search.

Discrete non-uniform dilation

To define the non-uniform dilation on discrete images, we start by defining a discrete 8-connected canvas, that will be the geometrical basis of our further definitions.

Definition 1 (Canvas). *We define a (8-connected) canvas as a weighted undirected graph* $C = (V, E, \ell)$ *where the vertices set* $V = [1, N] \times [1, M]$ *is a rectangular array of pixels, the edges* E *connect two distinct pixels* p *and* q *if they are 8-connected, that is,* $|p_x - q_x| \leqslant 1$ *and* $|p_y - q_y| \leqslant 1$*, and* $\ell : E \to \mathbb{R}_+$ *are the lengths of the edges. In practice, we will choose* $\ell_{\{p,q\}} = \|q - p\|$.

Definition 2 (Viscosity). *A viscosity on a canvas* $C = (V, E, \ell)$ *is a function* $\nu : V \to [1, \infty)$.

Definition 3 (Path). *If* $C = (V, E, \ell)$ *is a canvas, and* p, q *are two pixels in* V*, a path* $\gamma = (e_1, ..., e_n)$ *from* p *to* q *(denoted* $\gamma : p \rightsquigarrow q$*) is a sequence of consecutive edges from* E *starting from* p *and ending in* q.

Definition 4 (Path length). *If* $C = (V, E, \ell)$ *is a canvas and* ν *is a viscosity on* C*, we define length of a path* $\gamma = (e_1, ..., e_n)$ *by*

$$\pi(\gamma) = \sum_{i=1}^{n} \ell_{e_i} \nu_{e_i},$$

where $\nu_{\{p,q\}} = \frac{1}{2} \cdot \big(\nu(p) + \nu(q)\big)$.

Definition 5 (Distance). *If* $C = (V, E, \ell)$ *is a canvas and* ν *is a viscosity on* C*, the distance between two pixels* p *and* q *in* V *is the shortest path length from* p *to* q*, that is*

$$\delta(p, q) = \inf_{\gamma : p \rightsquigarrow q} \pi(\gamma),$$

and similarly, the distance between a pixel p *and a set* $s \subseteq V$ *is*

$$\delta(p, s) = \inf_{\gamma : p \rightsquigarrow s} \pi(\gamma),$$

Definition 6 (Discrete non-uniform dilation). *If* $C = (V, E, \ell)$ *is a canvas,* ν *is a viscosity on* C*, and* $s_1, ..., s_N$ *are disjoint subsets of* V *(the seeds), we define the dilated* $b_1, ..., b_N$ *(the blobs) as*

$$b_i = \{\, p \in V \mid \forall j \neq i,\ \delta(p, s_i) < \delta(p, s_j)\}$$

Algorithm

Let $C = (V, E, \ell)$ be a canvas, ν a viscosity on C, and $s_1, ..., s_N$ be the initial seeds. We now describe an efficient algorithm to compute the dilated $b_1, ..., b_N$. In what follows, we will assume that the seeds and the viscosity are such that each pixel from V belongs to exactly one of the $b_1, ..., b_N$. If this is not the case, we will randomly assign the pixel to one of the b_i at equal minimal distance from it.

The simplest way to compute the result of the non-uniform dilation takes $\mathcal{O}(N \cdot |V| \log|V|)$ time and $\mathcal{O}(N \cdot |V|)$ space. It consists in computing the distances from each seed to all the other pixels using the Dijkstra shortest-path computation algorithm [Dijkstra, 1959] and associating each pixel to the closest seed. The memory requirements can be reduced to $\mathcal{O}(|V|)$ by updating the labeling of the blobs and the minimal distances from the blobs to each pixel in a sequential way.

The computational requirements can however be improved by using an algorithm akin to a fast-marching algorithms [Sethian, 1996] modified to take into account the simultaneous dilation of the blobs. This reduces the computational time to $\mathcal{O}(|V| \log|V|)$ and the memory requirements to $\mathcal{O}(|V|)$, when using a modified heap structure with a back-pointer for the look-up and update of pixel distance bounds (see for instance ivi).

Algorithm 1) evolves an array of labels corresponding to each blob, where the label $\textsc{Ens}(i)$ corresponds to blob b_i, as well as an array of boolean that indicates which pixels have been definitely labeled. The pixels of the image are progressively attributed a label and frozen as the seeds are simultaneously dilated.

```
Algorithm: Discrete non-uniform dilation
input : $C = (V, E, \ell)$ the canvas
input : $\nu$ the viscosity
input : $s_1, ..., s_N$ the seeds
output: $b_1, ..., b_N$ the dilated blobs
// Initialization
Let L be an array of labels,
where $L(p) = \textsc{Ens}(i)$ if p belongs to $s_i$, and $L(p) = \textsc{Null}$ otherwise
Let F be an array of booleans (the frozen pixels),
where $F(p) = \mathsf{true}$ if p belongs to one of the $s_i$, and $F(p) = \mathsf{false}$ otherwise
Let H be the set of pixels in the seeds that have at least one unfrozen neighbor, sorted by
(a bound on) the distance to the seed they belong to (initially, this is 0):
$H \leftarrow \{ (p, 0) \mid \exists i, p \in s_i \text{ and } p \text{ has an unfrozen neighbor} \}$
// Computation
while H is not empty do
    $(p, \delta_p) \leftarrow$ remove from H the pixel having minimal distance
    $F(p) \leftarrow \mathsf{true}$ // we freeze p, its definitive distance is $\delta_p$
    foreach unfrozen neighbor q of p do
        if q is in H then
            $\delta_q \leftarrow$ the distance bound associated to q in H
            if $\delta_p + \nu_{\{p,q\}} < \delta_q$ then
                $L(q) \leftarrow L(p)$
                Update the distance bound on pixel q in H with $\delta_p + \nu_{\{p,q\}}$
            end
        else
            $L(q) \leftarrow L(p)$
            $H \leftarrow H \cup \{(q, \delta_p + \nu_{\{p,q\}})\}$
        end
    end
end
return $b_1, ..., b_N$, where $b_i = \{ p \mid L(p) = \textsc{Ens}(i) \}$
```

Algorithm 1: Compute the discrete non-uniform dilation of seeds in an image

Of course, simply dilating the seeds would completely fill the image (since the viscosity function does not explode to infinity), and in the experiments, we limited the dilation process to a fixed maximum distance $\delta_{\max}$.

The viscosity function we used in the experiments was arbitrary (but reasonable) and could be used with all the image sequences we encountered. Recall that since the images are normalized, the viscosity function is not image-dependent but defined on an absolute scale.

2.2.3 Blobs and connection graph simplification

The initial over-segmentation thus obtained generally contains a large number of small blobs, and can be improved by iteratively applying some conservative simplification rules to obtain a simpler over-segmentation with larger blobs.

Clean-up of the blobs

The first step is a clean-up of the very small blobs that are due to image noise or over-segmentation artifacts. We merge all very small blobs (in our experiments, we chose to a threshold of 90 pixels) with the neighboring blob sharing the largest border, and which is not itself a very small blob, and iterate this clean-up process until we reach a fixed-point where all spurious small blobs have been eliminated (see Figure 16).

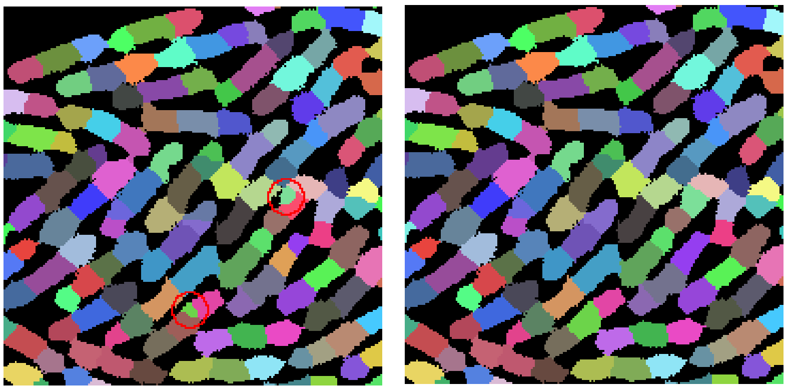

(1) Result of non-uniform dilation (2) After clean-up

Figure 16: Clean-up of small blobs. (1) Result of the non-uniform dilation, where small blobs have been highlighted in red, and (2) after merging small blobs with their largest neighbor once.

Connection graph

By definition of an over-segmentation, each blob which is not already a complete cell, and which is too small to be a cell by itself, must be connected to some neighboring blobs. Following this idea, we define a connection graph, where the blobs are the vertices, and edges link neighboring blobs (that is, blobs that share a long enough boundary). The cells are then "linear sub-graphs" (homeomorph to a segment) of this connection graph. In particular, if a blob has no neighbor in the connection graph, our hypothesis implies that it *must* be a perfectly segmented cell.

We first eliminate blobs that are due to spuriously detected seeds (see Figure 17) by removing all small blobs that are isolated in the connection graph (we chose a conservative threshold of 300 pixels on the data that we have been given).

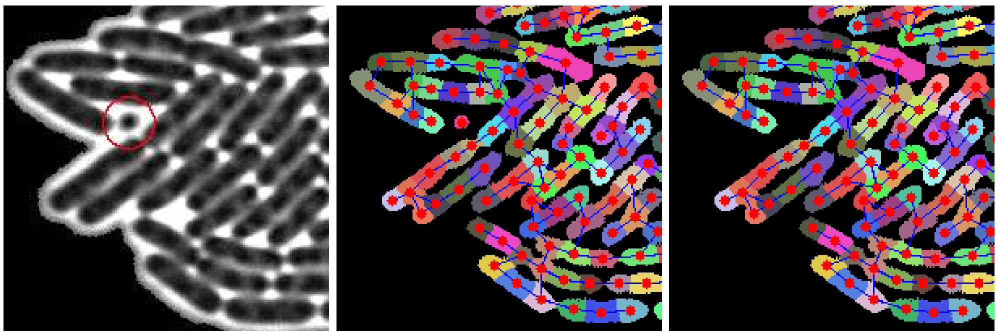


Figure 17: **Clean-up of spurious blobs.** The bright halo around the cells has left a small patch of dark pixels in the colony (highlighted in red), which has been detected has a small blob seed that has no neighbor in the connection graph. Such over-segmentation artifacts are easy to detect and eliminate.

Blob simplification

Not all edges in the remaining cleaned-up connection graph are meaningful, and many of them can be removed using some simple conservative criteria that ensure the correctness of the new over-segmentation. The assumptions we use are (A2) – the fact that a cell has a minimum width – and

(A4) There is a minimum area A of a cell

(A5) The border between two blobs in the same cell has a minimum length ℓ.

To cope with bent cells that are frequently found in the images, the definition of the width of a cell is that of the minimal-width enclosing annulus (see Figure 18)

$$\text{width}\,(u) = \inf\left\{\, w \geqslant 0 \mid \exists y \in \mathbb{R}^2, \exists s \geqslant 0, \quad \forall x \in u,\; s \leqslant \|x-y\| \leqslant s+w \,\right\}$$

Computing the exact minimal-width enclosing annulus rather complex, and we thus chose to approximate it: we first compute the orientation θ_u of the blob u

$$\theta_u = \frac{1}{2}\arctan\frac{2\mu_{1,1}}{\mu_{2,0}-\mu_{0,2}}, \quad \text{where} \tag{2.2}$$

$$\mu_{p,q} = \frac{1}{|u|}\sum_{x,y\in u}(x-x_u)^p(y-y_u)^q \quad \text{and} \quad (x_u, y_u) = \left(\frac{1}{|u|}\sum_{x\in u} x, \frac{1}{|u|}\sum_{y\in u} y\right)$$

and we restrict our search to enclosing annulus that lie on the line orthogonal to the blob orientation θ_u that passes through the blob centroid (x_u, y_u). In practice, we discretize a number of points regularly spaced on this line, we compute the enclosing annulus centered on those points, and we return the minimal width of such an annulus.

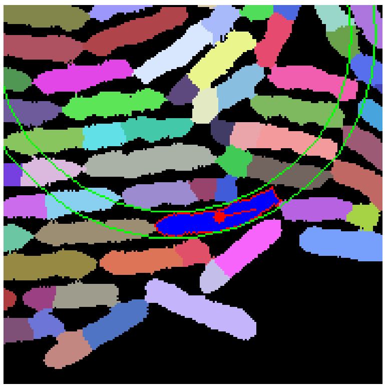


Figure 18: **Definition of the width of a blob.** We compute the width of a blob as that of the minimal-width enclosing annulus to cope with bent cells.

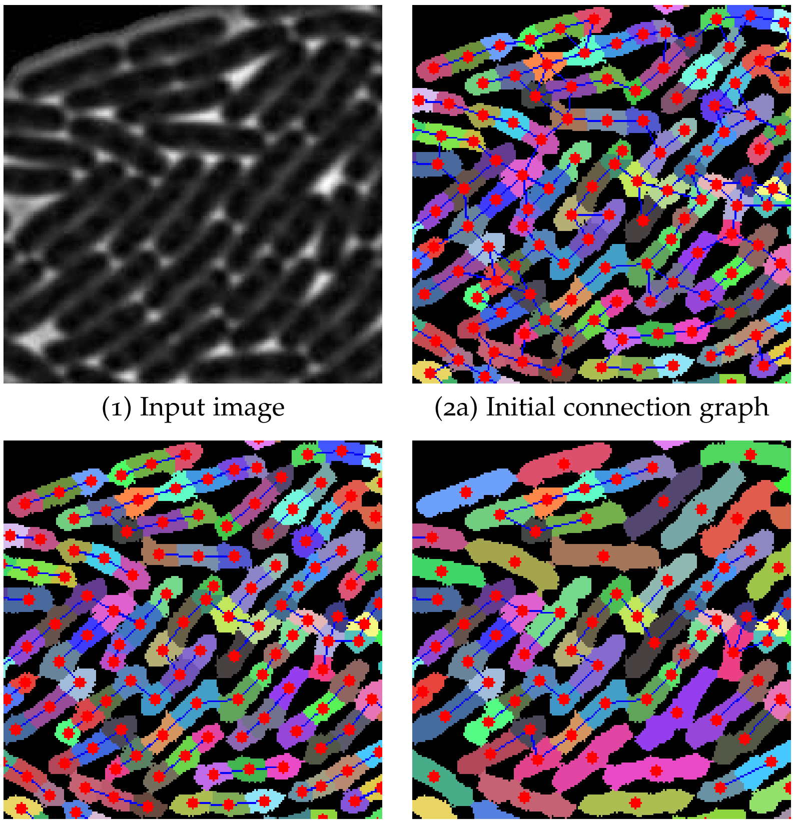
By iterating the processes of removing edges in the connection graph using assumption (A2) between two blobs whose union would be too wide to possibly be a cell or (A5) between two blobs whose common border is too small for the blobs to belong to the same cell, and merging the blobs using assumption (A4) (a blob that is too small to be a cell on its own and that only has one neighbor must be merged with its neighbor), we achieve a significant simplification of the over-segmentation (see Figure 19).

2.3 CELL SEGMENTATION AND TRACKING

Now that we have an over-segmentation and the connection graph, we can generate all potential cells that are consistent with these bounds – in other words, all the unions of blobs on "linear sub-graphs" of the connection graph described in the previous section. Some of these potential cells are necessarily true cells, as they are isolated vertices of the connection graph. In general, there are enough true cells after the blob simplification process, but we can always assume that one or two images (say the first and the last) have been manually segmented, and only contain perfectly segmented cells, so we obtain enough information to start the segmentation and tracking algorithm.

To segment and track the cells, we will define a probabilistic cell evolution model based on biological knowledge, and learn its parameters from manually segmented and tracked sequences. The idea is then to segment and track cells iteratively, by choosing the most obvious decisions first, in order to make very few errors. Once the obvious choices have been fixed, this will hopefully greatly increase the constraints on the other decisions, which will in turn become obvious. This notion of an obvious choice is translated in the probabilistic notion of risk – a ratio of transition likelihoods.

We chose to process the movie in a non-sequential way, to be able to make decisions at any time step, and not only from the first to the last frame. This constrains our cell evolution model, but enables us to always choose the most obvious decision in the complete sequence.


(1) Input image (2a) Initial connection graph

(2b) After edge removal (rule A2) (2c) After blob merging (rule A4)

Figure 19: Connection graph simplification. A typical cell image (1), the initial connection graph of blobs (2a), and the two steps of the graph simplification process (2b and 2c), iterated until convergence. Note that the isolated blobs of the graph (2c) necessarily are cells.

2.3.1 Cell transition likelihood

To assist us in tracking the cells, we define the likelihood of the transition of a cell A in frame n to a cell B in frame $n+1$ (denoted $A \to B$), as well as the likelihood of the division of a cell A in frame n into cells B and C in frame $n+1$ (denoted $A \to (B, C)$). Note that we are speaking here of *completely segmented cells*, and not just blobs.

We have chosen simple, yet discriminant enough measurements on the cells: we extract from each cell C the centroid position $x_C \in \mathbb{R}^2$, the area $\mathcal{A}_C \in \mathbb{R}_+$ and the orientation $\theta_C \in [-\pi, \pi]$ as defined in equation (2.2).

We will assume that the process is Markovian, and that the transition probability only depends on the state of the mother cell and the daughter or daughter cells, and not of the previous or next transitions. This is crucial in allowing us to make decisions at any time step in the sequence, as we do not want to rely on the fact that knowledge on the cell in the previous or next frames is readily available. We model the cell transition $A \to B$ as

$$\pi_{A\to B} = (1 - \pi_{\text{div}}) \cdot \pi_x \left(\frac{x_A - x_B}{|x_A|} \right) \cdot \pi_\theta \left(|\theta_A - \theta_B| \right) \cdot \pi_{\mathcal{A}} \left(\frac{\mathcal{A}_A}{\mathcal{A}_B} \right)$$

where the probability that a cell divides (π_{div}) is determined empirically (it depends on the frame rate) and the probability densities π_x, π_θ and $\pi_\mathcal{A}$ are designed according to biological knowledge and their parameters are learned from (previously processed) reference sequences (see Figure 20). Note that when analyzing sequences with varying frame-rates, we assume that the learned probability laws can be scaled linearly with the frame-rate.

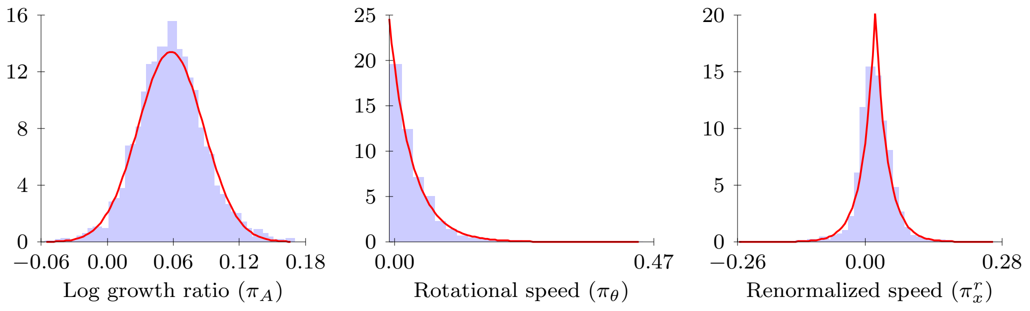


Figure 20: Cell evolution model. Learned probability density functions for the cell evolution model: log growth rate (Gaussian), rotational speed (exponential) and radial component of the speed (Laplace; the tangential component is similar up to a scale factor).

Indeed, the area of a cell is expected to grow exponentially, its orientation is often slowly-varying, and concerning the speed (π_x), we chose to measure the relative motion $(x_A - x_B)/|x_B|$ instead of the absolute motion $x_A - x_B$, as the cell motion results from the cells in the center of the colony pushing the other ones because of their growth, so that we expect the motion amplitude to be roughly proportional to the distance to the center of mass of the colony (see Figure 21).

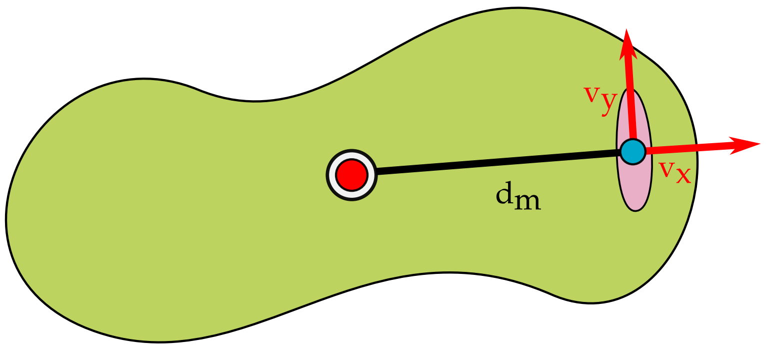


Figure 21: Motion estimation. The motion estimation relies on the hypothesis that the motion is mostly due to the pressure exerted by the colony on the cell during its growth. Thus, we will measure the variables v_x/d_m and v_y/d_m, where d_m is the distance from the center of the cell to the center of the colony, and (v_x, v_y) is the bacteria speed expressed in the local coordinate system shown in red.

To define a similar likelihood for the transition $A \to (B, C)$ (A divides into B and C), we note that if the frame rate is not too small, the union of the two new cells (B and C) can be considered as a single cell when the parameters x, $\mathcal{A}$ and θ are measured, so that it is natural to write $\pi_{A\to(B,C)} = \pi_{A\to B\cup C} \cdot \pi_{\text{div}}/(1-\pi_{\text{div}})$. In practice, we also add a deterministic threshold on the distance between the daughter cells (computed as the minimal distance between two points in each cell), to avoid considering cells that are too far apart.

2.3.2 Likelihood versus risk

By taking the product of the likelihood of all its local transitions $A \to B$ and $A \to (B, C)$, we can define the likelihood of a complete lineage (each cell of the lineage being a union of blobs). Ideally, we would like to find the lineage that has the highest likelihood, with the constraints that each blob belongs exactly to one single cell. However, this global optimization problem seems computationally intractable, and in particular, assignment algorithms Kuhn, 1955 cannot handle such constraints.

If we aim to recover the lineage by taking only local tracking decisions, the most natural way to define the successor of a cell is to associate it to its best match in the next frame. A natural idea would therefore be to start with the most likely decision, then the second possible most likely decision taking into account the first one, etc. However, the tracking decisions are order-dependent, as the best match of a cell could also be the best match of another one, so rather than taking first the most likely decision, we propose to take first the least ambiguous one.

Indeed, some of the segmentation and tracking choices seem obvious, and making those obvious decisions would help constrain the uncertainty for the rest of the decisions to take, without taking too much risk. If we observe a typical image sequence (see Figure 22), a natural idea would be to start by tracking the cells on the border of the colony, as they are naturally constrained (they have fewer neighbors). Once those cells have been correctly associated, we can "remove them" and continue the same process with the new border cells.

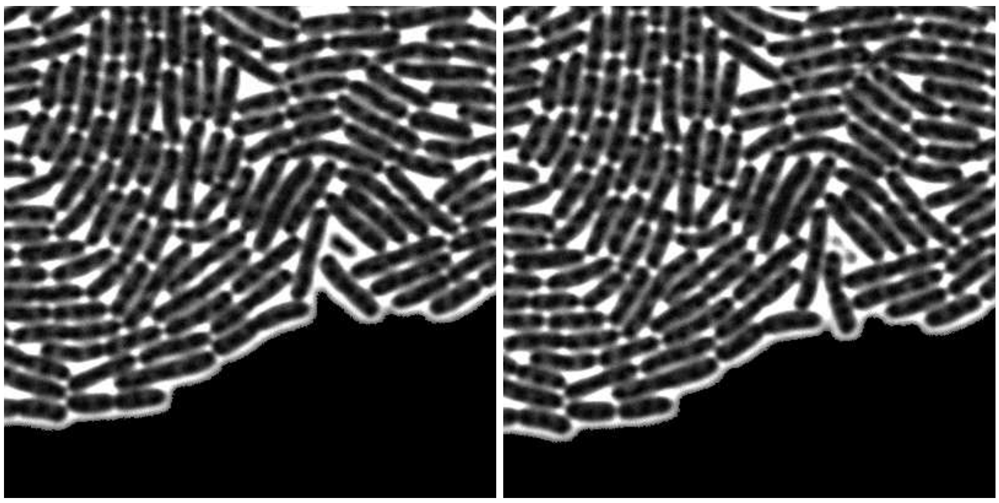


Figure 22: **Making obvious choices first.** Two successive frames of a typical sequence. An human operator tracking the cells would most likely start by associating the cells on the border, as they are more constrained than the other (they have fewer neighbors, and there is thus much less uncertainty on their real transition). We abstract this concept in a notion of risk for the transitions.

	original	1 frame / 2	1 frame / 4
(1)	$\rho_{\text{max}} = 0.0026$	0.23	0.89
	$\rho_{\text{average}} = 3.2 \cdot 10^{-6}$	0.001	0.03
(2)	$\rho_{\text{max}} = 0.0001$	0.06	0.24
	$\rho_{\text{average}} = 2.3 \cdot 10^{-7}$	0.0003	0.01
(3)	$\rho_{\text{max}} = 6.9 \cdot 10^{-6}$	0.004	0.11
	$\rho_{\text{average}} = 3.5 \cdot 10^{-8}$	$7 \cdot 10^{-5}$	0.005

Table 1: Maximum and average risks encountered by the three algorithms when extracting the lineage from the original movie and various sub-sampled versions. We clearly see that the max and average risks that have been taken are always lower when the "obvious-first" algorithm is used, thus giving an increased confidence in its results.

Rather than directly using this solution, we abstracted the concept of a the risk of a transition: given a transition $A \to B$ (or, similarly, of a cell division $A \to (B, C)$), we define its risk by

$$\rho_{A \to B} = \max_{X \neq B} \frac{\pi_{A \to X}}{\pi_{A \to B}},$$

the maximum being taken over all potential successors X of A (B excepted), and the risk being equal to 0 if there is only one potential successor. Intuitively, the risk is very low when the transition has no credible alternative, and rises when there is a doubt on the successor of the cell. Since most of the cells have a trivial motion, we can hope that they will be processed early and correctly and will rule out some choices concerning other cells for which the initial risk was high (see Fig. 23). Note that this notion of risk is also related to the robustness of the algorithm, since the maximum risk can be understood as the level of degradation that the algorithm can handle before changing its choices.

2.3.3 Tracking segmented cells

To quantify the benefits of using the notion of risk, we implemented three tracking algorithms working on completely segmented sequences:

(1) the *any-first* algorithm, which orders the cells arbitrarily,

(2) the *likely-first* algorithm, which sorts the cells by decreasing likelihood of their best transition,

(3) and the *obvious-first* algorithm, which sorts the cells by increasing risk of their best transition.

Each algorithm works in the same way: to each cell in the order given by the algorithm, we associate its best possible match in the next frame, and this process is iterated until all cells have been tracked.

To compare the three algorithms, a sequence of manually segmented cell images was degraded by under-sampling it twice (keeping one frame in two) and four times (one frame in four), and we verified that no cell divides more than once between two successive frames. The two first movies would be relatively easy to track for a human operator, but the third becomes more delicate, and one has to view successive frames back and forth several time to make sure to have the correct cell association. If we subsample more than twice the sequence, the cell association becomes difficult even for an human operator, some cells dividing several times between two successive frames, and we do not expect an algorithm to give a correct lineage.

We tracked the cells in each sequence with each algorithm, adapting the parameters of the probability laws to the corresponding frame rates, and we computed the average and the maximum risk taken (see Table 1). As can be seen, the more degraded the film, the higher the risk. Algorithm (1) makes some errors when tracking the most degraded film (one frame in four), and the other algorithms always give correct results, although we see that algorithm (3) leads to minimal risks, and thus gives us more confidence in the result of the tracking.

2.3.4 Tracking over-segmented cells

We apply this risk-based approach to the potential cells defined by the blobs and the connection graph, as union of consecutive blobs in a linear subgraph of the connection graph (see Figure 24).

We eliminate some of the potential cells using the conditions (A2) and (A4), allowing us to keep the number of potential cells reasonable for the first 7 or 8 generations, but this number quickly explodes afterward, and more conservative conditions will be needed to let us process longer sequences.

Among these potential cells are some perfectly segmented cells (isolated vertices of the connection graph), that we label as initial "active" cells. For each active cell, we compute the risk of its associated best transition to any potential cell in the next frame, and then select the transition having the minimal risk among the active cells. The target cell of the selected transition now becomes active, all the potential cells overlapping it are deleted, and we recompute the risks of the transitions for the new set of active cells. This process is then iterated until the complete lineage has been obtained. We also consider the risks of the transitions in the backward direction (knowing the cell, what is its best-match predecessor?) and apply the same process to these transitions (see algorithm 2).

Segmentation errors

Some errors made by Céleste are potentially easy to correct: sometimes, a small blob is missing from a cell (see Figure 25), and the cell it belongs to will generally be easy to guess using post-processing heuristics.

A more difficult to solve error is depicted in Figure 26. This is caused by a previous segmentation error that propagates in the sequence, cascading into several other segmentation and tracking errors. Sometimes, such segmentation errors result in new situations where a cell has no likely successor, yet one of them is still much higher than any other possibility (for some reasons), yielding a very low risk for the transition. A way to avoid this kind of error would be to only allow transitions having a low risk and a high likelihood.

This phenomenon is amplified by the fact that, contrary to the case where we were tracking already segmented cells, we no longer make decision by considering all the cells in all images, but only those who have already been completely segmented. In particular, when starting the segmentation and tracking process, there are only few such completely segmented cells, and choosing the transitions with minimal risk makes less sense.

To remedy this problem we do several successive segmentation and tracking passes, by feeding the next segmentation and tracking pass with the cells that have been segmented in the previous pass (we can heuristically remove some cells that are deemed to be poorly

```
Algorithm: Céleste
input  : $I = (I_1, ..., I_K)$ the image sequence
output: $C = (C_1, ..., C_K)$ the segmented cells and the lineage
$s = (s_1, ..., s_N) \leftarrow$ extract_seeds($I$)
$\bar{I} \leftarrow$ renormalize_intensities($I, s$)
$C = (C_1, ..., C_N) \leftarrow$ over − segmentation($\bar{I}, s$) // the over-segmentation and connection
    graph
All perfectly segmented cells are activated
while true do
    foreach active cell c that has no successor do
        Compute the risk of the transition to any potential cell without predecessor in the
        next frame
    end
    foreach active cell c that has no predecessor do
        Compute the risk of the transition to any potential cell without successor in the
        previous frame
    end
    if there is no such transitions then
        Stop
    else
        Choose the transition of minimal risk
        Update the corresponding over-segmentation and lineage by segmenting and
        linking the cells in the transition
        The cells in the transition become active
    end
end
return $(C_1, ..., C_K)$
```

Algorithm 2: The Céleste cell segmentation and tracking algorithm

segmented if needed). Hence, in successive passes, the number of initial cells increase, and the risk-first tracking algorithm becomes meaningful and useful.

In the first pass, we use a threshold on the risk and likelihood as suggested above, to ensure that we only make very likely segmentation choices, and thus "bootstrap" our segmentation. In the next passes, we successively lower the thresholds until all the cells in the images can be successfully tracked and segmented (see Figure 27). This enables us to correct some of the segmentation and tracking errors, as shown in Figure 28.

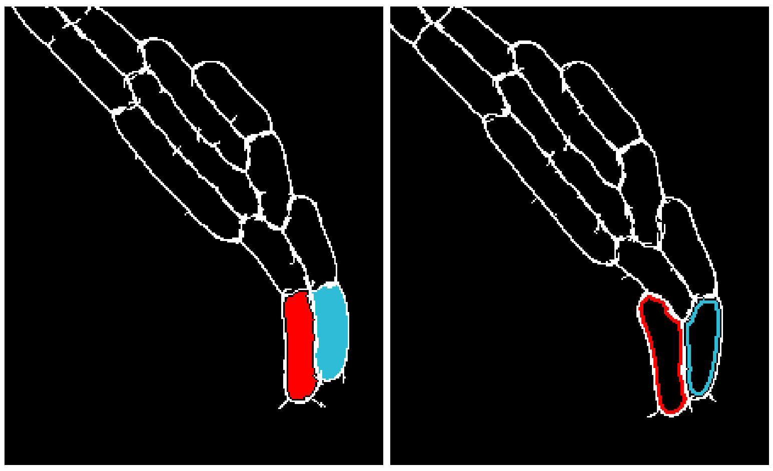
2.3.5 Software

We implemented in C (using the MegaWave library) a complete segmentation and tracking suite called Céleste that has a user-friendly interface and permits to visualize and correct the results in a straightforward and natural manner (see Figure 29). With this software, a complete sequence requires almost no interaction for the 5-6 first generations, and about a couple of hours for 9-10 generations, which seriously improves previously-used algorithms. This software is planned to be revised in 2012 by a software engineer to meet open-source standards and be made available to the research community.

2.3.6 Current results

On the films that we processed, we could handle images containing about 120 cells (about 7 generations) before the number of blobs and connections between them was too large for the potential cells to be efficiently generated. Developing discriminatory but conservative shape constraints on the potential cells to simplify the connections between the blobs could break this limit, and is thus an interesting challenge. Currently, we sometimes use the graphical user interface to segment manually a few blobs with a high number of connexions in the blob graphs to reduce the combinatorial complexity. With the current algorithm we proposed, if we want to handle films up to 100 frames (9-10 generations), we have to assist the connection graph simplification algorithm by manually deleting some connection edges (or by using supervised heuristics). Such an interaction also permits to fix some little segmentation errors, that can propagate and induce tracking errors. On the first frames (about 5 to 6 generations on most films), we have almost no error (see Fig. 30). In the last frame (9-10 generations), the number of cells suddenly increases, so the connection graph becomes very large, and there are a lot of potential cells, which slows down the computations and introduce more errors (when there are many potential cells, there are more cells that "look like good successor"). Figure 32 shows an extracted lineage tree, where the length of the branches are proportionnal to the duration between two successive divisions, and the color corresponds to the distance between the cell and the center of the colony.

Our software is in use in the TaMaRa's lab, and further analysis at the laboratory by our collaborator in this team, Alice Demarez, has acknowledged in her doctoral thesis that Céleste not only decreased significantly the time that users spend on the segmentation and tracking of cells in comparison with the previous BHV software, but also that the quality of the results has increased. Jean-Pascal Jacob has also worked on the project to define a better model for the bacteria shape as a skeletal segment that is recovered using energy-minimization techniques, and subsequently dilated in the shape of a rod to recover high-quality contours of the cells (see Figure 31).



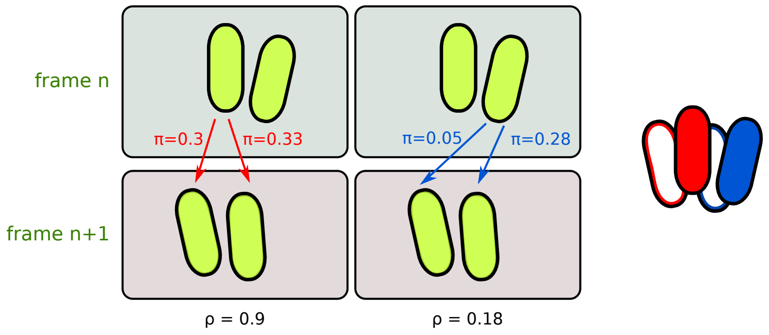


Figure 23: **Illustration of the notion of risk.** (**First line**) A sequence segmented using the legacy BHV algorithm, used to learn the parameters of the probability laws. On this sequence, the likelihood of the transitions from the red cell in the first frame to either the red or blue cell in the second frame are equivalent, but the likelihood of the transitions from the blue cells are more contrasted. An algorithm associating a cell to its most likely successor, and starting with the red cell would make an error, while an algorithm ordering the cells by their transition risk would make the correct decision, as illustrated on the second line. (**Second line**) Example of a dubious (left) and an obvious (right) choice, π is the likelihood of the associated transition, ρ is the risk of the best-match transition. If we take the most likely transition first (left), we will make the wrong affectation, but if we take the most obvious transition first (right), the final result we be correct.

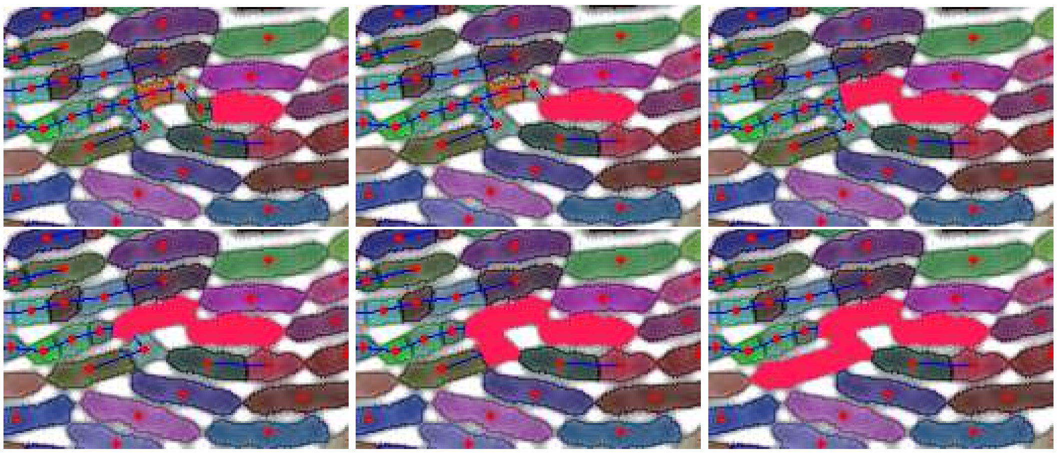


Figure 24: **Potential cells.** Sample potential cells (in red) generated as linear subgraphs of the blob graph. Only the potential cell in the second image corresponds to an actual cell of the image.

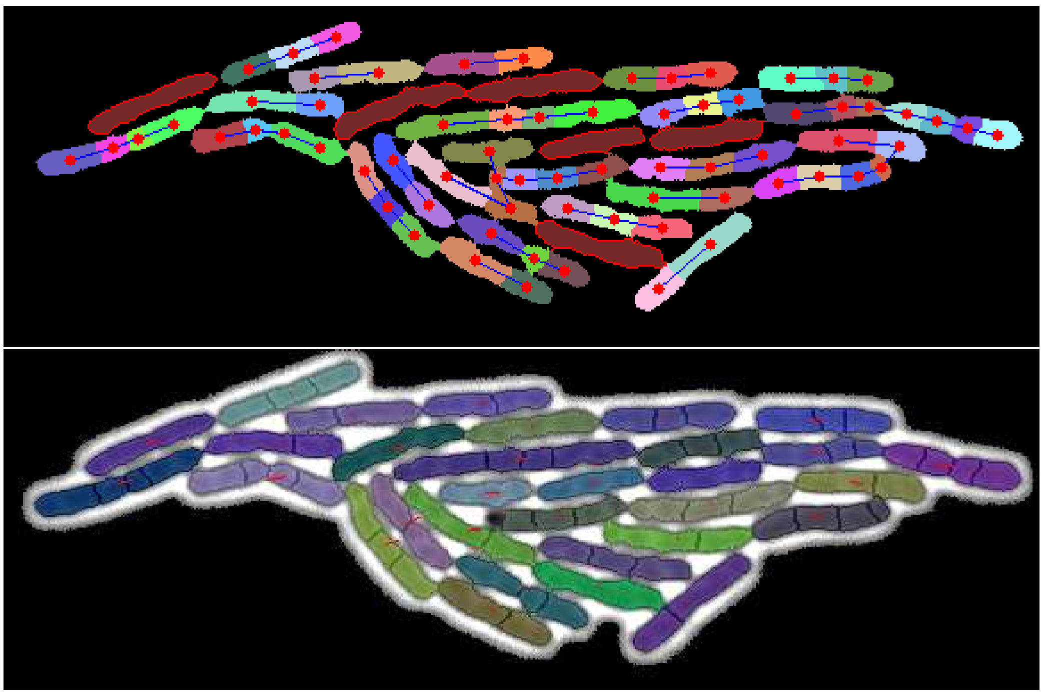


Figure 25: **Missing blob.** (**Above**) The simplified blob graph obtained by the over-segmentation process, and (**Below**) a chunk of a cell corresponding to a small blob is missing from the final segmentation (red vectors represent the cells speeds). Such errors should be relatively easy to detect and correct using a post-processing.

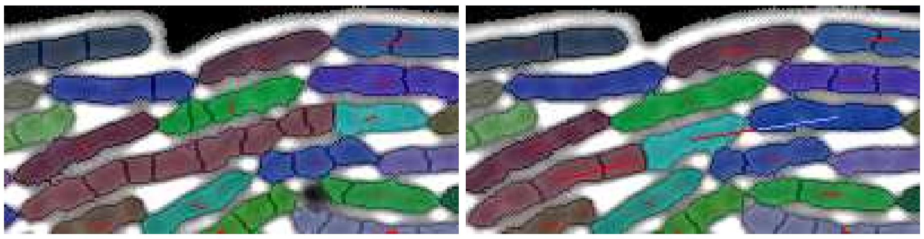


Figure 26: **Cascade segmentation errors.** Two segmentation errors (the red arrows are the cell speeds). As before, a small blob is missing from the green cell. The large cell has been wrongly divided in two (brown and cyan) just before the real cell division. In the next image, one of the daughter cells corresponding to the real division (in blue) has been correctly segmented. When Céleste tries to find the correct successor of the cyan cell, it has the choice to connect the cyan cell to the blue cell (not likely, because cell have wildly varying sizes), or to move it somewhere else. It happens that this latter choice has a much lower risk, and this will cascade in a series of bad segmentation and tracking decision (the brown cell must then shrink).

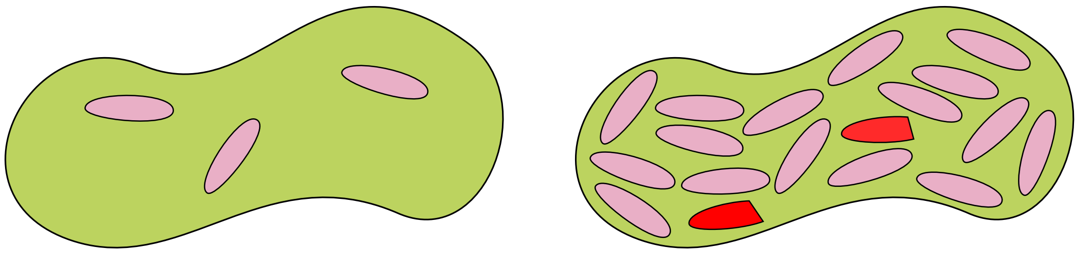


Figure 27: **Multiple passes.** When running the Céleste algorithm on the result of the over-segmentation, only few cells have been statically segmented, and using the risk-first tracking algorithm makes less sense, as we only compare the risks of few cells. To remedy this problem, we iterate the Céleste algorithm by successively feeding to the next pass the results of the previous one. To avoid resuming the next pass with the possibly poorly segmented cells of the previous iteration (shown in red), we use conservative thresholds on the risk and likelihood of allowed transitions in the first passes, and successively decrease them in later passes.

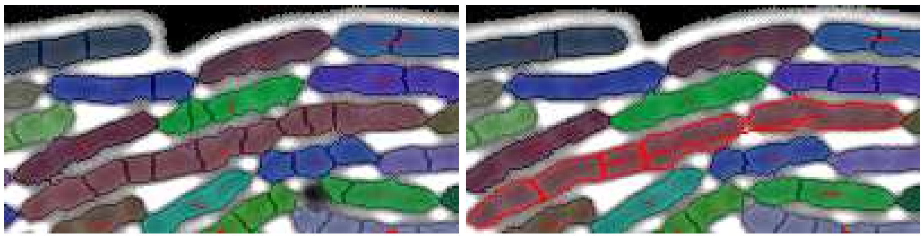


Figure 28: **Correct segmentation using the multiple passes algorithm.** The segmentation error described in Figure 26 has been corrected by using several passes of the algorithm that use decreasingly conservative risk and likelihood thresholds, each successive pass taking as input the completely segmented cells of the previous one.

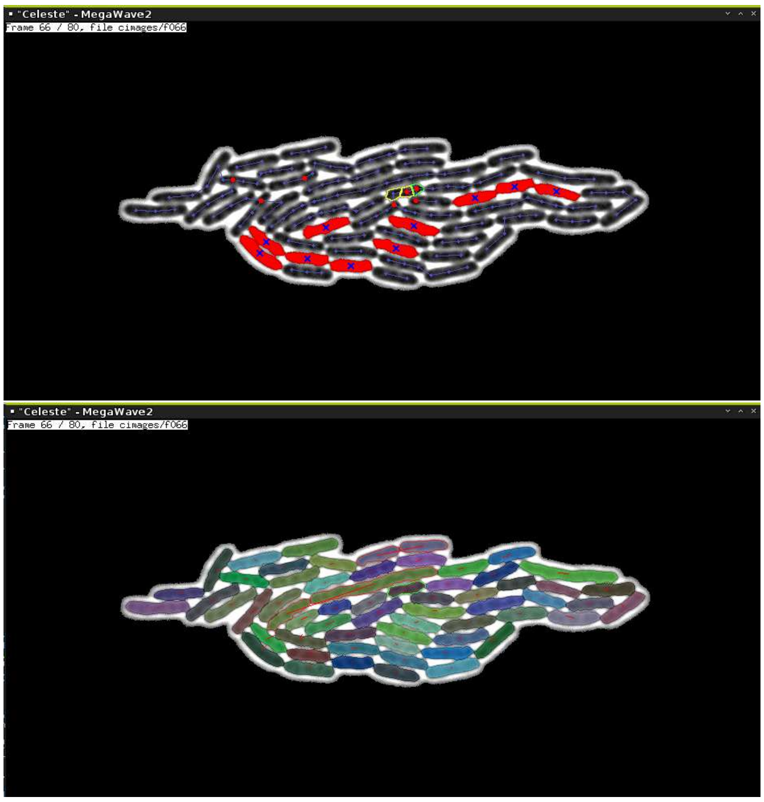

"Celeste" - MegaWave2
Frame 66 / 80, file cimages/f066
"Celeste" - MegaWave2
Frame 66 / 80, file cimages/f066

Figure 29: Céleste. Screenshots from the Céleste segmentation and tracking suite permitting the easy visualization and correction of the segmentation and lineages of the cells.

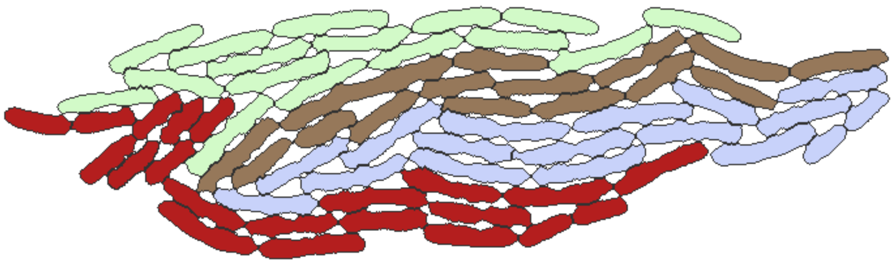


Figure 30: Segmentation result. A part of the result of the tracking algorithm we propose: a completely segmented image, where each cell is colored according to its 3rd generation ancestor. This result was obtained in completely unsupervised way (no human interaction).

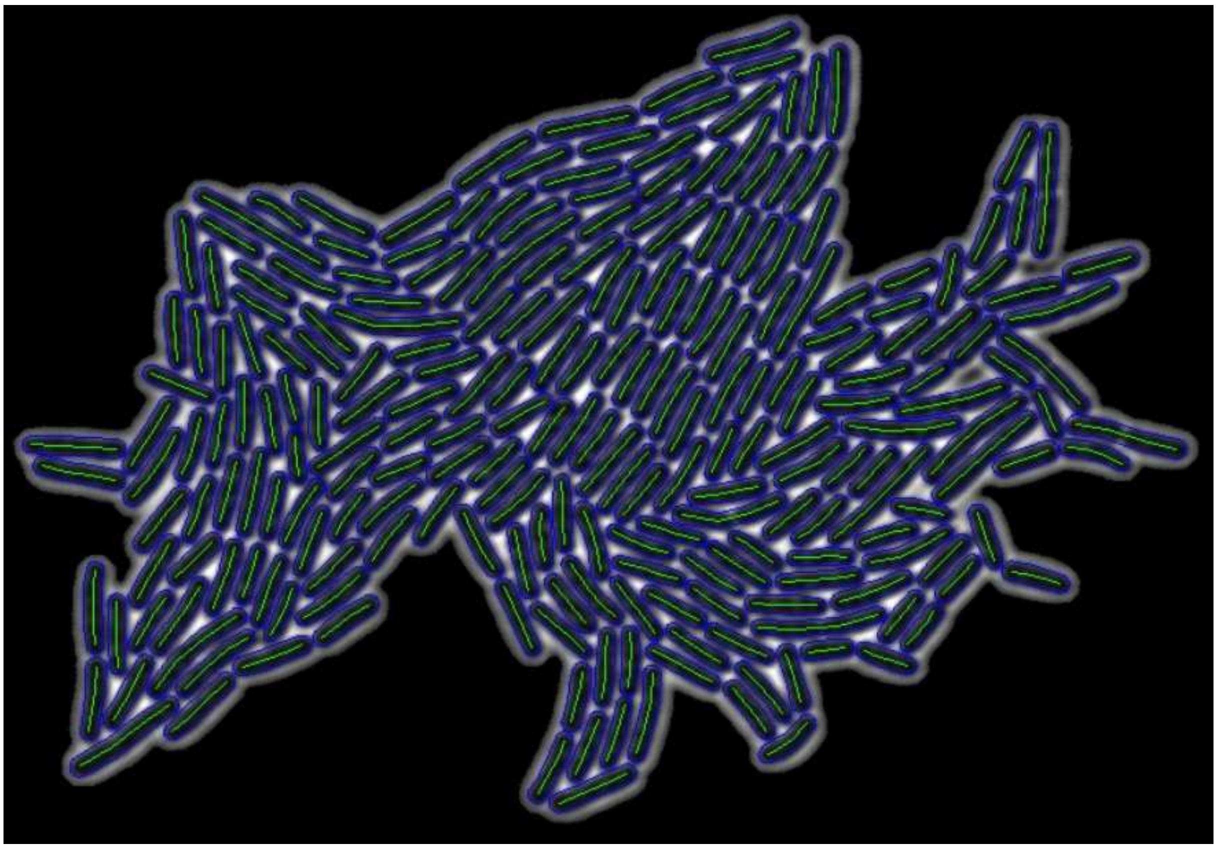


Figure 31: **Post-analysis.** After the segmentation, a post-analysis significantly ameliorates the contour of the cell by representing them as a dilated segment and matching their shape using energy-minimization. *Illustration from Alice Demarez*

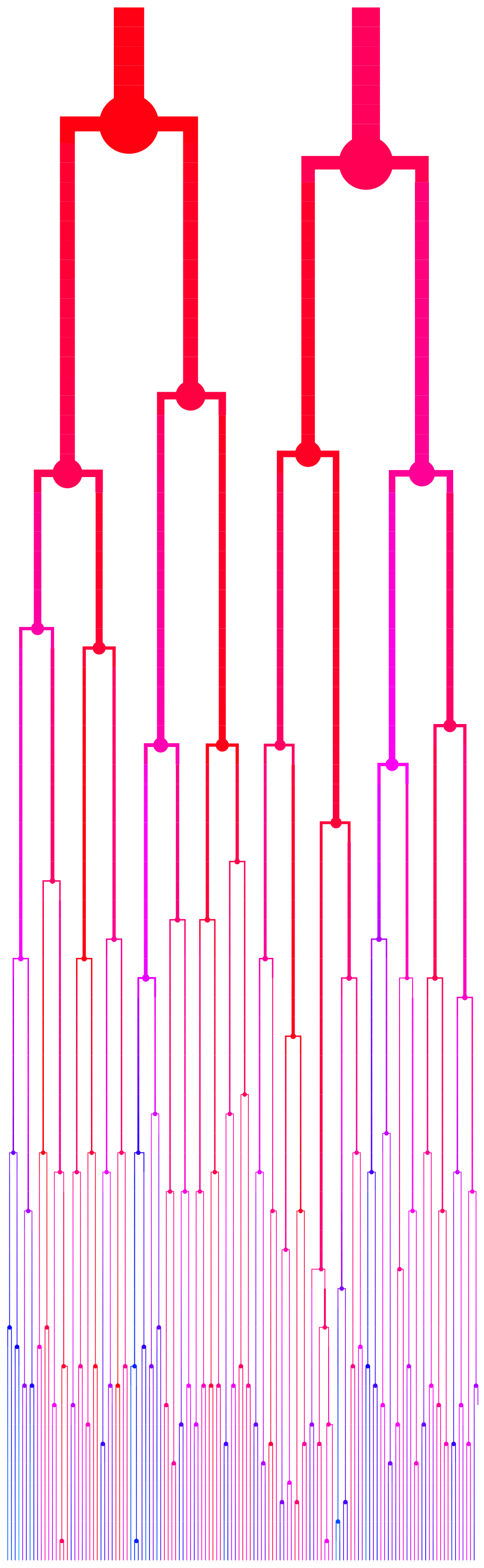
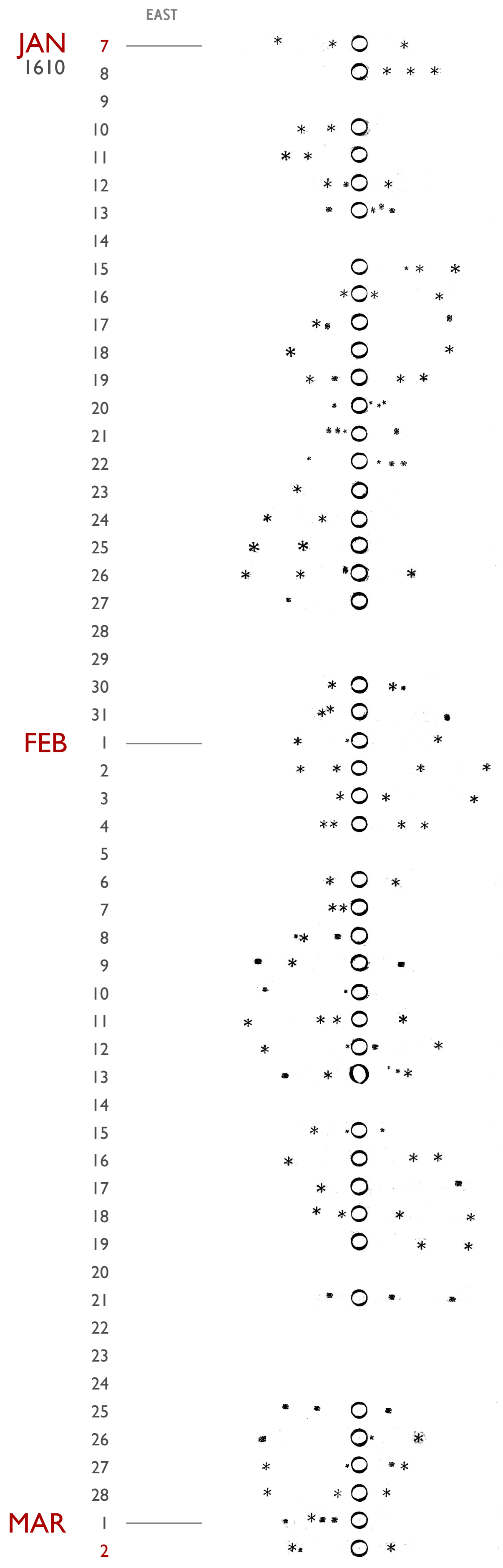


Figure 32: **Bacteria lineage.** Representation of a lineage, where nodes correspond to divisions. The color encode the distance to the center of mass: when red, the cell is close, when blue, the cell is far



Observations of the four moons of Jupiter by Galileo Galilei from january 7 to march 2, 1610 that convinced him of the invalidity of the Aristotelian view that all heavely bodies were circling the earth.

(When several observations were made on the same day, only one has been retained.)

Adapted from *Sidereus Nuncius*,
Galileo Galilei, Venice, 1610

3
ASTRE: A-contrario single trajectory extraction

CONTENTS

In order to better understand the challenges of object tracking we chose to restrict ourselves to the simpler problem of point tracking, and to derive a statistical criterion, called ASTRE, that could bring a useful insight on the limits of the trajectory tracking problem. The next two chapters – the present one introducing the statistical criterion, and the second comparing its performances to those of a state-of-the-art algorithm – have been directly extracted from a research paper that has been submitted and is waiting for approval.

3.1 INTRODUCTION

Object tracking plays an essential role in a large variety of Computer Vision tasks, among which, for example, particle image velocimetry [Gui and Merzkirch, 1996], monitoring cars

[Koller, Weber, and Malik, 1994], detecting and tracking cells in microscopy sequences [Sbalzarini and Koumoutsakos, 2005; Smal, Niessen, and Meijering, 2008], recognizing human activities [Ali and Aggarwal, 2001], improving human-computer interfaces with head-tracking [Ashdown, Oka, and Sato, 2005], generating special effects for movies [Pighin, Szeliski, and Salesin, 1999], or tracking particles in accelerators [Cornelissen *et al.*, 2008].

Numerous variations on this problem have been formulated, and several algorithms have been developed to try to solve them. A common strategy (see Yilmaz, Javed, and Shah, 2006 for a recent review) is to detect the objects in each image and associate them with an object shape representation, such as points [Serby, Koller-Meier, and Gool, 2004; Veenman, Reinders, and Backer, 2003b], geometric shapes [Comaniciu, Ramesh, and Meer, 2003], outlines [Yilmaz, Li, and Shah, 2004], skeletal models [Ali and Aggarwal, 2001], etc. and with appearance features described, for example, by templates [Fieguth and Terzopoulos, 1997], active appearance models [Edwards, Taylor, and Cootes, 1998], or probability densities of object appearance [Paragios and Deriche, 2000; Zhu and Yuille, 1996].

The detected objects are then tracked across frames using an algorithm that closely depends on the object representation. According to Yilmaz, Javed, and Shah, 2006, tracking algorithms can be broadly classified in three categories:

1. Point tracking [Bar-Shalom, Fortmann, and Scheffe, 1983; Reid, 1979; Shafique and Shah, 2003; Streit and Luginbuhl, 1994; Veenman, Reinders, and Backer, 2003b]. Objects are represented as points and are generally tracked across frames by evolving their state that consists of the object position and motion.

2. Kernel tracking [Avidan, 2001; Black and Jepson, 1998; Comaniciu, Ramesh, and Meer, 2003; Tao, Sawhney, and Kumar, 2002]. Objects are represented by a combination of shape and appearance, for instance an ellipse with a color histogram. They are tracked by computing the motion of the kernel in consecutive frames, often modeled with parametric transforms such as translations and rotations.

3. Silhouette tracking [Bertalmío, Sapiro, and Randall, 2000; Blake and Isard, 1998; Huttenlocher, Noh, and Rucklidge, 1993; Kang, Cohen, and Medioni, 2004; Ronfard, 1994; Sato and Aggarwal, 2004]. Objects regions are estimated in each frame, and are usually tracked by either shape matching or contour evolution.

In this work, we restrict ourselves to the tracking of objects as points, without appearance information (even if, as we shall see later, such an information could be easily included in the model we propose). We assume that these points have already been detected in the sequence, but imperfectly, in the sense that we may observe both spurious points and missing detections. Given such a sequence of frames containing the detected points in each image, the goal is to extract the trajectories as lists of points appearing in successive frames, possibly separated by holes (missing detections), while avoiding spurious points. As usual, we will assume that a given point can belong to only one trajectory, which means that point collisions are ruled by an occlusion principle.

3.1.1 Related work

The most well-known point tracking algorithm certainly is the Multiple Hypothesis Tracker (MHT) of Reid, 1979, that, in theory, enumerates all possible trajectory combinations of the observed points, and selects the one having the maximal likelihood (a probabilistic motion model being given). This problem is in effect NP-hard, and leads to exponentially many trajectory combination lookups (if there are n frames and m points, and we assume there is no occlusion, noise points, or objects leaving or entering the scene, there are already $m!^{n-1}$ possible trajectory combinations), and thus approximate solutions and heuristics are needed to accelerate the search (often requiring thresholds to prune the search tree early). Moreover, this complex model implies parameter tuning to optimize the underlying motion model and the efficiency/coverage trade-off of the heuristics.

To overcome the exponential growth of the state space, researchers have proposed a wealth of heuristics and approximations to the point tracking problem over the years. Such an approximation is the Joint Probabilistic Data Association Filter (JPDAF) proposed by Bar-Shalom, Fortmann, and Scheffe, 1983, which relaxes the hypothesis that the points must be disjoint. Instead of assigning each track ending in frame $k-1$ to a particular object in frame k as in MHT (and thus having several possible hypotheses, resulting in an increase of complexity), the JPDAF algorithm assigns to every track ending in frame $k-1$ a weighted combination of all points of frame k, depending on a likelihood estimate with respect to a predicted position. However, this approach assumes that the number of objects tracked in the images is tracked in the images is constant, and the relaxation of the disjointness hypothesis leads to trajectory merges, which is often an undesired feature.

Sethi and Jain, 1987 propose to solve the correspondence problem greedily. They initialize the trajectories using the nearest-neighbor criterion, and then try to improve the current solution greedily by exchanging correspondences between frames in order to minimize the global cost. They also propose a modified algorithm that alternates between forward and backward passes through the sequence to help mitigate the problem of the nearest neighbor initialization. Their approach is much faster than MHT, but does not permit to take noise, occlusions, entries or exits into account.

Salari and Sethi, 1990 address some shortcomings of the previous method, namely the fact that it assumes a constant number of points in the sequence, and that there can be no entries or exits of objects in the scene. It therefore allows for occlusion, entry and exit of points, as well as the presence of spurious points. Each trajectory is made of either points detected in a frame, or "phantom points", that correspond to added (interpolated) points when there are missing detections. Note that their approach involves two parameters (the maximum allowed speed and the maximum allowed acceleration) that may be difficult to set.

Rangarajan and Shah, 1991 propose to solve the problem by using a proximal uniformity constraint, that combines requirements on the maximum speed and the acceleration of objects. They propose to make the assignment choices in an order driven by a notion of minimal risk, still in a greedy way. They use an optical flow algorithm to initialize the point motion between the first two images, and deal with occlusion and missing detections by using linear interpolation. They do not allow for spurious points, or objects leaving or entering the scene.

Veenman, Reinders, and Backer, 2003b propose a greedy tracking algorithm called ROADS, that is capable of handling missing and spurious detections, as well as entries and exits of objects. Rather than optimizing a global cost, they consider a restricted temporal scope (usually two or three frames forward), and find the optimal assignments minimizing the cost on these frames. Since the restricted problem is still NP-hard, they have some heuristics that help them prune the search tree. They keep the assignment between the two first frames of the local scope, and iterate the process on the following frames. The assignment between the first two frames of the sequence is initialized using the nearest neighbor criterion, and the effect if this approximation is mitigated as in [Sethi and Jain, 1987] by a forward and a backward pass.

Fleuret *et al.*, 2008 propose to track multiple persons in multiple camera views using a probabilistic map of the individuals locations, coupled with a dynamic programming tracking algorithm that tracks each person in isolation, rather than conjointly. The tracking uses both an appearance model and a motion model to describe the objects to track.

In essence, the approaches above either try to bound the search space by restricting the optimization to local choices, or by restricting the simultaneous number of objects tracked. A recent and promising approach introduced in Jiang, Fels, and Little, 2007 tries to do both by optimizing a global criterion using Linear Programming, where the correspondence decisions are not "hard" binary choices, but continuous values in $[0, 1]$, rendering the optimization problem convex, and thus efficiently computable. In practice, the values are almost always equal to either 0 or 1, and it is easy to convert them into disjoint trajectories. The algorithm assumes that the number of objects in the images is constant, but has been later extended in Berclaz, Fleuret, and Fua, 2009; Berclaz *et al.*, 2011 to accommodate a variable number of objects that can enter and leave the scene in prespecified locations. The authors also prove that when there exists a unique global optimum, it is necessarily a boolean optimum.

3.1.2 Trajectory estimation versus trajectory detection

These algorithms have some common limitations. First, when they take a varying number of objects into account, as well as spurious and missing detections, they face the difficulty of choosing an appropriate global cost for their optimization problem. Indeed, they need a penalization for spurious points, and most of the time they will simply introduce a fixed cost for them. This creates a subtle (and quite arbitrary) interplay between the cost of spurious points and the cost of detected trajectories, that is not easy to control and grasp. Second, these algorithms often have many parameters, which is fine in theory, but quickly becomes a hassle when one needs to set them for each practical use. Last, as they all try to solve an optimization problem, these algorithms suffer from a classical flaw: they always find something, since they try to find *the best explanation* of the data in terms of some structure, without trying first to *prove that the structure is present*. In other terms, all these algorithms will, for some values of the parameters, find trajectories in random data made of pure random points (without motion coherence).

In the present work, we propose a new approach for trajectory detection, that can guarantee that no trajectory will be found in general in such random data. This work is based on the a-contrario framework [Desolneux, Moisan, and Morel, 2008], that permits to derive absolute thresholds, that are then used to drive a dynamic programming algorithm. This algorithm is able to analyze trajectories globally in time, and avoids the three aforementioned limitations.

In Section 2, we first consider trajectories without holes, that is, the case where no data point is missing (but, of course, spurious points are expected). After recalling the basic principles of the a-contrario statistical framework, we derive an explicit criterion for trajectory detection and present an algorithm based on dynamic programming. We also analyze some theoretical consequences of the a-contrario thresholds, in particular the link between the number of points, the number of images and the maximum allowed acceleration. Then we arrive at Section 3, where the theory and the algorithm are extended to the more general case of trajectories that contain holes. In Section 4, the state-of-the-art ROADS algorithm [Veenman, Reinders, and Backer, 2003b] is considered and various experiments (following, for most of them, the methodology proposed in the original ROADS paper) are led to compare its performances with the a-contrario algorithm we propose. Aside from a very convenient reduction of the number of parameters (1 for the NFA algorithm, versus 4 in the ROADS experiments), the a-contrario algorithm significantly outperforms the ROADS algorithm in terms of precision, robustness, and sensibility to parameters, both in the no-hole and in the hole version. Experiments are also conducted on real data, namely a *snow* sequence for which the ground truth has been manually obtained. Again, the results are clearly in favor of the a-contrario algorithm. We finally conclude in Section 5, and comment on the strengths, limitations, and perspectives offered by the present work.

3.2 TRAJECTORIES WITHOUT HOLES

In this part, we consider trajectories without holes, that is, we assume that there are no missing detections (but possibly spurious detections, and points leaving and entering the scene).

3.2.1 Principles of the a-contrario framework

The trajectory detection method that we propose relies on the a-contrario framework introduced by Desolneux, Moisan and Morel (see Desolneux, Moisan, and Morel, 2008 for a recent presentation). The idea underlying its development (dubbed "Helmholtz Principle") is that the human visual system detects structures in an image as coincidences that could not appear purely by chance in a random setting. Conceived at first to detect structures issued from Gestalt Theory [Kanizsa, 1980; Wertheimer, 1922], this methodology has been applied to a large variety of image processing tasks, aiming at detecting structures like alignments [Desolneux, Moisan, and Morel, 2000], edges [Desolneux, Moisan, and Morel, 2001], stereo coherence [Moisan and Stival, 2004], spots [Grosjean and Moisan, 2009], image changes [Robin, Moisan, and Hégarat-Mascle, 2010], etc. It has also been successfully applied to the related problem of motion detection in an image sequence [Veit, Cao, and Bouthemy, 2007] where an a-contrario criterion is used to group together close detections that display a similar local motion.

We here recall the formalization of the a-contrario framework as it was presented in Grosjean and Moisan, 2009. The a-contrario methodology is based on two ingredients: a *naive model*, and one or several measurements defined on the structures of interest. The naive model describes typical situations where no structure should be detected. For instance, when trying to discover alignments of points in an image, a naive model could consist of uniform and independent draws of the point locations, where no interesting structure would usually appear (see Fig. 33).

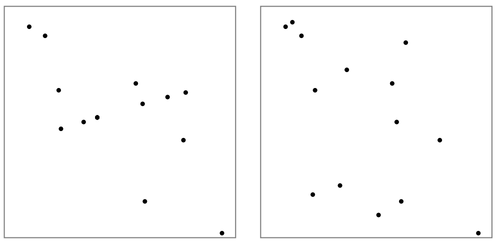


Figure 33: **Illustration of Hemholtz principle.** Why can't we help seeing an alignment of dots on the left image? According to Helmholtz principle, we *a priori* assume that the dots should have been drawn uniformly and independently as in the right image, and we perceive a structure (here a group of aligned dots) because such an alignment is very unlikely to happen by chance. Alignments of four dots can be found in the right image, but they do not pop out, because they are likely to happen by chance considering the total number of points. The formalization of this principle is realized in a-contrario detection models.

To detect structures (e.g. alignments of points) in data using Helmholtz Principle, we need to define in what way an observation can be significant. If the measurement function is high when the structure is pregnant, we can relate the "amount of surprise" when observing the measurement x to the probability $\mathbb{P}(X \geqslant x)$, where X is the random variable corresponding to the distribution of x in the naive model. We will usually have several measurements $(x_i)_{i \in I}$ (in the example above, one for each possible alignment), and in the classical a-contrario framework the amount of surprise will be measured by a *number of false alarms*. More formally, we have the following

Definition 1 (Number of False Alarms). *Let $(X_i)_{1 \leqslant i \leqslant N}$ be a set of random variables. A family of functions $\left(F_i(x)\right)_i$ is a* NFA *(number of false alarms) for the random variables $(X_i)_i$ if*

$$\forall \varepsilon > 0, \quad \mathbb{E}\left(\#\{i \mid F_i(X_i) \leqslant \varepsilon\}\right) \leqslant \varepsilon \tag{3.1}$$

A measurement x_i such that $F_i(x_i) \leqslant \varepsilon$ is said to be *detected at level* ε, or ε*-meaningful*. We say that a measurement is *meaningful* if it is 1-meaningful. This number of false alarms ensures that the average number of detections made in the naive model (that is, false detections) at level ε is less than ε.

The classical way to construct a NFA is given by the following proposition (see Grosjean and Moisan, 2009).

Proposition 1 (NFA construction). *Let $(X_i)_{1 \leqslant i \leqslant N}$ be a set of random variables and $(w_i)_{1 \leqslant i \leqslant N}$ a set of positive real numbers, then the function*

$$\mathrm{NFA}(i, x_i) = w_i \cdot \mathbb{P}(X_i \geqslant x_i) \tag{3.2}$$

is a NFA *as soon as* $\displaystyle\sum_{i=1}^{N} \frac{1}{w_i} \leqslant 1$ *and in particular if $w_i = N$ for all i.*

Remark 1 (NFA approximation). *If $(F_i)_i$ is a* NFA, *then any family of functions $(G_i)_i$ verifying $F_i(t) \leqslant G_i(t)$ for all t is still a* NFA. *Hence, a function satisfying*

$$\mathrm{NFA}(i, x_i) \geqslant w_i \cdot \mathbb{P}(X_i \geqslant x_i)$$

will define a NFA *as soon as $\sum_{i=1}^{N} 1/w_i \leqslant 1$.*

3.2.2 Trajectory detection

We are given a sequence of K images, each containing N points (to ease the notations we consider a constant number of points throughout the sequence for now, but everything can be smoothly extended to the non-constant case as will be shown later), and whose support domain is taken to be the square $[0, 1] \times [0, 1]$ (again, the method can be adapted to arbitrary image sizes, as shown in Section 3.2.3). Following Helmholtz Principle, the naive model will here be a random uniform draw of N points in each of the K images (intuitively, we should not see trajectories appearing in the realizations of this model). The corresponding i.i.d. uniformly distributed random variables corresponding to the points of image k ($1 \leqslant k \leqslant K$) will be denoted by $X_1^k, ..., X_N^k$. We now define the structures of interest.

Definition 2 (Trajectories without holes). *A trajectory T of length ℓ starting at frame k_0 is a tuple $T = (k_0, i_1, ..., i_\ell)$, where $1 \leqslant i_p \leqslant N$ for all p and $1 \leqslant \ell \leqslant K - k_0 + 1$. We will denote by $\mathbb{T}$ the set of all trajectories.*

There is a natural equivalence between a trajectory $T \in \mathbb{T}$ and the tuple of variables $X_T = (X_{i_1}^{k_0}, ..., X_{i_\ell}^{k_0+\ell-1})$ that we shall therefore sometimes abusively call a (random) trajectory too.

A realization t of the random variable X_T will be called a *realization of the trajectory* T. We have to keep in mind that we are working with the naive model (where points are randomly distributed), and thus, a realization of X_T *should not* look like what we would intuitively call a trajectory.

The second ingredient of the a-contrario model is the measurement function. We could define a "good trajectory" as one that moves slowly, and take the measurement function to be, for instance, the maximal speed of the object along the trajectory. For most applications, a natural choice is to define a good trajectory as a smooth one, that is, a trajectory with a small acceleration. Of course, the model could easily be adapted to a variety of other measurements like the maximal velocity, the greatest direction change, etc.). Let us stick to the idea of good trajectories being those bearing a small acceleration. We still have many ways to define this, the two most obvious choices being to control the *maximal* acceleration, or to control the *average* acceleration of a trajectory realization $t = (\mathbf{y}_1, ..., \mathbf{y}_\ell)$. Again, the model can be adapted to both, but for practical reasons we will choose to control the maximal acceleration that involve simpler computations, thus leading to the measurement function

Definition 3 (Maximal acceleration). *The maximal acceleration of a tuple of points* $t = (\mathbf{y}_1, ..., \mathbf{y}_\ell)$ *is*

$$a_{\max}(t) = \max_{2 \leqslant i \leqslant \ell-1} \|\mathbf{y}_{i+1} - 2\mathbf{y}_i + \mathbf{y}_{i-1}\|. \tag{3.3}$$

We will now of course consider only trajectories having at least 3 points. Let T be a trajectory, Proposition 1 tells us that we can define a Number of False Alarms by an appropriate weighting of the probability $\mathbb{P}(a_{\max}(X_T) \leqslant \delta)$. In the naive model, this probability only depends on the length ℓ of the trajectory (and, of course, δ), and verifies

Proposition 2 (Probability bound). *If* X_T *is a random trajectory of length* ℓ, *and* δ *is a positive real number, then*

$$\mathbb{P}(a_{\max}(X_T) \leqslant \delta) \leqslant (\pi \cdot \delta^2)^{\ell-2}. \tag{3.4}$$

Proof — Assume that $T = (k_0, i_1, ..., i_\ell)$, and call $\bar{B}(x, r)$ the closed disc with center x and radius r. Writing $X'_p = X_{i_p}^{k_0+p-1}$, we get

$$\begin{aligned}
\mathbb{P}(a_{\max}(\quad X_T \quad) \leqslant \delta) & \\
&= \mathbb{P}\left(\bigcap_{p=3}^{\ell} \left\{X'_p \in \bar{B}(2X'_{p-1} - X'_{p-2}, \delta)\right\}\right) \\
&\leqslant \prod_{p=3}^{\ell} \mathbb{P}\left(X'_p \in \bar{B}(2X'_{p-1} - X'_{p-2}, \delta) \mid X'_{p-1}, X'_{p-2}\right) \\
&\leqslant (\pi \cdot \delta^2)^{\ell-2}
\end{aligned}$$

because the area of $\bar{B}(x, \delta) \cap [0, 1]^2$ is bounded from above by $\pi \cdot \delta^2$ for all x. □

By Remark 1, we know that we can use the upper bound (3.4) to construct a NFA, as in (3.2). There are many possibilities to define the weights $(w_T)_T$ subject to the constraint $\sum_T 1/w_T \leqslant 1$. This gives us a way to adjust the detection thresholds for each structure. We choose to group trajectories together according to their length, dividing the set of trajectories into K groups $\mathbb{T} = \mathbb{T}_1 \cup \ldots \cup \mathbb{T}_K$ (here, $\mathbb{T}_\ell$ denotes the set of trajectories of length ℓ), and weigh trajectories of a group uniformly by $w_T = K \cdot |\mathbb{T}_\ell|$ for any $T \in \mathbb{T}_\ell$.

Proposition 3 (Continuous NFA for trajectories without holes)**.** *The family of functions* $(\mathrm{NFA}_T)_{T\in\mathbb{T}}$ *defined by*

$$\forall \ell, \forall T \in \mathbb{T}_\ell, \quad \mathrm{NFA}_T(\delta) = K(K-\ell+1)N^\ell \cdot (\pi \cdot \delta^2)^{\ell-2} \tag{3.5}$$

is a Number of False Alarms for the measurement $a_{\max}$.

Proof — Since $|\mathbb{T}_\ell| = (K-\ell+1)N^\ell$, we have

$$\sum_{T\in\mathbb{T}} \frac{1}{K(K-\ell+1)N^\ell} = \sum_{\ell=1}^{K} \sum_{T\in\mathbb{T}_\ell} \frac{1}{K|\mathbb{T}_\ell|} = 1,$$

and we conclude that (3.5) defines a NFA thanks to Proposition 1. □

Let us quickly comment Proposition 3. We can rewrite (3.5) into $\mathrm{NFA}_T(\delta) = K(K-\ell+1)N^2 \cdot \alpha^{\ell-2}$ by using the relative density $\alpha = N\pi\delta^2$ (which corresponds to the average number of points falling in a disc with radius δ). We see that for a trajectory to be meaningful, we need to have $\alpha < 1$. In other terms, only trajectories with maximal acceleration $\delta < 1/\sqrt{N\pi}$ might be detected as meaningful. Such kinds of bounds will be analyzed more precisely in Section 3.2.6.

3.2.3 Data quantization

In many applications, point detection is realized on a discrete grid of integer pixel coordinates, so that it may happen that three successive points in the sequence have a null acceleration. This is a very strong contradiction to the naive model, since this event has probability zero. Thus, if a long trajectory has a subtrajectory with a null acceleration, an algorithm that detects the most meaningful trajectories first will cut the longer trajectory into chunks to isolate the (optimal) null-NFA subtrajectory.

To avoid this kind of behavior we need to handle data quantization carefully. There are two ways we could do this: assume that the data has been properly quantized on the integer grid of the image and define a discrete version of the NFA, or consider a measurement imprecision and always consider the "worst-case scenario" for the measurements when computing accelerations.

First, we assume that the data has been quantized on an integer grid, say a rectangle Ω of $\mathbb{Z}^2$ containing $|\Omega|$ pixels. We can define a discrete version of the NFA by replacing the continuous acceleration area $\pi \cdot \delta^2$ by its discrete equivalent. More precisely, we round the accelerations components δ_x, δ_y to the nearest integers $[\delta_x]$, $[\delta_y]$, and define the *discrete acceleration area* by

$$a^d(\delta_x, \delta_y) = \frac{|S_r|}{|\Omega|},$$

where $|S_r|$ is the number of pixels enclosed in the discrete disc $S_r = \mathbb{Z}^2 \cap \bar{B}(0,r)$ (see Fig. 34) and

$$r = \sqrt{[\delta_x]^2 + [\delta_y]^2}.$$

In particular, when $([\delta_x], [\delta_y]) = 0$ we obtain a discrete area of $1/|\Omega|$ that no longer leads to a null NFA.

Definition 4 (Discrete maximal acceleration)**.** *The discrete maximal acceleration of a tuple of points* $t = (\mathbf{y}_1, ..., \mathbf{y}_\ell)$ *is*

$$a^d_{\max}(t) = \max_{2 \leqslant i \leqslant \ell - 1} a^d(\mathbf{y}_{i+1} - 2\mathbf{y}_i + \mathbf{y}_{i-1}). \tag{3.6}$$

Proposition 4 (Discrete NFA for trajectories without holes)**.** *The family of functions* $(\mathrm{NFA}^d_T)_{T \in \mathbb{T}}$ *defined by*

$$\forall \ell, \forall T \in \mathbb{T}_\ell, \quad \mathrm{NFA}^d_T(a) = K(K - \ell + 1)N^\ell \cdot a^{\ell - 2} \tag{3.7}$$

is a Number of False Alarms for the measurement $a^d_{\max}$.

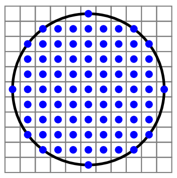


Figure 34: **Discrete discs.** A continuous disc and its corresponding discretization composed of all pixels whose centers lie inside the continuous disc. A discrete measure of area is better suited to the analysis of quantized data that might result in observing degenerate null-area continuous disc (the corresponding discrete disc has a non-null area).

We now examine the "worst-case scenario" acceleration. We assume that we have a measure $\eta > 0$ of the measurements imprecision (corresponding roughly to the radius of one pixel in the previous example). We keep the same NFA than in the continuous case (Equation 3.5), but we replace the measurement function by

$$a^w_{\max}(t) = \max_{2 \leqslant i \leqslant \ell - 1} a^w(\mathbf{y}_i, \mathbf{y}_{i+1}, \mathbf{y}_{i+1}), \quad \text{where}$$

$$a^w(\mathbf{x}, \mathbf{y}, \mathbf{z}) = \max_{dx, dy, dz \in \bar{B}(0, \eta)} \|(\mathbf{x} + dx) - 2(\mathbf{y} + dy) + (\mathbf{z} + dz)\|.$$

One easily shows that $a^w(\mathbf{x}, \mathbf{y}, \mathbf{z}) = \|\mathbf{x} - 2\mathbf{y} + \mathbf{z}\| + 4\eta$, and this therefore equivalently amounts to keeping the same measurement function than in the continuous case and replacing the NFA (3.5) with

$$\mathrm{NFA}^w_T(\delta) = K(K - \ell + 1)N^\ell \cdot (\pi \cdot (\delta + 4\eta)^2)^{\ell - 2} \tag{3.8}$$

We see that a null-acceleration trajectory will be counted as an acceleration of 4η, thus incurring a penalty to all accelerations. This is why in practice we use the discrete NFA.

3.2.4 Algorithm

In this section we consider the discrete NFA given in Equation (3.7). When a meaningful trajectory is present, any slight deviation from it (removing or adding a point, for instance) will usually also be meaningful, so that we expect to detect a large number of overlapping meaningful trajectories. Hence, we choose to detect the trajectories greedily, by iterating the following process:

1. compute the most meaningful trajectory, that is, the one having the smallest NFA;
2. remove its points from the data.

To compute the most meaningful trajectory in a sequence of points, that is, K images $I_1, \dots, I_K$, each containing N points, we use a dynamic programming strategy. Indeed, we compute for each point $\mathbf{x}$ in image I_k the most meaningful trajectory ending in this point (note that in the following, we shall sometimes write $\mathbf{x}^k$ instead of $\mathbf{x}$ to recall that $\mathbf{x}$ belongs to I_k). Denoting by $\mathcal{G}(\mathbf{x}^k, \mathbf{y}^{k-1}, \ell)$ the minimal acceleration of a trajectory of length ℓ ending with points $\mathbf{y}^{k-1}$ and $\mathbf{x}^k$, we obtain a Bellman equation

$$\mathcal{G}(\mathbf{x}^k, \mathbf{y}^{k-1}, \ell) = \begin{cases} 0 & \text{if } \ell = 2, \\ \min_{\mathbf{z} \in I_{k-2}} \overline{\mathcal{G}}(\mathbf{x}, \mathbf{y}, \mathbf{z}, \ell) & \text{otherwise,} \end{cases} \tag{3.9}$$

where

$$\overline{\mathcal{G}}(\mathbf{x}, \mathbf{y}, \mathbf{z}, \ell) = \max\Big(a^d(\mathbf{x} - 2\mathbf{y} + \mathbf{z}) \ , \ \mathcal{G}(\mathbf{y}, \mathbf{z}, \ell - 1) \Big) . \tag{3.10}$$

This recursive formulation translates to Algorithm 3.

```
Algorithm: compute_$\mathcal{G}$
input  : $f_1, \dots, f_K$ the sets of $\{n_k \leqslant N\}_{1 \leqslant k \leqslant K}$ points contained in each frame
output: $\mathcal{G}$
for $2 \leqslant k \leqslant K$ do
    for $x$ in $f_k$ do
        for $y$ in $f_{k-1}$ do
            $\mathcal{G}(x, y, 2) \leftarrow a^d(0)$
            for $3 \leqslant \ell \leqslant k$ do
                $\mathcal{G}(x, y, \ell) \leftarrow +\infty$
                for $z$ in $f_{k-2}$ do
                    $a \leftarrow \max(a^d(x - 2y + z), \mathcal{G}(y, z, \ell - 1))$
                    $\mathcal{G}(x, y, \ell) \leftarrow \min(a, \mathcal{G}(x, y, \ell))$
                end
            end
        end
    end
end
return $\mathcal{G}$
```

Algorithm 3: Dynamic programming computation of $\mathcal{G}$. We start by computing the values of $\mathcal{G}(x^k, y^{k-1}, \ell)$ for $k = 2$, then $k = 3$, ... each time reusing the results of the previous round.

Now let us write

$$\mathrm{NFA}^d_\ell(a) = K(K - \ell + 1)N^\ell \cdot a^{\ell - 2}, \tag{3.11}$$

so that $\mathrm{NFA}^d_T(a) = \mathrm{NFA}^d_\ell(a)$ for any trajectory T with length ℓ. Since the function $a \mapsto \mathrm{NFA}^d_\ell(a)$ is monotone, the most meaningful trajectory with length ℓ is the one having the least maximal acceleration. Hence we can use Algorithm 3 to compute the smallest NFA of the point sequence (Algorithm 4). Moreover, if $\mathcal{B}(\mathbf{x}^k, \mathbf{y}^{k-1}, \ell)$ represents the most meaningful trajectory with length ℓ ending with points $\mathbf{y} \to \mathbf{x}$ (where $t \to \mathbf{x}$ denotes the concatenation of trajectory t and point $\mathbf{x}$), we can write

$$\mathcal{B}(\mathbf{x}^k, \mathbf{y}^{k-1}, \ell) = \begin{cases} \mathbf{y} \to \mathbf{x} & \text{if } \ell = 2, \\ \mathcal{B}(\mathbf{y}, \hat{\mathbf{z}}, \ell - 1) \to \mathbf{x} & \text{otherwise,} \end{cases} \tag{3.12}$$

where $\hat{\mathbf{z}} \in \arg\min_{\mathbf{z}} \mathcal{G}(\mathbf{y}, \mathbf{z}, \ell - 1)$ (strictly speaking, the most meaningful trajectory might not be unique, so we have to choose one most meaningful trajectory arbitrarily; to simplify the description we shall nevertheless use the term "the most meaningful trajectory" throughout the paper). Finally, for each $\mathbf{x}^k$, the most meaningful trajectory ending in $\mathbf{x}^k$ is $\mathcal{B}(\mathbf{x}^k, \hat{\mathbf{y}}^{k-1}, \hat{\ell})$ where

$$(\hat{\mathbf{y}}^{k-1}, \hat{\ell}) \in \arg \min_{(\mathbf{y}^{k-1}, \ell)} \mathrm{NFA}^d_\ell \left(\mathcal{G}(\mathbf{x}^k, \mathbf{y}^{k-1}, \ell) \right).$$

An algorithm similar to Algorithm 3 can hence be used to compute a trajectory having the smallest NFA. In practice, we choose an arbitrary tuple $(\mathbf{x}, \mathbf{y}, \ell)$ such that there is an optimal trajectory of length ℓ ending on points $\mathbf{y} \to \mathbf{x}$, and we extract a trajectory by backtracking, each time selecting the predecessor that minimizes locally the maximal acceleration among all predecessors that lead to an optimal trajectory.

```
Algorithm: minimal_NFA
input  : $\mathcal{G}$
output: $m$ the minimal NFA of a trajectory
$m \leftarrow +\infty$
for $3 \leqslant k \leqslant K$ do
    for $x$ in $f_k$ do
        for $y$ in $f_{k-1}$ do
            for $3 \leqslant \ell \leqslant k$ do
                $m \leftarrow \min(m, \mathrm{NFA}^d_\ell(\mathcal{G}(x, y, \ell)))$
            end
        end
    end
end
return $m$
```

Algorithm 4: Find the minimal NFA value among all trajectories.

Note that we could also extract the trajectory having the minimal average acceleration (for example) among all optimal trajectories, simply by rerunning a similar dynamic-programming algorithm restricted to optimal trajectories.

In the end, by applying the process greedily (as mentioned above), we obtain an algorithm that extracts all meaningful (or ε-meaningful) trajectories from a sequence of points (Algorithm 5).

Often we can save some computation time by extracting several trajectories at once without recomputing the function $\mathcal{G}$ each time, because removing points from the current data set cannot decrease any value of $\mathcal{G}$. This idea can be used, for instance, when the two most meaningful trajectories in the sequence do not share any point. The inner loop of Algorithm 6 does just that: if we have a way to define a set of disjoint minimal-NFA trajectories (for instance,

```
Algorithm: trajectory_detection
input  : $\varepsilon$ the maximal allowed NFA
         $f_1, ..., f_K$ the sets of points contained in each frame
output: $S = t_1, ..., t_m$ the extracted trajectories
$S \leftarrow \emptyset$
repeat
    compute_$\mathcal{G}$
    $m \leftarrow$ minimal_NFA
    if $m \leqslant \varepsilon$ then
        $t \leftarrow$ a trajectory of NFA $= m$
        $S \leftarrow S \cup \{t\}$, and remove all points in $t$ from the corresponding frames
    end
until $m > \varepsilon$ or there are no more points
return S
```

Algorithm 5: Greedy trajectories extraction using the NFA criterion.

greedily), we can extract them all at once (since they are of minimal NFA, and each are non-overlapping, we could extract them sequentially with interleaving $\mathcal{G}$ function recomputations, but this would not change the NFA of those trajectories, that would still be minimal).

Then if we have been able to extract a set of disjoint trajectories of minimal NFA that covers *every* point where a trajectory of minimal NFA could end, we can continue doing the extraction for the next minimal NFA without recomputation. When this is no longer possible (because of some point removal) we need to recompute the $\mathcal{G}$ function to reactualize the NFAs.

We now examine the space and time complexity of algorithm 5 for an extraction round. The most expensive computation is that of function $\mathcal{G}$. The space (memory) required is $\mathcal{O}(N^2K^2)$, since we have to store a value for each triplet $(\mathbf{x}^k, \mathbf{y}^{k-1}, \ell)$ in each image frame k. Each value computation takes $\mathcal{O}(N)$ operations because we have to consider all the points in the previous image, leading to a $\mathcal{O}(N^3K^2)$ time complexity. The search for the minimal NFA and the extraction of the most meaningful trajectory have negligible time and space complexities. As the extraction must be repeated as long as there is any remaining meaningful trajectory, the global time complexity is $\mathcal{O}(sN^3K^2)$, where s denotes the number of extracted ε-meaningful trajectories. In practice, on a standard PC desktop, for $K = 50$ images, the number N of points per image can go up to several hundreds (and about one thousand for $K = 20$).

3.2.5 Variable number of points

In real data, the number of points is hardly ever constant throughout the sequence, so that instead of having N points in each of the K images, we have $N_1, N_2, \dots N_K$ points on images $1, 2, \dots K$. Since the NFA is an upper bound on the average number of false alarms (Remark 1) we can simply take $N = \max_k N_k$ and keep the NFA unchanged. However, to obtain more accurate results, we can refine Proposition 4 with

Proposition 5 (Discrete NFA for trajectories without holes, general case)**.** *The family of functions* $(\text{NFA}^d_T)_{T \in \mathbb{T}}$ *defined, for any trajectory T with length* ℓ *starting at frame* k_0*, by*

$$\text{NFA}^d_T(a) = K(K - \ell + 1) \left(\prod_{k_0 \leqslant k \leqslant k_0 + \ell - 1} N_k \right) \cdot a^{\ell - 2}, \tag{3.13}$$

```
Algorithm: trajectory_detection_accelerated
input  : $\varepsilon$ the maximal allowed NFA
         $f_1, ..., f_K$ the sets of points contained in each frame
output: $S = t_1, ..., t_m$ the extracted trajectories
$S \leftarrow \varnothing$
repeat
    compute_$\mathcal{G}$
    $m \leftarrow$ minimal_NFA
    stop $\leftarrow$ false
    while $m \leqslant \varepsilon$ and stop = false do
        $U \leftarrow \{ x \mid \exists$ a traj. with NFA $= m$ ending in point $x \}$
        $V \leftarrow$ a set of disjoint trajectories of NFA $= m$ ending on a point of $U$
        $S \leftarrow S \cup V$
        remove all points from the trajectories in $V$
        stop $\leftarrow$ true if not all points of $U$ have been removed
        $m \leftarrow$ minimal_NFA()
    end
until $m > \varepsilon$ or there are no more points
return S
```

Algorithm 6: Greedy trajectories extraction, accelerated by extracting several trajectories at once.

is a Number of False Alarms for the measurement $a^d_{\max}$.

The proof is very similar to that of Proposition 3, except that now the set of trajectories $\mathbb{T}_\ell$ itself has to be decomposed with respect to the index of the starting frame.

3.2.6 Theoretical analysis

We now examine some theoretical consequences of Equation (3.5), that can have very practical consequences in the design of the data acquisition process. For simplicity reasons, we use the continuous NFA formulation, with a fixed number of points per image, but (3.7) or (3.13) would lead to the same conclusions.

Relation between the number of points and the maximal acceleration

Consider, as before, a sequence of K frames, each containing N points. We recall that the Number of False Alarms associated to a trajectory T with length $\ell \geqslant 3$ and maximal acceleration δ is (see Equation 3.5)

$$\mathrm{NFA}_T(\delta) = K(K-\ell+1)N^\ell \cdot (\pi \cdot \delta^2)^{\ell-2}.$$

Such trajectory is ε-meaningful as soon as

$$K(K-\ell+1)N^\ell \cdot (\pi \cdot \delta^2)^{\ell-2} \leqslant \varepsilon,$$

which can be rewritten $\delta \leqslant \delta_c$, where the upper bound is the critical acceleration

$$\delta_c = \frac{1}{\sqrt{N\pi}} \left(\frac{\varepsilon}{K(K-\ell+1)N^2} \right)^{\frac{1}{2\ell-4}}. \tag{3.14}$$

Hence, as we already remarked at the end of Section 3.2.2, a necessary condition for trajectory detection is $\delta \leqslant \Delta$ with $\Delta = \frac{1}{\sqrt{N\pi}}$, which gives an order of magnitude of the typical accelerations that can be handled by the NFA approach (and, in some sense, by any approach since accelerations greater than Δ would allow detections in pure noise). Since the acceleration is inversely proportional to the squared frame rate (by doubling the frame rate, one divides accelerations by 4), this absolute bound can be useful in the design of the data acquisition process. Indeed, given the expected number of detected points in each frame (N), and the expected physical accelerations of objects (δ), one can compute the critical frame rate, under which no trajectory detection is possible. Note, however, that the upper bound Δ is not very accurate (see Table 2), thus using the exact value δ_c (Equation 3.14) is probably a better idea.

N	15	50	200	1000
Δ	146	80	40	18
δ_c ($l = 5$, $K = 20$)	23	8.3	2.6	0.7
δ_c ($l = 10$, $K = 20$)	74	35	15	5.4
δ_c ($l = 10$, $K = 50$)	64	30	13	4.7
δ_c ($l = 30$, $K = 50$)	117	61	29	12

Table 2: Acceleration bounds $\Delta = \frac{1}{\sqrt{N\pi}}$ and δ_c (Equation 3.14) in function of N, expressed in pixel.image^{-2} in a 1000×1000 image for some values of l and K ($\varepsilon = 1$).

Influence of the trajectory length

A nice property of the a-contrario approach is that it permits to relate different parameters by observing the way they are linked in the NFA formula. Table 2 shows that the trajectory length has a significant impact on the critical acceleration δ_c (whereas the number of frames, K, has a much smaller impact). Thus, it could be interesting to study more precisely how the NFA balances the trajectory length and the acceleration, that is, how the critical acceleration grows as the trajectory length increases. Since $1 \leqslant K - \ell + 1 \leqslant K$, we can write $\log(K(K - \ell + 1)) = 2\beta(\ell) \log K$ with $\beta(\ell) \in [1/2, 1]$, so that from (3.14) we get

$$\log \delta_c = \log \Delta - \frac{\log N + \beta(\ell) \log K - \log \sqrt{\varepsilon}}{\ell - 2}.$$

Hence, $\log \delta_c$ grows approximately like $1/\ell$, and attains for $\ell = K$ a value close to (and below) $\log \Delta$. This is illustrated on Fig. 35.

Last, we show the monotony of the critical acceleration with respect to the trajectory length (that is, the longer the trajectory, the looser the constraint on the acceleration).

Proposition 6. *If $\varepsilon \leqslant 1$, then the critical acceleration δ_c given by (3.14) increases with respect to ℓ.*

Proof — Rather than using (3.14), we go back to (3.5) and write, for $\ell \in \{1, \dots, K\}$,

$$F(\ell, \delta) = N^2 K(K - \ell + 1) \cdot (N\pi\delta^2)^{\ell - 2}$$

(thus, $F(\ell, \delta)$ is the NFA associated to a trajectory with size ℓ and maximal acceleration δ). Now, if ℓ is such that $F(\ell, \delta)$ is smaller than 1 (since we suppose that $\varepsilon \leqslant 1$), then $N\pi\delta^2 < 1$ so that both $\ell \mapsto (N\pi\delta^2)^{\ell-2}$ and $\ell \mapsto N^2 K(K - \ell + 1)$ are decreasing with respect to ℓ, and so is $\ell \mapsto F(\ell, \delta)$. Hence, if $\ell_1 > \ell_2$, we have, for $\delta = \delta_c(\ell_2)$,

$$F(\ell_1, \delta_c(\ell_2)) < F(\ell_2, \delta_c(\ell_2)) = \varepsilon,$$

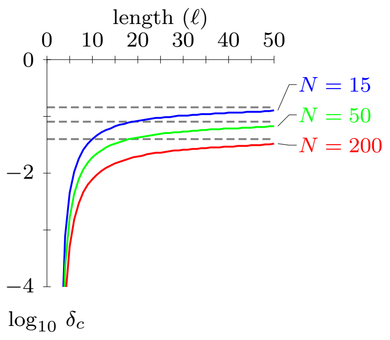


Figure 35: **Asymptotic and non-asymptotic critical accelerations.** The (base-10 log) critical acceleration $\log_{10}\delta_c$ is an increasing function of ℓ that approaches its upper bound (dotted line) $\log_{10}\Delta = \log_{10}\frac{1}{\sqrt{N\pi}}$ when $\ell = K$. Here $K = 50$ and the three curves correspond to $N = 15$, $N = 50$, and $N = 200$ respectively.

which proves that $\delta_c(\ell_1) > \delta_c(\ell_2)$ since $\delta \mapsto F(\ell_1, \delta)$ is increasing. □

Asymptotic bounds and the importance of the combinatorial factor

Now we would like to assess the importance of the combinatorial factor in the definition of the NFA. As was discussed above, there are several ways to define the weights of the structures. In (3.5), we chose to weigh the trajectories uniformly with respect to their length (that is, such that the expected number of false alarms is equally shared among all possible trajectory lengths), that is

$$\mathrm{NFA}_T(\delta) = K(K-\ell+1)N^{\ell} \cdot \mathbb{P}(a_{\max}(X_T) \leqslant \delta). \tag{3.15}$$

Another more classical choice would be to set a uniform weight $w_T = |\mathbb{T}|$ for *all* trajectories, thus obtaining

$$\mathrm{NFA}'_T(\delta) = \left(\sum_{m=1}^{K}(K-m+1)N^m\right) \cdot \mathbb{P}(a_{\max}(X_T) \leqslant \delta). \tag{3.16}$$

Suppose that we observe a trajectory t with length ℓ and maximal acceleration $\delta = a_{\max}(t)$, what difference will each NFA definition make? This trajectory is detected if $\mathrm{NFA}_T(\delta)$ (or $\mathrm{NFA}'_T(\delta)$) is below a certain threshold ε, hence it is interesting to estimate the ratio

$$\begin{aligned}
\frac{\mathrm{NFA}'_T(\delta)}{\mathrm{NFA}_T(\delta)} &= \frac{1}{K(K-\ell+1)N^{\ell}}\sum_{m=1}^{K}(K-m+1)N^m \\
&\geqslant \frac{1}{K^2N^{\ell}}\sum_{m=\ell}^{K}N^m \\
&\geqslant \frac{N^{K-\ell}}{K^2}.
\end{aligned}$$

This lower bound shows that when ℓ is small, the detection penalty incurred when using NFA$'$ is very large and it will thus be more difficult to detect small trajectories with this criterion.

In the following, we compare more precisely NFA and NFA$'$ and compute asymptotic estimates when the number of frames (K) becomes large. Let us deal first with the function $\mathrm{NFA}_T(\delta)$. We consider a trajectory T spanning $|T| = \mu K$ images of the sequence for a fixed $\mu \in (0, 1]$, with a maximal acceleration δ, among N points per frame. We write $\alpha = N\pi\delta^2$, and propose to determine, when K gets very large, if the trajectory is meaningful. We have

$$\begin{aligned}\mathrm{NFA}_T(\delta) &= K \cdot (K - \mu K + 1) \cdot N^{\mu K} \cdot (\pi\delta^2)^{\mu K - 2} \\ &\underset{K\to\infty}{\sim} \pi^{-2}\delta^{-4}K^2(1-\mu)\alpha^{\mu K},\end{aligned}$$

so that

$$\log \mathrm{NFA}_T(\delta) \underset{K\to\infty}{\sim} \mu K \log \alpha$$

and

$$\lim_{K\to+\infty} \mathrm{NFA}_T(\delta) \leqslant 1 \quad \Longleftrightarrow \quad \alpha < 1 \tag{3.17}$$

This means that for any maximal acceleration δ such that $\delta < \Delta$ (that is, $\alpha < 1$), all trajectories spanning μK frames ($0 < \mu \leqslant 1$) will eventually become meaningful when K is large enough. In practice, values of α near 1 are not very efficient, since they require a very large value of K to lead to meaningful trajectories. This phenomenon is illustrated on Fig. 36.

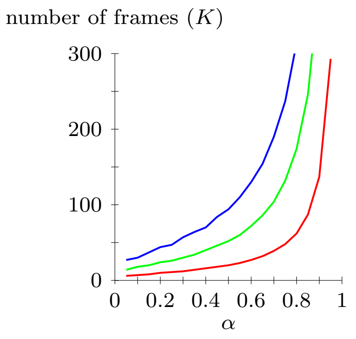


Figure 36: Non-asymptotic counterpart of (3.17). Theses three curves represent, as a function of $\alpha = N\pi\delta^2$, the minimum number of frames (K) required for a trajectory with length $\ell = \lfloor \mu K \rfloor$ and maximal acceleration δ to be 1-meaningful according to the NFA criterion (3.15). Upper curve: $\mu = 0.3$; middle curve: $\mu = 0.5$; lower curve: $\mu = 1$. The number of points per frame is $N = 100$.

Now we study the asymptotic behavior of $\mathrm{NFA}'_T(\delta)$ (see Equation 3.16). First, we notice that

$$\sum_{m=1}^{K} (K - m + 1)N^m = N^{K+2} \sum_{m=1}^{K} mN^{-m-1} \underset{K\to\infty}{\sim} \frac{N^{K+2}}{(N-1)^2}$$

and thus

$$\mathrm{NFA}'_T(\delta) \underset{K\to\infty}{\sim} \frac{N^{K+2-\mu K}\alpha^{\mu K}}{(N-1)^2}.$$

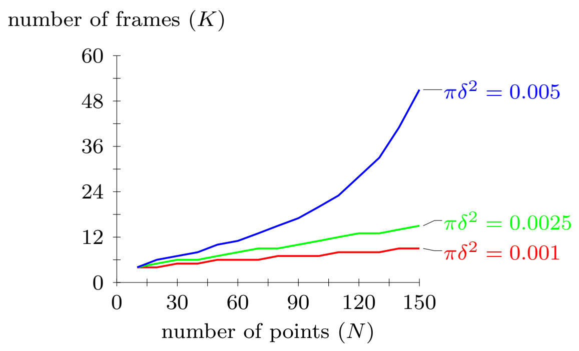


Figure 37: Non-asymptotic counterpart of (3.17). Theses three curves represent, as a function of the number of points N, the minimum number of frames (K) required for a trajectory with length $\ell = K$ and various values of the maximal acceleration δ to be 1-meaningful according to the NFA criterion (3.15).

Therefore,

$$\log \mathrm{NFA}'_T(\delta) \underset{K\to\infty}{\sim} (1-\mu)K\log N + \mu K \log \alpha$$

and

$$\lim_{K\to+\infty} \mathrm{NFA}'_T(\delta) < 1 \iff \log\alpha < \frac{\mu-1}{\mu}\log N. \tag{3.18}$$

Since $\frac{\mu-1}{\mu} \to -\infty$ as $\mu \to 0^+$, Equation (3.18) shows that it is indeed asymptotically much harder to detect small trajectories with NFA$'$ than with NFA. This fact is illustrated by Fig. 38, on which we can see that even when μ is close to 1, NFA permits to detect much more trajectories than NFA$'$, both asymptotically and non-asymptotically.

3.3 TRAJECTORIES WITH HOLES

In many practical situations, because of occlusions or acquisition noise, some trajectory points will not be detected in one or more frames. In this section, we generalize the previous framework to trajectory detection in the case of missing points.

3.3.1 Number of false alarms

The naive model remains unchanged (keeping the notations of the previous section): we are given K images $I_1, ..., I_K$, each image I_k containing N points $X_1^k, ..., X_N^k$, and we assume by Helmholtz principle that all random points X_i^k are independent and uniformly distributed in $[0,1]^2$.

Definition 5 (Trajectories with holes). *A* trajectory *of size s is a sequence of pairs $T = \{(i_1, \tau_1), ..., (i_s, \tau_s)\}$, such that $\tau_1 < ... < \tau_s$. We denote by $\mathbb{T}$ the set of all trajectories, and by $\mathbb{T}_s$ the set of trajectories of size s. We bijectively associate to the trajectory T the tuple of random (i.i.d., uniformly distributed) variables $X_T = (X_{i_1}^{\tau_1}, ..., X_{i_s}^{\tau_s})$, that we also abusively call a (random) trajectory.*

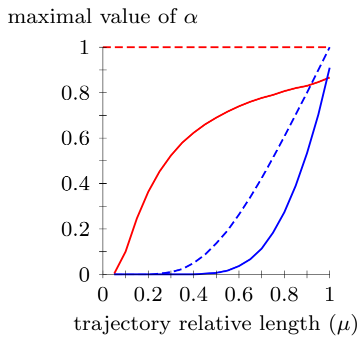


Figure 38: **Comparison between the** NFA **and** NFA$'$ **models (3.15, 3.16).** Each curve represents, as a function of μ, the minimal value of $\alpha = N\pi\delta^2$ required for a trajectory with length $\ell = \lfloor \mu K \rfloor$ and maximal acceleration δ to be 1-meaningful in a sequence of K images, among $N = 100$ points per frame. The red upper curves are obtained with the criterion NFA, whereas the blue lower curves are obtained with the criterion NFA$'$. The full curves correspond to $K = 100$, and the dashed curves correspond to the asymptotic estimates obtained when $K \to +\infty$, that is, $\log \alpha = 0$ for NFA and $\log \alpha = \log N \cdot (\mu - 1)/\mu$ for NFA$'$. These curves clearly demonstrate that not only NFA is better suited to the detection of small trajectories than NFA$'$ (it allows for trajectories having a much larger maximal acceleration), but it is also more efficient even for relatively large values of μ (NFA$'$ being slightly better only for almost complete trajectories). Asymptotically, NFA is always the best choice.

Since the definition above is more general than the special case of trajectories without holes (Definition 2), we chose to keep the same word (trajectory) and the same notations ($\mathbb{T}$, $\mathbb{T}_s$) as in Section 3.2. This should not lead to ambiguities, since we will only consider trajectory with holes in this section.

As in Section 3.2, we would like to build an a-contrario detection model to detect trajectories (here, with holes). We consider three parameters of interest for the computation of the NFA of a trajectory $X_T = (X_{i_1}^{\tau_1}, \ldots, X_{i_s}^{\tau_s})$: the trajectory length, its size and its number of runs. The *length* is the total number of frames that the trajectory spans ($\tau_s - \tau_1 + 1$), the *size* is the number of (detected) points it contains (s), and a *run* is a maximal set of consecutive points. Note that if we call *hole* a maximal set of consecutive missing points, then the number of runs equals the number of holes plus one.

We first need to generalize the notion of maximal acceleration a_{max} (Definition 3) to the case of trajectories with holes. A natural way to do this consists in interpolating the missing points of the trajectory and compute its maximal acceleration. Since we would like to keep using an algorithm based on dynamic programming, we use the most local choice, that is, a simple constant speed interpolation. This leads to the following

Definition 6 (Maximal acceleration with holes)**.** *The maximal acceleration of the realization* $t = (\mathbf{y}_1^{\tau_1}, \ldots, \mathbf{y}_s^{\tau_s})$ *of a trajectory* T *is*

$$a_{max}^h(t) = \max_{2 \leqslant i \leqslant s-1} \| a^h(\mathbf{y}_{i-1}^{\tau_{i-1}}, \mathbf{y}_i^{\tau_i}, \mathbf{y}_{i+1}^{\tau_{i+1}}) \|, \tag{3.19}$$

with, for all points $\mathbf{x}^i, \mathbf{y}^j, \mathbf{z}^k$ *($i < j < k$),*

$$a^h(\mathbf{x}^i, \mathbf{y}^j, \mathbf{z}^k) = \frac{\mathbf{z} - \mathbf{y}}{k - j} - \frac{\mathbf{y} - \mathbf{x}}{j - i}. \tag{3.20}$$

We now compute, as in Proposition 2, a probability bound for the maximal acceleration of a random trajectory with holes.

Proposition 7 (Simple probability bound). *If a random trajectory* X_T *with size* s *has holes of size* $h_1, \dots, h_{p-1}$*, then for any* $\delta > 0$ *one has*

$$\mathbb{P}(a^h_{\max}(X_T) \leqslant \delta) \leqslant (\pi \cdot \delta^2)^{s-2} \cdot \prod_{1 \leqslant i \leqslant p-1} (h_i + 1)^2. \tag{3.21}$$

Proof — We assume that $T = \{(i_1, \tau_1), \dots, (i_s, \tau_s)\}$, and write $X'_q = X^{\tau_q}_{i_q}$ and

$$M_q = X'_{q-1} + \frac{\tau_q - \tau_{q-1}}{\tau_{q-1} - \tau_{q-2}} (X'_{q-1} - X'_{q-2}),$$

so that

$$\begin{aligned}
\mathbb{P}(a_{\max}(\quad X_T \quad) \leqslant \delta) & \\
\leqslant \quad & \mathbb{P}\left(\bigcap_{q=3}^{s} \left\{ X'_q \in \bar{B}(M_q, (\tau_q - \tau_{q-1})\delta) \right\} \right) \\
\leqslant \quad & \prod_{q=3}^{s} \mathbb{P}\left(X'_q \in \bar{B}(M_q, (\tau_q - \tau_{q-1})\delta) \mid X'_{q-1}, X'_{q-2} \right) \\
\leqslant \quad & (\pi \cdot \delta^2)^{s-2} \prod_{q=3}^{s} (\tau_q - \tau_{q-1})^2 \\
\leqslant \quad & (\pi \cdot \delta^2)^{s-2} \prod_{i=1}^{p-1} (h_i + 1)^2.
\end{aligned}$$

□

For efficiency reasons, we want to design an algorithm that can share computations, that is, we want to be able to reuse the computations made on subtrajectories and extend them to obtain the results for bigger trajectories. To do this efficiently, we shall not consider, for a given trajectory, the individual sizes h_i of its holes, but simply its length ℓ, its size s and its number of runs p. This is why we derive from (3.21) the following

Proposition 8 (Practical probability bound). *If a random trajectory* X_T *has length* ℓ*, size* s *and number of runs* p*, then for any* $\delta > 0$ *one has*

$$\mathbb{P}(a^h_{\max}(X_T) \leqslant \delta) \leqslant (\pi \cdot \delta^2)^{s-2} \cdot \left(\frac{\ell - s}{p - 1} + 1 \right)^{2p-2} \tag{3.22}$$

with the convention that the right-hand parenthesis equals 1 ($(\frac{0}{0} + 1)^0$) *when* $p = 1$.

Proof — We consider the maximum value of the right-hand term of (3.21) over all possible hole sizes $h_1, \dots, h_{p-1}$ that are feasible for parameters ℓ, s and p. Relaxing the constraint that the h_i have to be integers, we face the optimization problem

$$\max_{(h_i)} \prod_{i=1}^{p-1} (h_i + 1)^2 \quad ; \quad \sum_i h_i = \ell - s, \text{ and } \forall i, \;\; h_i \geqslant 0,$$

which, denoting $\xi_i = h_i + 1$, has the same solutions as the problem $\max_{\xi \in C} E(\xi)$, with $E(\xi) = \sum_{i=1}^{p-1} \log(\xi_i)$ and

$$C = \left\{ \xi \in [1, +\infty)^{p-1}, \sum_{i=1}^{p-1} \xi_i = \ell - s + p - 1 \right\}.$$

Now if $\xi \in C$ has not identical coordinates, we can choose two different values, say $1 \leqslant \xi_{i_1} < \xi_{i_2}$, and replace them both by $(\xi_{i_1} + \xi_{i_2})/2 \geqslant 1$ to form a new $\xi' \in C$. Then,

$$E(\xi') - E(\xi) = 2\log((\xi_{i_1} + \xi_{i_2})/2) - \log(\xi_{i_1}) - \log(\xi_{i_2}),$$

and this quantity is positive by strict concavity of the log function. Thus, the unique solution of $\max_{\xi \in C} E(\xi)$ satisfies $\xi_i = (\ell - s)/(p-1) + 1$ for all i, and the maximum value of $\prod_i (h_i + 1)^2$ over feasible hole sizes $h_1, ..., h_{p-1}$ is bounded from above by

$$\left(\frac{\ell - s}{p-1} + 1 \right)^{2p-2}.$$

Using this bound in (3.21) yields the announced result. $\square$

Now we need to define the combinatorial term w_T that premultiplies the NFA. Recalling the discussion of the end of Section 3.2.2, we choose to group the trajectories by their length and size, and to use uniform weights in each category. The number of trajectories of length ℓ and size s that can fit in K frames is bounded from above by $(K - \ell + 1)\binom{\ell}{s}N^s$, and since we cluster the trajectories by their lengths and sizes, we have to count the number of such clusters. Indeed, it is bounded from above by $K\ell$, since there are less than K ways to choose the length, and knowing that the length is ℓ there are less than ℓ ways to choose the size. Combining these remarks with Proposition 8 establishes the following

Proposition 9 (NFA for trajectories with holes)**.** *The family of functions* $(\mathrm{NFA}_T)_{T \in \mathbb{T}}$ *defined for any trajectory* T *of length* ℓ*, size* s *and number of runs* p *by*

$$\mathrm{NFA}_T(\delta) = K\ell(K - \ell + 1)\binom{\ell}{s} N^s (\pi\delta^2)^{s-2} \left(\frac{\ell - s}{p-1} + 1 \right)^{2p-2} \tag{3.23}$$

is a Number of False Alarms for the measurement $a^h_{\max}$.

This new function NFA_T is a kind of generalization of (3.5). Indeed, for a trajectory T without hole (that is, such that $p = 1$, and consequently $\ell = s$), we have

$$\mathrm{NFA}_T(\delta) = \ell \cdot K(K - \ell + 1)N^\ell(\pi\delta^2)^{\ell-2}$$

which is, up the a new factor ℓ, the value given in (3.5). This new factor simply comes from the fact that we do not know *a priori* that the number of runs of the trajectory is one.

3.3.2 Algorithm

In the practical implementation that we describe below, we use a discrete version of the acceleration, obtained by replacing the norm involved in (3.19) by a discrete area measure, exactly as we did in Section 3.2 (see Definition 4).

We want to compute, for each point $\mathbf{x}$ of each image I_i (that we denote by $\mathbf{x}^i$), the most meaningful trajectory $T = \mathcal{N}(\mathbf{x}^i)$ that ends in $\mathbf{x}^i$ (or, to be more precise, one of such most meaningful trajectories) . This information can be extracted from the function $\mathcal{G}(\mathbf{x}^i, \mathbf{y}^j, \ell, s, p)$, which represents the least maximal acceleration of a trajectory of length ℓ, size s, and having p consecutive runs (that is, $p-1$ holes), ending with the point $\mathbf{y}^j$ in frame $j < i$ followed by the point $\mathbf{x}^i$ in frame i.

We say that a tuple (i, j, ℓ, s, p) is *undefined* if there is no trajectory ending with its two last points in frame i and j, with length ℓ, size s and having p runs of consecutive points. For instance, if $\ell < s$ or $s < 2$, the tuple is undefined. We define for $i > j$

$$\mathcal{G}(\mathbf{x}^i, \mathbf{y}^j, \ell, s, p) = \begin{cases} +\infty & \text{if } (i, j, \ell, s, p) \text{ is undefined,} \\ 0 & \text{if } s = 2, \\ \min_{u \geqslant 1, \mathbf{z} \in I_{j-u}} \bar{\mathcal{G}}(\mathbf{x}, \mathbf{y}, \mathbf{z}, \ell, s, p) & \text{otherwise,} \end{cases} \tag{3.24}$$

with the convention, for $i > j > k$, that

$$\bar{\mathcal{G}}(\mathbf{x}^i, \mathbf{y}^j, \mathbf{z}^k, \ell, s, p) = \max\left(a^h(\mathbf{x}, \mathbf{y}, \mathbf{z}),\ \mathcal{G}(\mathbf{y}, \mathbf{z}, \ell - (i-j), s-1, p - \mathbb{1}_{i \neq j+1})\right) \tag{3.25}$$

and as usual $\mathbb{1}_{a \neq b} = 1$ if $a \neq b$ and 0 otherwise. Notice that as in Section 3.2.4, the superscript k in $\mathbf{x}^k$ simply reminds us that the point $\mathbf{x}$ belongs to image I_k, so we sometimes omit it and simply write $\mathbf{x}$.

We deduce from (3.25) a dynamic programming algorithm to compute $\mathcal{G}$, similar to the one we presented in Section 3.2 for the trajectories without holes. We can then backtrack to find the most meaningful trajectory ending in each point $\mathbf{x}^i$ by defining, for $i > j$, the recursive function

$$\mathcal{B}(\mathbf{x}^i, \mathbf{y}^j, \ell, s, p) = \begin{cases} \text{undefined} & \text{if } (i, j, \ell, s, p) \text{ is undefined,} \\ \mathbf{y} \to \mathbf{x} & \text{if } s = 2, \\ \mathcal{B}(\mathbf{y}, \hat{\mathbf{z}}^{j-\hat{u}}, \ell - (i-j), s-1, p - \mathbb{1}_{i \neq j+1}) \to \mathbf{x} & \text{otherwise,} \end{cases}$$

where $\hat{\mathbf{z}}^{j-\hat{u}}$ realizes the minimum in the last line of (3.24). Finally, for each $\mathbf{x}^i$, the most meaningful trajectory with holes ending in $\mathbf{x}^i$ is $\mathcal{B}(\mathbf{x}^i, \hat{\mathbf{y}}^{\hat{j}}, \hat{\ell}, \hat{s}, \hat{p})$, where

$$(\hat{j}, \hat{\mathbf{y}}^{\hat{j}}, \hat{\ell}, \hat{s}, \hat{p}) = \arg \min_{j < i, \mathbf{y}^j, \ell, s, p} \mathrm{NFA}^d_{\ell, s, p}(\mathcal{G}(\mathbf{x}^i, \mathbf{y}^j, \ell, s, p)).$$

We can analyze the spatial and temporal complexities of the algorithm. As in the case of trajectories without holes, the most costly operation is the computation of the $\mathcal{G}$ function. Its complexity is $\mathcal{O}(N^2K^5)$ in space and $\mathcal{O}(N^3K^6)$ in time. Since the extraction must be repeated until there are no more meaningful trajectories, the global time complexity is $\mathcal{O}(sN^3K^6)$, where s is the number of extracted ε-meaningful trajectories. In practice, on a standard PC desktop, for $K = 30$ images, the number N of points per image can go up to about one hundred. For long sequences containing much more that 30 images, the algorithm cannot be used directly (due to the N^6 term in the time complexity), but one could probably obtain good results by cutting the image sequence into small parts and applying the algorithm on each part (raising the issue of long trajectories spanning several parts).

3.3.3 Variable number of points and rectangular images

As for the NFA of trajectories without holes (Section 3.2), we can adapt the NFA given in Proposition 9 to the case of a variable number of points per image. Let us write N_k the number of points present in image k. The simplest strategy consists in applying directly the definition of Proposition 9 with $N = \max_k N_K$. If N_k has strong variations, a more sensitive detection can be obtained by replacing in $\mathrm{NFA}_T(\delta)$ the term N^s by

$$\max_{k_0=i_1<i_2<\ldots<i_s=k_0+\ell-1} N_{i_1} \cdot \ldots \cdot N_{i_s},$$

where T is a trajectory starting in image k_0, with length ℓ and size s. This term is easily computed once the sequence $(N_k)_{1\leqslant k\leqslant K}$ has been sorted.

As in the case of trajectories without holes, the NFA can also be adapted to rectangular images (see Section 3.2.3).

3.3.4 Theoretical results

We now analyze the asymptotic behavior of the NFA on some particular cases. They are all composed of one trajectory spanning the K images, and $N-1$ additional spurious points in each frame. The trajectory has a maximal acceleration of δ.

Long trajectory with a single hole

We first study the case where the trajectory is composed of two parts separated by a unique hole of length $h = \varepsilon K$. We thus have $\ell = K, s = (1-\varepsilon)K, p = 2$, and we write $\alpha = N\pi\delta^2$ as in Section 3.2.6.

We study under which conditions, when K gets large, the trajectory is meaningful, and if it is more meaningful than its first (or equivalently last) part. First we derive an asymptotic expansion of

$$\begin{aligned}\mathrm{NFA}_T(\delta) &= K\cdot(K-\ell+1)\cdot\ell\cdot\binom{\ell}{s}\cdot\frac{\alpha^s}{(\pi\delta^2)^2}\cdot(\varepsilon K)^{2(p-1)}\\ &= K^2\binom{K}{(1-\varepsilon)K}\frac{\alpha^{(1-\varepsilon)K}}{(\pi\delta^2)^2}(\varepsilon K)^2.\end{aligned}$$

From Stirling's Formula, one easily derives the expansion

$$\log\binom{K}{\eta K} = -Kh(\eta) - \frac{1}{2}\log K + \underset{K\to+\infty}{\mathcal{O}}(1),$$

where $\eta \in (0,1)$ is fixed, and

$$h(\eta) = \eta\log(\eta) + (1-\eta)\log(1-\eta).$$

Hence, we have

$$\log \mathrm{NFA}_T(\delta) = K\Big((1-\varepsilon)\log\alpha - h(\varepsilon)\Big) + \frac{7}{2}\log K + \underset{K\to+\infty}{\mathcal{O}}(1),$$

which proves the asymptotic equivalence

$$\lim_{K\to+\infty} \mathrm{NFA}_T(\delta) \leqslant 1 \quad\Longleftrightarrow\quad \log(\alpha) < \frac{h(\varepsilon)}{1-\varepsilon}. \tag{3.26}$$

This asymptotic condition on α is illustrated in Fig. 39. We can notice that the asymptotic limit on $\log\alpha$ given by the right-hand term of (3.26) is quite accurate (and almost linear) as long as

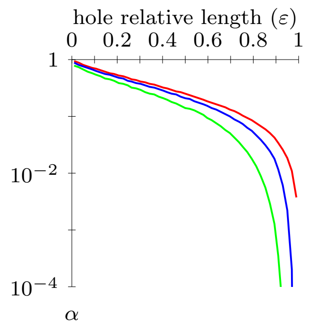


Figure 39: Influence of the hole size. Plot, in function of ε, of the maximal value of α (in log scale) for a trajectory with a single hole of size $h = \lfloor \varepsilon K \rfloor$ to be meaningful. The total number of points in each frame is $N = 100$, and the length of the sequence is $K = 100$ for the green lower curve and $K = 400$ for the blue middle curve. The red upper curve is the asymptotic limit $h(\varepsilon)/(1-\varepsilon)$ of $\log \alpha$ corresponding to the case $K = +\infty$. We can see that when the trajectory hole becomes fairly important, the maximal allowed acceleration for a trajectory to be meaningful plummets.

the relative hole size ε is not too large. If the hole size is half the trajectory length ($\varepsilon = 1/2$), then the asymptotic condition is $\alpha < \frac{1}{4}$.

Now we would like to investigate the condition under which the complete trajectory is more meaningful than its starting or ending parts. Writing $\gamma = (1-\varepsilon)/2$ so that each small trajectory has a size γK, this condition writes

$$\mathrm{NFA}_{T_{1+2}}(\delta)/\mathrm{NFA}_{T_1}(\delta) \leqslant 1,$$

that is

$$\frac{K}{K} \cdot \frac{1}{(1-\gamma)K+1} \cdot \frac{K}{\gamma K} \cdot \frac{\binom{K}{2\gamma K}}{\binom{\gamma K}{\gamma K}} \cdot \frac{\alpha^{2\gamma K}}{\alpha^{\gamma K}} \cdot \frac{((1-2\gamma)K+1)^2}{1} \leqslant 1$$

or equivalently

$$-Kh(2\gamma) + \gamma K \log \alpha + \log K + \underset{K \to +\infty}{\mathcal{O}}(1) \leqslant 0.$$

Since $h(2\gamma)/\gamma = 2h(\varepsilon)/(1-\varepsilon)$, we thus have the property that the whole trajectory is asymptotically more meaningful than its parts when $K \to +\infty$ if and only if

$$\log(\alpha) < 2\frac{h(\varepsilon)}{1-\varepsilon}. \tag{3.27}$$

Note that this inequality constraint on α is stronger (the quantities are negative) than the one obtained in (3.26), hence the case when a trajectory is 1-meaningful but less meaningful than its parts can be encountered.

Dotted trajectories

If the trajectory is made of a succession of single points and one-frame holes, what is the condition as $K \to +\infty$ to have a meaningful trajectory? We now have $\ell = K$, $s = (K+1)/2$ (K being odd), $p = (K-1)/2$, so that from (3.23) we can derive the asymptotic expansion

$$\log \mathrm{NFA}_T(\delta) = \frac{K}{2}\log(16\alpha) + \frac{3}{2}\log K + \underset{K\to+\infty}{\mathcal{O}}(1). \tag{3.28}$$

Hence, a dotted trajectory is asymptotically meaningful when $K \to +\infty$ as soon as

$$\alpha < \frac{1}{16}. \tag{3.29}$$

Dashed trajectories

In the more general setting of a dashed trajectory made of p runs of u consecutive points separated by holes spanning v frames, we have $\ell = K = p(u+v) - v$ and $s = pu$, so that if u and v are fixed,

$$\log \mathrm{NFA}_T(\delta) = 2\log p + \log \binom{p(u+v)-v}{pu} + pu\log\alpha + 2p\log(1+v) + \underset{p\to+\infty}{\mathcal{O}}(1).$$

Now we have

$$\frac{pu}{p(u+v)-v} = \eta + \underset{p\to+\infty}{\mathcal{O}}\left(\frac{1}{p}\right) \quad \text{with} \quad \eta = \frac{u}{u+v},$$

so that

$$\log \binom{p(u+v)-v}{pu} = -p(u+v)h(\eta) - \frac{1}{2}\log p + \underset{p\to+\infty}{\mathcal{O}}(1)$$

and

$$\log \mathrm{NFA}_T(\delta) = \frac{3}{2}\log p + pu\left(\log\alpha + \frac{2}{u}\log(1+v) - \frac{h(\eta)}{\eta}\right) + \underset{p\to+\infty}{\mathcal{O}}(1).$$

Hence, a dashed trajectory is asymptotically meaningful when p tends to infinity if and only if

$$\log\alpha \leqslant \frac{h(\eta)}{\eta} - \frac{2}{u}\log(1+v), \tag{3.30}$$

where $\eta = \frac{u}{u+v}$ is the asymptotic density of known points. This formula yields an interesting relation between the density of known points and the allowed maximum acceleration, as illustrated in Fig. 40. When $u = v = 1$, we have $\eta = 1/2$, $h(\eta) = -\log 2$ and the right-hand term of (3.30) is $-\log 16$, in accordance with (3.29).

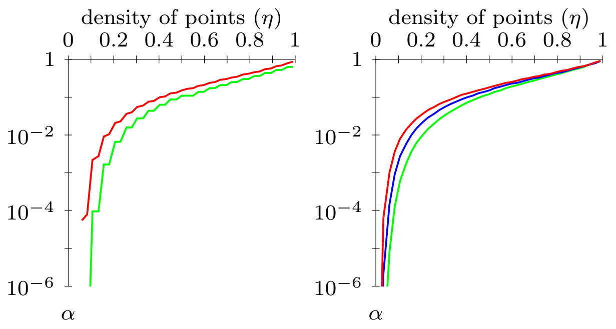


Figure 40: **Required precision for "dashed" trajectories.** Left: The green lower curve shows the maximal value of α allowed (in log scale) for a trajectory to be meaningful when it is made of repetitions of runs of length u and holes of length v, with $u+v = 20$ and $u = \lfloor \eta(u+v) \rfloor$. The length of the sequence is $K = 100$, and the total number of points in each frame is $N = 100$. The upper red curve is the asymptotic limit $h(\eta)/\eta - 2\log(1+v)/u$ corresponding to $p = +\infty$. (note that the staircasing effect is due to the definition of u). As long as the density of known points (η) is large enough, the critical value of $\log \alpha$ is quite well approximated by the asymptotic bound and the relation to η is almost linear. When the density η becomes too small, the maximal allowed acceleration for the trajectory to be meaningful quickly plummets. Right: The asymptotic condition on α given by (3.30), for a variety of values of $u+v = 10$ (lower curve), 20 and 40 (upper curve). The curves are close, hence showing that the minimal required precision for dashed trajectories to be meaningful mostly depend on the density $\eta = u/(u+v)$ and not on the period $u+v$ of the runs.

Geometrical composition
based on a system of particles with attractors and repulsors

4
ASTRE performances

CONTENTS

We analyze in this chapter the performances of the ASTRE algorithm on synthetic and real-world data, and compare them with those of ROADS, a state-of-the-art tracking algorithm which compares favorably to most of the classical tracking algorithms of the literature presented earlier [Veenman, Reinders, and Backer, 2003b]. We then conclude on the respective benefits and drawbacks of both approaches.

4.1 THE ROADS TRACKING ALGORITHM

The ROADS point tracking algorithm can handle points entering and leaving the scene, as well as missing and spurious points. It requires the setup of several parameters, that are listed in Table 3.

w	smoothness model parameter
d_{max}	maximal allowed speed
φ_{max}	maximal allowed smoothness criterion
s	scope width parameter
a_{max}	max. # of missing consecutive points on a track
p_{min}	min. # of present consecutive points on a track
F_g^γ, F_l^γ	optimization cut-off constants
h_{max}	max. # of hypotheses made when optimizing

Table 3: Parameters used in the ROADS algorithm.

The criterion measuring the local smoothness of a trajectory on the consecutive points $(\mathbf{x}, \mathbf{y}, \mathbf{z})$ is

$$\varphi(\mathbf{x},\mathbf{y},\mathbf{z}) = w\left[1 - \frac{v(\mathbf{x},\mathbf{y}) \cdot v(\mathbf{y},\mathbf{z})}{\|v(\mathbf{x},\mathbf{y})\| \cdot \|v(\mathbf{y},\mathbf{z})\|}\right] + (1-w)\left[1 - 2\frac{\sqrt{\|v(\mathbf{x},\mathbf{y})\| \cdot \|v(\mathbf{y},\mathbf{z})\|}}{\|v(\mathbf{x},\mathbf{y})\| + \|v(\mathbf{y},\mathbf{z})\|}\right]$$

where $v(\mathbf{x},\mathbf{y}) = \mathbf{y} - \mathbf{x}$. As we can see, this criterion combines (with a weight parameter w) an angular variation (first term) and a speed variation (second term). Assume that M_k objects are tracked until the k^{th} frame, and N_{k+1} points are observed in frame $k+1$. The trajectories already constructed can either link to one of the observed points, or to a missing "slave measurement", meaning the corresponding object in frame $k+1$ is missing. Additionally, a point of frame $k+1$ can be tagged as spurious. All these possibilities are called *individual assignments*.

Each individual assignment a has a cost $c(a)$. The cost of linking a trajectory to a point in frame $k+1$ is the smoothness criterion as defined above (if one of the past measurements is missing, we estimate its position through linear interpolation). The cost of considering a point in frame $k+1$ as spurious and the cost of linking a trajectory to a slave (missing) measurement are both equal to the value of the parameter φ_{max} (eg. a missing point has the cost of the worst possible trajectory continuation). The algorithm restricts its possible assignments using its cut-off values, for instance, two points in consecutive frames can be linked only if they are at most d_{max} pixels apart.

The local cost of all the individual assignments between two consecutive frames is obtained by averaging their costs. Let $A^k = \{a_1, ..., a_p\}$ be the set of individual assignments between frame k and $k+1$, that is, such that every trajectory tracked in frame k appears in exactly one of the assignments, and every measurement in frame $k+1$ appears in exactly one of the assignments:

$$C^k(M_k, A^k) = \frac{1}{M_k}\sum_{i=1}^{p} c(a_i)$$

The optimization of this cost for a fixed k is a minimum-weight perfect matching problem, and can be solved efficiently using for instance the Hungarian algorithm [Munkres, 1957]. Finally, the *global motion model* averages costs over the whole sequence,

$$C(A^1, ..., A^K - 1) = \sum_{k=2}^{K-1} C^k(M_k, A^k)$$

where A is a multi-assignment $A = (A^2, ..., A^{K-1})$. Other optimization objectives are: as many points as possible should be included in a trajectory, and there should be as few trajectories as possible. In its generality, the global motion model optimization is a NP-hard problem, thus intractable in practice. One of the approximation made by the ROADS algorithm is to sequentially optimize the global model on a restrained time window (typically using $s = 2$ or $s = 3$).

$$A^{k:s}_{\min} = \underset{A^{k:s}}{\operatorname{argmin}}\, C^{k:s}(A^{k:s})$$

where

$$C^{k:s}(A^{k:s}) = \sum_{p=1}^{s} C^{k+p-1}(M^{k+p-1}, A^{k:s}[p])$$

and $A^{k:s} = (A^k, ..., A^{k+s-1})$ is a multi-assignment. The approximation to the global solution is then

$$A = (A^{2:s}_{\min}[1], ..., A^{K-1:s}_{\min}[1]).$$

This approach results in an initialization problem at the beginning of the sequence: the assignment between the first two frames is considered given. To mitigate this strong requirement, the ROADS algorithm uses a "minimal-motion" criterion $c(\mathbf{x}, \mathbf{y}) = \|\mathbf{y} - \mathbf{x}\|$ to initialize the assignment between the first two frames of the sequence, and then a successive up- and down-processing to reduce the imprecision of the initial assignments. We refer the reader to Veenman, Reinders, and Backer, 2003b for a detailed explanation, including how tracks can be ended and started.

The core of the ROADS algorithm (see Algo. 7) is the computation of the local scope optimization. In order to compute $A^{k:s}$ at each frame k, the algorithm recursively enumerates all potential assignments between successive frames in the scope. Of course, this set of assignments is too large to be exhaustively enumerated, and therefore the algorithm uses a branch-and-bound approximation strategy. It first makes an "optimal cost bound" guess C_b by initializing the global solution on the time scope with the local solutions of the minimum-weight perfect matching between each two consecutive frames in the scope. This cost bound is then gradually lowered.

At each recursion step (that is, each frame in the scope), the bound on the current optimal matching cost is lowered by using a cost-bound constraint called $\gamma_{\max}$. It is derived from the cost $C^k_{\min} = C^k(M_k, A^k_{\min})$ of the best possible assignment $A^k_{\min}$ between frames k and $k+1$ (which can be obtained by the Hungarian algorithm) and the current global cost bound C_b by

$$\gamma_{\max} = \min(F^{\gamma}_{l} C^k_{\min}, F^{\gamma}_{g} C_b / s'),$$

where $F^{\gamma}_{l} \geqslant 1$ is the local cost factor, $F^{\gamma}_{g} \geqslant 1$ is the global cost factor and s' is the length of the remaining scope.

The intuition behind this bound is that the cost of the assignment at frame k corresponding to the optimal solution on the time scope cannot be too far from the cost $C^k_{\min}$ of the optimal (local) assignment between the frames k and $k+1$, and that the cost of the assignment on the time scope is more or less uniformly distributed between all pairs of frames, and thus should not be too far from C_b/s'. To enumerate the successive best assignment between frames, ROADS uses Murty's algorithm [Murty, 1968] that takes a cost matrix D^k and a set of previous assignments Y and returns the next best assignment not in Y. The set of all assignments between frames k and $k+1$ is denoted by U^k.

The $\mathrm{costMatrix}(A^{k-1}, k)$ function returns a matrix containing the cost of each possible assignment between a trajectory of A^{k-1} and a point of the kth image.

```
Algorithm: $\mathrm{ROAD}(A^{k-1}, k, C_b, A^{k:s}_{sol})$
input : $A^{k-1}$ the previous assignment,
        $k$ the current frame number,
        $C_b$ the current cost bound,
        $A^{k:s}_{sol}$ the current best assignment
output: $A^{k:s}_{sol}$ the new best assignment
$D^k \leftarrow \mathrm{costMatrix}(A^{k-1}, k)$
if $s = 1$ then
    $A^k_{\min} = \mathrm{minCostAssignment}(D^k)$
    if $C^k(M^k, A^k_{\min}) < C_b$ then
        $A^{k:s}_{sol} \leftarrow (A^k_{\min})$
    end
else
    $Y \leftarrow \varnothing$
    repeat
        $A \leftarrow \mathrm{nextBestAssignment}(Y, D^k)$
        $Y \leftarrow Y \cup \{A\}$
        $C_0 \leftarrow C^k(M^k, A)$
        $T_{sol} \leftarrow A^{k:s}_{sol}[2..s]$
        $R \leftarrow \mathrm{ROAD}(A, k+1, s-1, C_b - C_0, T_{sol})$
        $A^{k:s} = (A) :: R$
        if $C^{k:s}(M^k, A^{k:s}) < C_b$ then
            $C_b \leftarrow C^{k:s}(M^k, A^{k:s})$
            $A^{k:s}_{sol} \leftarrow A^{k:s}$
        end
        $\gamma_{\max} = \min(F^{\gamma}_{l} C^k_{\min}, F^{\gamma}_{g} C_b / s)$
    until $Y = U^k$ or $C_0 \geqslant C_b$ or $C_0 \geqslant \gamma_{\max}$
end
return $A^{k:s}_{sol}$
```

Algorithm 7: Core of the ROADS algorithm

4.2 EXPERIMENTAL SETUP

In the following, we propose to compare the NFA algorithm with ROADS to evaluate its strengths and weaknesses against a state-of-the-art solution. We start with experiments on synthetic data, similar to those used by the authors of ROADS in their presentation papers [Veenman, Reinders, and Backer, 2003a,b]. Let us first briefly present the way they generate trajectories using the Point-Set Motion Generation (PSMG) algorithm (experiments having different parameters will be signaled):

- the initial position of each trajectory is chosen uniformly at random in the first image;
- the initial velocity magnitude is chosen using a normal random variable $\nu_0 \sim \mathcal{N}(\mu = 5, \sigma = 0.5)$ and its angle β_0 is chosen using a uniform distribution in $[0, 2\pi]$;
- the velocity magnitude and angle are updated in each frame using $\begin{cases} \nu_{k+1} \sim \mathcal{N}(\mu = \nu_k, \sigma = 0.2) \\ \beta_{k+1} \sim \mathcal{N}(\mu = \beta_k, \sigma = 0.2). \end{cases}$

The image domain is divided in 100×100 pixels, and the length of the sequence is set to 20 frames (see Fig. 41 for an illustration of the trajectories generated). Most of the experiments are realized with 20 trajectories (like in the ROADS paper).

Since the ROADS authors were comparing their algorithm with an algorithm that did not allow trajectories entering or leaving the scene, they required that all trajectories fit completely inside the frames and span the whole sequence, and we will usually do the same (if a trajectory does not fit inside the frame, we regenerate it). They also impose that in the experiments where points are missing, all points are still detected in the first and last two frames. To have experiments coherent with theirs we generally impose the same constraints, but in some experiments (with a great number of trajectories in the images) constraining trajectories to stay inside the frame seemed unnatural since it forced trajectories to have a beating and swirling motion inside the frames. We chose to keep a constant number of trajectories, but to allow them to leave the image, and when this happens, to generate a new random trajectory that starts on the border of the frames (we force all trajectories to have at least three points in the frame).

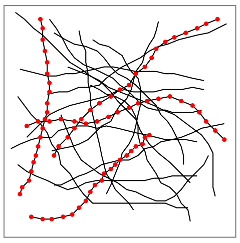


Figure 41: **Point-Set Motion Generation (PSMG) algorithm.** Here we display a sample of 20 trajectories spanning 20 frames generated with the PSMG algorithm, that will be used to produce synthetic data to assess the performance of the ROADS and NFA algorithms. The trajectories have an homogeneous and smooth motion. The points of some trajectories have been highlighted to show the speed.

Additionally, we impose the following constraints:

- we quantize the trajectory coordinates to the nearest integers, thus *implicitly defining the scale of the measurements* to the size of one pixel;
- to avoid ambiguities when comparing the detection results to the ground truth, trajectories cannot share points (otherwise we regenerate one of the trajectories) and when adding noise points we avoid covering already existing points. See Fig. 42 for an illustration of various densities of noise points;
- when we remove points we target solely trajectory points (we do not remove noise points, so we can make experiments with a varying number of trajectory points removed, while keeping a constant number of noise points). We choose a certain uniform probability α of removing a point.

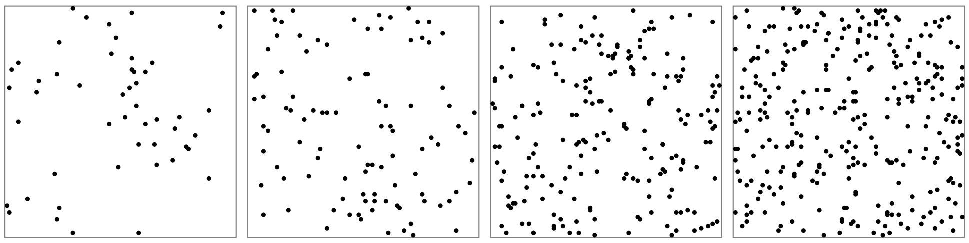


Figure 42: A sample of 50, 100, 200, 300 points uniformly drawn in the image plane. When sequences contain a great number of noise points, the human eye is still capable of detecting trajectories, but, to our knowledge, most of the tracking algorithms have difficulties to extract them correctly.

Performance estimates gathered in the experiments below are averages of a measure over 400 runs of the algorithms. However, in some experiment results, the measure that we compute might be undefined (for instance when no trajectory was detected). In this case, we only take experiment results that have a defined measure into account, and measurements might thus consist of averages of less than 400 repetitions.

A well-known interest of a-contrario methods are their small number of parameters, which simplifies their use and their study. More accurately, the NFA algorithm has exactly one explicit parameter, the maximal NFA value of a trajectory we can extract. The effect of this threshold is simple: it drives the selection of a subsequence of the successively extracted trajectories. In other words, if $\varepsilon < \eta$, $\mathbb{T}(\varepsilon) \subseteq \mathbb{T}(\eta)$, where $\mathbb{T}(x)$ is the set of trajectories extracted by the algorithm for a maximal value of the NFA equal to x. This implies that changing the threshold will not dramatically change the results, contrary to methods like ROADS that use their parameters *in* the computations. In practice, as usually done in a-contrario methods (see Desolneux, Moisan, and Morel, 2008) and unless otherwise specified, we set this threshold to 1.

In contrast, ROADS has many parameters, which can be tuned to set ROADS in different "modes" that may be better suited to certain types of data. Since these parameters might (at least in theory) be learned on data, we felt it was fair to try several sets of parameters and show the best results that can be achieved in the comparisons.

Here is the way we proceeded: we tested six "modes" for the ROADS algorithm on a small batch of data (40 repetitions) for each experiment. We then selected the three modes that would compare the best with the NFA algorithm on the various experiments. Some of the modes will have strengths and weaknesses compared to others, but they mostly have the same global behavior. In practice, the strengths and weaknesses of the NFA method when compared to ROADS do not dramatically change when including several modes, rather than just the most general parameters for the ROADS algorithm (mode 1 below). However, we include the results of the three selected modes for the sake of completeness.

To be fair with the ROADS algorithm that relies on knowing the maximal speed and maximal smoothness criterion of the trajectories in the data, we compute these values and give them to the algorithm. More precisely, for each experiment, and each parameter (eg. number of noise points added), we compute the maximal speed $d_{\max}$ and maximal smoothness criterion $\varphi_{\max}$ before crippling (eg. removing points) across the batch of 400 repetitions (rather than on a per-file basis), and we feed them to the algorithm when processing those 400 repetitions.

The first mode is the general ROADS algorithm with the minimal number of present points set to $p_{\min} = 1$ and the maximal number of interpolated points equal to $a_{\max} = +\infty$. The second mode is $p_{\min} = 1$ and $a_{\max} = 0$, that is, we disallow interpolation. The third mode is

$p_{min} = 3$ and $a_{max} = 0$, we disallow interpolation and we expect to see at least 3 consecutive points on each trajectory segment.

For the three other modes, we set $p_{min} = 3$, $a_{max} = 3$, but rather than choosing the maximal speed and maximal smoothness criterion as given by their maximal value on the batch of 400 repetitions, we select in turn: $d^4_{max} = 0.8 \cdot d_{max}$, $\varphi^4_{max} = 0.8 \cdot \varphi_{max}$ for mode 4, $d^5_{max} = 0.5 \cdot d_{max}$, $\varphi^5_{max} = 0.5 \cdot \varphi_{max}$ for mode 5 and $d^6_{max} = 0.5 \cdot d_{max}$, $\varphi^6_{max} = 0.8 \cdot \varphi_{max}$ for mode 6.

The other default ROADS parameters given in the implementation that was sent to us by its authors were kept unchanged ($w = 0.1$, $F_\ell = F_g = 1.05$, $s = 2$). We tried to make some experiments with $s = 3$, but this would generally not change the results (and sometime even have a negative impact) and be much more computationally intensive. The maximal number of hypotheses h_{max} that ROADS can explore when trying to find the best assignment in the time scope has been kept equal to 300 as in the given implementation.

After running all the ROADS modes on small batches of repetitions, we selected modes 1 (original ROADS), 3 and 4 as giving the best results. They will now be called modes A, B and C (see Table 4). Note that these modes are not real algorithms, since their parameters depend on true data values (d_{max} and φ_{max}) that are not estimated but computed from an oracle. In that sense, the methodology we use to compare the NFA and ROADS algorithms minimizes the issue of parameter selection that is recurrent with ROADS (but this issue will be discussed later, in particular in Section 4.3.4). Note also that ROADS results could probably be slightly improved by trying a larger number of parameters, however our goal is not to make a study of ROADS, but rather to give an idea of the state of the art performances, to make it possible for the reader to appreciate the NFA results.

mode	p_{min}	a_{max}	d_{max}	φ_{max}
A	1	$+\infty$	$1 \cdot d_{max}$	$1 \cdot \varphi_{max}$
B	3	0	$1 \cdot d_{max}$	$1 \cdot \varphi_{max}$
C	3	3	$0.8 \cdot d_{max}$	$0.8 \cdot \varphi_{max}$

Table 4: Parameters defining the three (best) modes of the ROADS algorithm in the experiments used for comparison with the NFA algorithm. Note that these three modes are based on an oracle, that observes the values of φ_{max} and d_{max} on the (supposedly unknown) true trajectories.

4.2.1 Comparison criteria

In the literature, tracking algorithms are generally compared using two sets of criteria: the qualitative description of the situations that the algorithm can handle (missing points, entry of points, etc), and the quantitative criteria given by the number of real structures found in the sequence (eg. the number of real trajectories, of real links between points, etc.), as well as the *precision* and *recall* of the algorithm for these different structures, defined by

$$\text{precision} = \frac{\text{\# of correct structures found}}{\text{\# of structures found}}, \tag{4.1}$$

$$\text{and recall} = \frac{\text{\# of correct structures found}}{\text{\# of actual structures}}. \tag{4.2}$$

The precision allows to measure the number of false positives (more precisely, $1 - \text{precision}$ is the proportion of false positives among found structures), while the recall is linked to the number of false negatives ($1 -$ recall represents the proportion of false negatives among actual structures). It is important to realize that the analysis of an algorithm must be done by considering simultaneously the precision and recall (or equivalent variables), since varying a parameter or a threshold of an algorithm generally does not improve both quantities but sets a different trade-off between the two, resulting in a better recall and worse precision or vice-versa.

In some experiments, the presence of noise points limits the interest of the number of real (whole) trajectories found as a significant criterion, although it is widely used in the literature. Indeed, a well-placed noise point can sometime better fit the trajectory than its "real" counterpart, thus giving a realistic and usable trajectory as output, yet one that will not be counted as a real trajectory. We therefore chose to generally use the *number of correct links* as a significant structure for the precision and recall criteria. A link is simply two points that appear consecutively on a trajectory (possibly separated by a hole). Thus, if a noise point better fits a trajectory than its "real" counterpart, we will only "miss" two correct links (that include the real point), and "create" two false links (that include the noise point). However, when using the number of correct links, we do not account for trajectory over-segmentation, under-segmentation, or mixing. More precisely, if we split a trajectory in halves, or if we join two distinct trajectories, we will barely notice it from the point of view of the number of correct links criterion, but we would have noticed it using the number of correct trajectories criterion. The same is true for "mixed" trajectories: if two trajectories cross at a point in time, we might start by following trajectory A, and then either choose to continue with trajectory A or to "hop" on trajectory B. In the latter case, the number of correct links criterion will barely be affected, but the number of correct trajectories criterion would. This particular problem of crossing trajectories appears however to be difficult to solve properly, and would certainly requires *a priori* knowledge. We believe that once the trajectories have been detected, even if they have been mixed, a simple post-processing task might be sufficient to split crossing trajectories in part at the crossing points, and reconstruct the real trajectories using an *a priori* (having the trajectories bounce if we are following billiard balls, or having them cross if we are looking at fishes in an aquarium).

For the qualitative criteria, ROADS is able to account for missing and spurious points, as well as points leaving and entering the scene. The NFA algorithms come in two flavors, one that allows for missing points, and the other that does not. The latter is used for computational reasons (it is much faster) in some of the following experiments. Both NFA algorithms allow spurious points, as well as points leaving and entering the scene.

4.3 TRAJECTORIES WITHOUT HOLES

4.3.1 Variable number of spurious points experiment

First, we investigate the influence of noise (spurious points) on trajectory detection. We generate sequences spanning 20 frames and having 20 trajectories, and we add 0 to 320 noise points uniformly at random to each image. Since we do not remove points, we can use the version of the NFA algorithm that does not take holes into account. We then run the NFA ($\varepsilon = 1$) and the ROADS (modes A,B,C) algorithms, and compare the results by computing the average recall and precision over 400 repetitions. For the precision estimate, the averaging is limited to the repetitions that lead to at least one detection, since the precision is not well defined when no structure is found.

As we explained earlier, the precision and recall are computed for two different criteria: the number of correct links and the number of correct trajectories. For the criterion based on

# spurious points	0	40	120	200	280	320
NFA	20.1	20.2	18.9	15.5	7.1	6.1
ROADS(A)	20.0	70.0	151.5	227.1	300.9	335.7

Table 5: **Average number of detected trajectories depending on the level of noise.** We compare the average number of trajectories detected by the NFA and ROADS algorithms on data made of 20 real trajectories spanning the entire sequence plus a varying number of spurious points (from 0 to 320) in each frame. We see that NFA is very conservative in its detections (it only detects the trajectories that it considers to be non-ambiguous), and this results in a high precision (see Fig. 43). On the other hand, ROADS makes many false detections (it should not find more than 20 trajectories per sequence).

the number of correct links (Fig. 43), the NFA algorithm performs much better in terms of precision: the precision remains very high (above 80%) for the NFA algorithm, but drops very fast for ROADS (under 20% when the number of spurious points exceeds 140). This illustrates a classical property of a-contrario detection models: the robustness to noise. As concerns the recall, the NFA algorithm performs better than all versions of ROADS up to 200 spurious points, and is slightly under the C mode of ROADS beyond this level of noise. Considering the number of false detections made by ROADS at these levels of noise (the precision is under 10%), this is not very significant and the global comparison is clearly in favor of the NFA algorithm. Table 5 clearly illustrates this: if ROADS manages to find a lot of correct links, it is solely because it makes a huge number of detections when the number of noise points increases, whereas the NFA algorithm correctly finds 20 trajectories in low noise and makes fewer detections when the noise level increases.

When we look at the number of correct trajectories found (Fig. 44), we see that ROADS is very good when there are no noise points, which will be confirmed below. The NFA algorithm is a bit less efficient (both in terms of precision and recall) when the number of spurious points is under 40, but for higher levels of noise it is much more robust than ROADS, whose performances collapse very quickly (both in terms of precision and recall). Anyway, we argue here that the correct number of trajectories criterion, although often used in the literature, is not the best way to assess the quality of algorithms in the case of spurious points. As we remarked before, there are plenty of reasons why a detected trajectory could be counted as undetected while it is very near an actual trajectory (a missing endpoint, a noise point fitting better the trajectory smoothness, trajectory crossings, etc.). Also, it is clear that applications based on data corrupted by a medium or high level of noise are more interested in a high rate of local point tracks (links) than in the unlikely perfect reconstruction of each trajectory.

4.3.2 Variable density experiment

We now test how the algorithms behave when we increase the number of points. In this experiment, we do not consider spurious or missing points, so there is no noise and the difficulty of the trajectory detection problem only comes from the ambiguities produced by the large number of mixed trajectory points. We generate sequences of 20 frames, containing 10 to 140 points moving according to the PSMG model, where we allow trajectories to leave the image frame (when a trajectory leaves the image frame, we generate a new trajectory starting at a random position on the image frame, in order to keep a constant number of points throughout the sequence). Then we compute the precision and recall for the correct links criterion (Fig. 45) for the ROADS and NFA (without holes) algorithms.

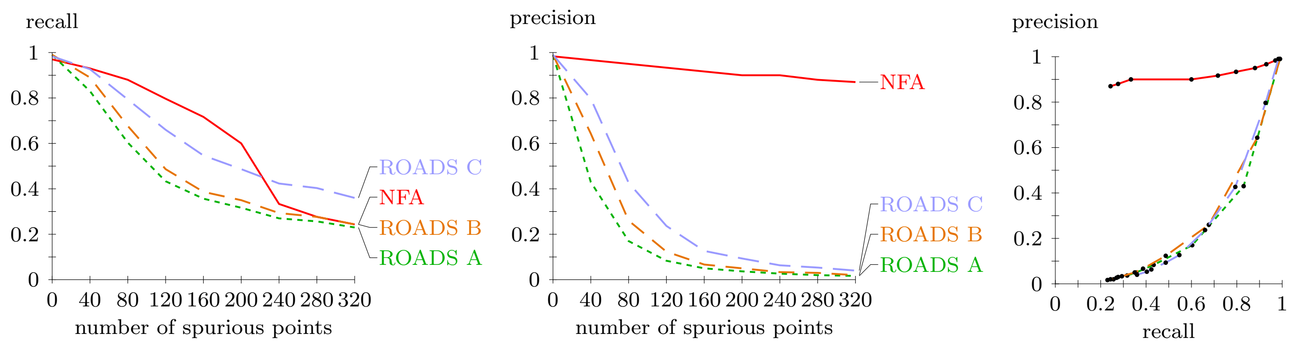


Figure 43: Influence of spurious points (# correct links criterion). On synthetic data containing 20 real trajectories spanning the entire sequence (20 frames) plus a varying number of spurious points (from 0 to 320), we compute the average recall (left) and precision (middle) obtained with the NFA and ROADS algorithms over 400 realizations, as a function of the level of noise (number of spurious points), or together (right). The most striking result here is that the precision of the NFA algorithm is almost constant, no matter the number of spurious points. This means that the NFA algorithm makes very few false detections (which is how we designed it), while keeping a recall rate that is above the one of ROADS as long as the number of spurious points is under 200 (which is more surprising). On the contrary, the poor precision of the ROADS algorithm in medium or high noise conditions makes its recall values quite insignificant: if ROADS finds a large number of correct links, it is mostly because it proposes a high number of links, most of which are false detections (see also Table 5).

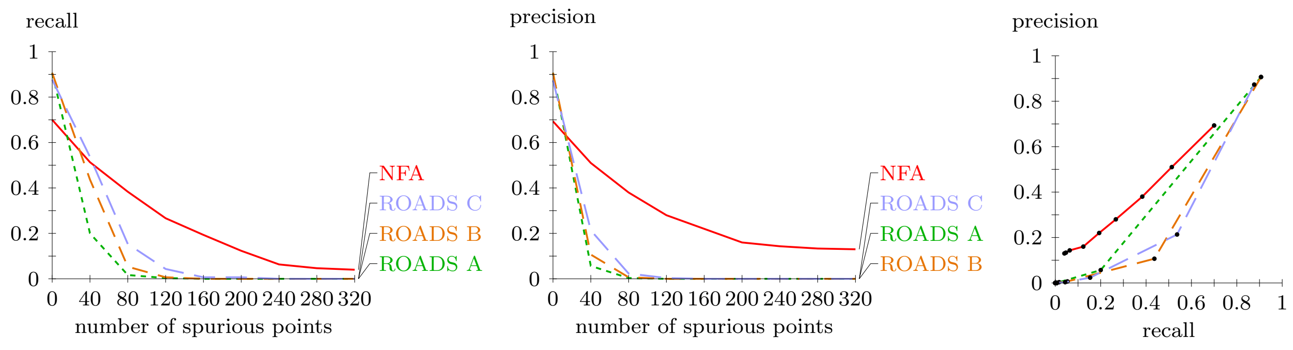


Figure 44: Influence of spurious points (# correct trajectories criterion). The results of the experiments conducted in Fig. 43 are now analyzed with a different criterion (the number of correct trajectories, instead of the number of correct links) for the definition of precision and recall. We can see that the NFA algorithm behaves better than all ROADS modes as soon as there is a reasonable level of noise, but still behaves pretty poorly in comparison with the "number of links" criterion (see Fig. 43). In fact, the number of exact trajectories is a questionable criterion in the presence of noise, so we shall not use it any more in the following.

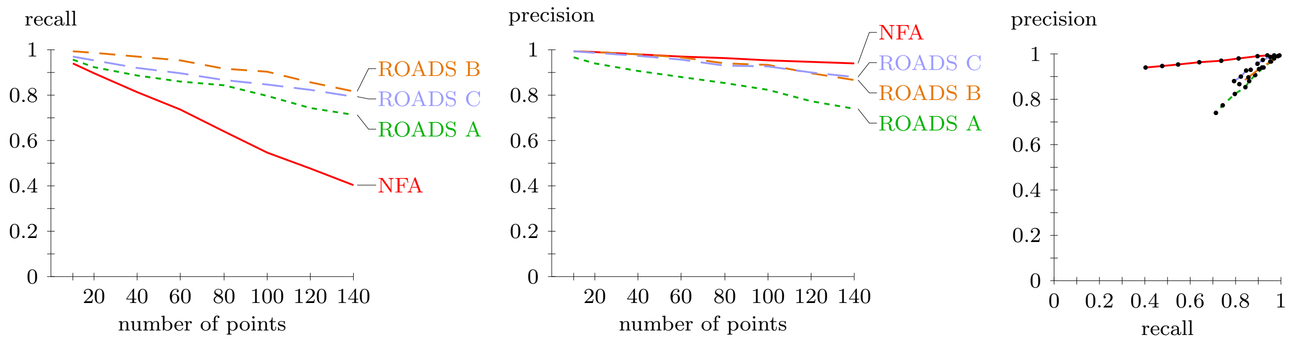


Figure 45: Influence of the number of trajectories Average recall (left) and precision (middle) are computed for the number of correct links criterion on 400 repetitions of synthetic data made of a given number of random trajectories (varying from 10 to 140) in a sequence of 20 frames. The analyzed algorithms are the three modes of ROADS (A, B, C), and two variants of the proposed NFA algorithm: the standard variant (threshold $\varepsilon = 1$ on the expected number of false alarms), and the no-threshold variant ($\varepsilon = +\infty$). As we can see, the precision of both NFA variants is very high (like for ROADS B and C), but the recall of the standard NFA algorithm is significantly worse than the one of ROADS. In this setting where no noise points are present, these missing detections can be avoided by removing the thresholding process in the NFA algorithm: for this $\varepsilon = +\infty$ variant, both recall and precision are as good as the best modes of ROADS.

When using the standard threshold ($\varepsilon = 1$) in the NFA algorithm, we obtain results that are similar (slightly better) than ROADS in terms of precision but significantly worse in terms of recall. However, knowing that there is no noise in these data, it makes sense to try to set the NFA threshold to $+\infty$ (that is, no threshold), and in this case the results obtained by the NFA algorithms are similar to the best modes of ROADS. This is an unexpected good surprise for the NFA algorithm, that detects trajectories in a greedy way (by iterating a best-trajectory-detection/trajectory-removal process) without considering at all the global inter-frame assignment problem like ROADS. In the absence of noise points, one could have expected this assignment step to bring a significant edge to ROADS.

4.3.3 Sensitivity to data smoothness

The trajectories generated in the previous experiments are somehow smooth and quasi-linear (see Fig. 41). In order to see if the algorithms can cope with trajectories that do not completely fit the model, we try to detect trajectories having a potentially high acceleration. For that purpose, we consider different values of σ, the standard deviation of the acceleration magnitude used in the PSMG synthesis procedure (see the very beginning of Section 4.2). The effect of this parameter on the synthesized trajectories is illustrated in Fig. 46.

We first reproduce the last experiment (Fig. 45), in which there are no noise points and the number of synthesized trajectories ranges from 10 to 140, and evaluate the effect of the σ parameter both for the unthresholded NFA algorithm (Fig. 47) and the ROADS B algorithm (Fig. 48). Whereas the performance of the NFA algorithm barely depends on σ (and remains high), ROADS exhibits a high sensitivity to this parameter, and its performance quickly col-

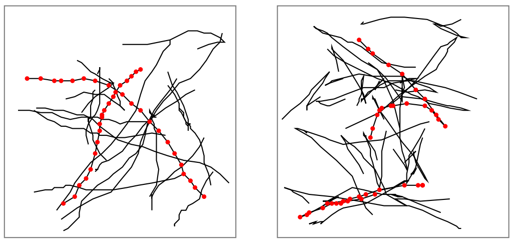


Figure 46: **Changing the acceleration variance**. A sample of 20 trajectories generated using the PSMG algorithm, when the standard deviation of the acceleration magnitude (σ) is 1 (left) and 4 (right). The points of two trajectories have been highlighted to show the speed. We study the sensitivity of the algorithms to data variability by analyzing their performances when we increase σ.

lapses as σ increases. The same conclusion arises from the analysis of data generated with 10 noise points per frame plus 20 synthetic trajectories (see Fig. 49).

Thus, the sensitivity to data smoothness is a major difference between the NFA and ROADS algorithms. The poor results obtained by ROADS for $\sigma = 1$ (see Fig. 48) could probably be improved by a specific choice of the ROADS parameters (specially adapted to $\sigma = 1$), but this kind of optimization will not be efficient on most real-world data, since one can expect to observe a high variability of accelerations on such data. Conversely, the robustness of the NFA algorithm to the σ parameter is an indication that it can probably handle well real-world data containing various levels of acceleration.

4.3.4 Parameter tuning

One major interest of most a-contrario models is that they permit to obtain detection algorithms "without parameters", or, more precisely, algorithms for which there exist natural values of the parameters that work well in all situations. Both NFA algorithms we propose here (the no-hole and hole versions) have only one parameter: the threshold ε used to decide whether a trajectory should be detected or not. Since ε corresponds to an upper bound on the expected number of false alarms in pure noise data, its default value is classically set to 1 (see Desolneux, Moisan, and Morel, 2008). In Fig. 50, we examine the sensitivity of the NFA no-hole algorithm with respect to the choice of ε. We use the same experimental setting as in Fig. 43 (20 frames containing 20 real trajectories plus several spurious points), and examine how recall and precision are affected by different choices of ε. The results clearly show that the default value $\varepsilon = 1$ ($\log_{10} \varepsilon = 0$) is nearly optimal, in the sense that it is small enough to guarantee a strong precision control, and large enough to offer good recall performances. It is nonetheless interesting to notice that slightly better performances (same precision and better recall) can be obtained with greater values of ε (typically $\log_{10} \varepsilon = 2$ or 3).

In Fig. 51, the average precision/recall curve obtained with the NFA algorithm for different values of the threshold $\log_{10} \varepsilon$ (in the case of 160 spurious points) is displayed on the left. On the right, we report the average performances of ROADS on the same data points, considered for several values of the two main parameters of this algorithm, namely the maximal speed and the maximal smoothness. The maximum speed parameter varies from the actual value in the $[-50\%, +50\%]$ range (from one curve to another) and the maximum smoothness varies from

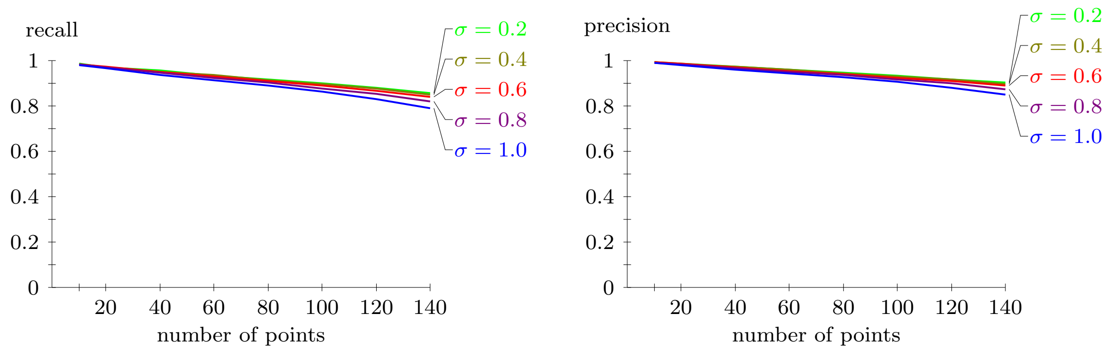


Figure 47: Influence of the data smoothness (unthresholded NFA algorithm). We use the same experimental setting as Fig. 45, and examine the sensitivity of the unthresholded ($\varepsilon = \infty$) NFA algorithm to the smoothness of the analyzed synthetic data. More precisely, we consider several values of σ, the standard deviation of the acceleration magnitude (a parameter of the synthesis algorithm, PSMG), and estimate the precision and recall (correct links criterion) as functions of the number of synthetic trajectories. We can see that the NFA algorithm is extremely robust to σ, since both the precision and recall performance curves remain unchanged when σ varies.

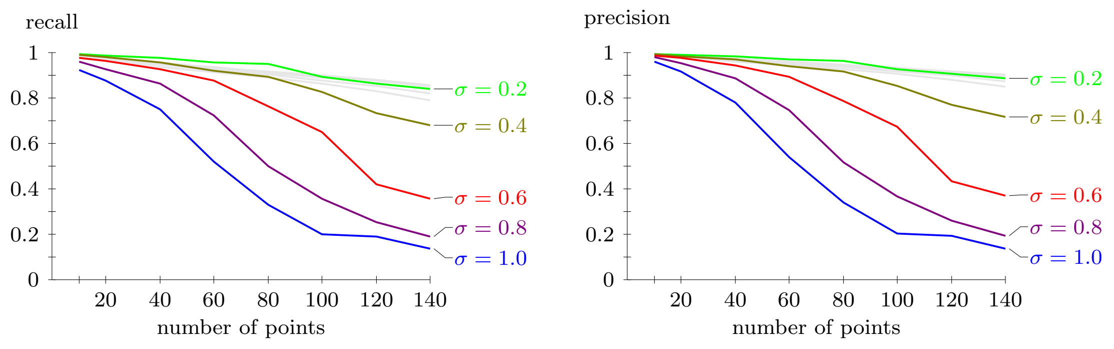


Figure 48: Influence of the data smoothness (ROADS B algorithm). We analyze the same data as in Fig. 47, now with the ROADS algorithm, mode B (the best mode for these data, see Fig. 45). Contrary to what happens for the NFA algorithm, the ROADS method exhibits a severe sensitivity to σ, since both recall and precision performances, that were at the same level as the NFA algorithm for $\sigma = 0.2$ (grey shadow curves), are strongly affected when σ increases. As we shall see in Section 4.5, the sensitivity/robustness to data variability has strong consequences when real-world data are analyzed. Note incidentally the strong similarity between the recall and the precision curves, which comes from the fact that in this set of experiments, the number of detected links is most of the time equal to the number of actual links (see Equation (4.1) and (4.2)), probably because there are no spurious points.

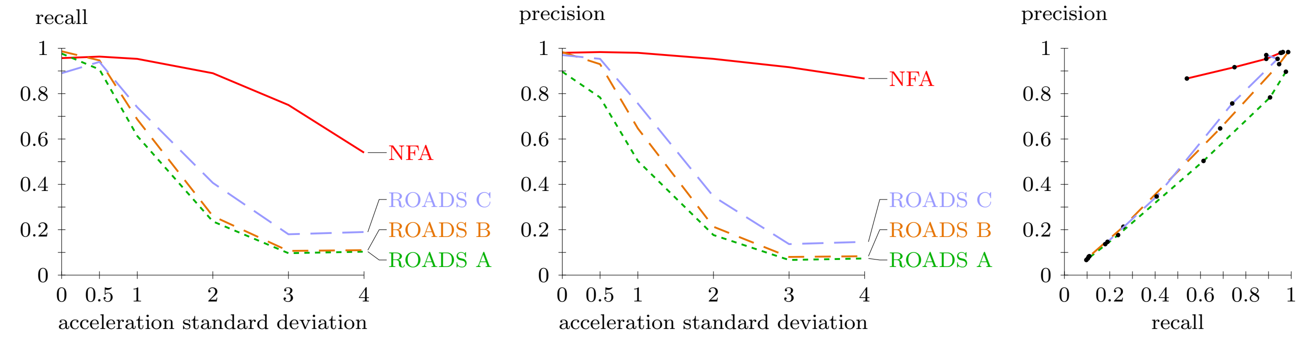


Figure 49: Sensitivity to data smoothness. The average recall and precision of the ROADS algorithm (modes A, B, C) and the standard NFA algorithm ($\varepsilon = 1$) are compared in synthetic data made of 10 noise points per frame plus 20 random trajectories spanning 20 frames, in function of the standard deviation of the acceleration magnitude, a parameter used in the PSMG synthesis procedure. These results corroborate the ones obtained in Fig. 47 and 48: the performances of the NFA algorithm are not too much affected by the increase of σ (except for the recall when the variance becomes large, probably because the problem of recovering the true trajectories becomes objectively difficult), whereas the performances of all ROADS algorithms collapse, both in terms of precision and recall.

the actual value in the $[-95\%, 50\%]$ range (inside each curve). Note that the best performances of ROADS are obtained inside these ranges (-25% speed, -90% smoothness).

As we can see, not only the performances of ROADS are way under those of NFA on these data, but also the parameter tuning is much more difficult and crucial (we have to explore carefully a bidimensional domain, while $\varepsilon = 1$ is almost optimal for the NFA algorithm).

4.3.5 NFA as a criterion for trajectory selection

Contrary to ROADS, which is by nature an algorithm (relying in particular on some heuristics), the NFA we propose here is first and foremost a criterion to compare trajectories. The greedy algorithm we described, based on the iteration of a "best trajectory (minimal NFA) detection / trajectory removal" process, is only one possibility to use the NFA criteria (3.5) and (3.23), and it is possible to design other algorithms based on these criteria. In particular, given a trajectory detection algorithm, it is always possible to use the NFA criterion as a post-processing step, that simply keeps from the output of the considered algorithm the trajectories having a NFA under a certain threshold ε.

We tested this possibility with the ROADS algorithm, and reported in Fig. 52 the results obtained on the synthetic data used in the previous section (parameter tuning). It appears that the mixed ROADS+NFA algorithm we obtain this way performs much better than ROADS alone in terms of precision (because the NFA filtering permits to eliminate most false detections), but the performances in terms of recall do not attain the ones of the NFA algorithm alone. Hence, the "NFA filtering" strategy is efficient but does not provide a particularly interesting new algorithm when applied to ROADS. It is not impossible, however, that such a strategy could be successful, in particular in situations where only special kinds of trajectories appear and a good detection algorithm (in terms of recall) exists. In that kind of situation, one could expect NFA filtering to increase the precision up to a high level, without damaging to

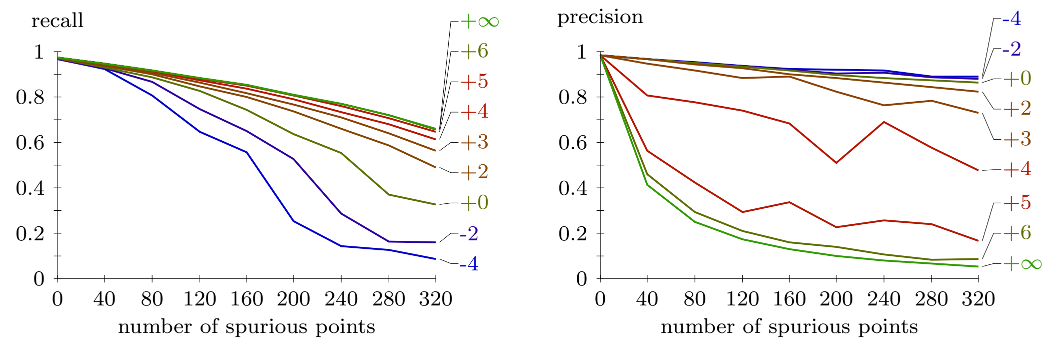


Figure 50: Influence of the NFA threshold ("). We consider the same experiment as in Fig. 43 (that is, 20 real trajectories spanning 20 frames, with a given number of spurious points in each frame), and examine the influence of the threshold ε arising in the NFA algorithm. Recall and precision curves are plotted in function of the number of spurious points, for different values of $\log_{10} \varepsilon$ (ranging from -4 to $+\infty$). We can see that the good precision control predicted by the theory for $\varepsilon \leqslant 1$ ($\log_{10} \varepsilon \leqslant 0$) is well achieved, since the first significant precision losses occur around $\log_{10} \varepsilon = 3$. Hence, the default value $\log_{10} \varepsilon = 0$ is a good compromise in this experiment, even if slightly better recalls (without significant precision losses) can be achieved by using greater values like $\log_{10} \varepsilon = 2$.

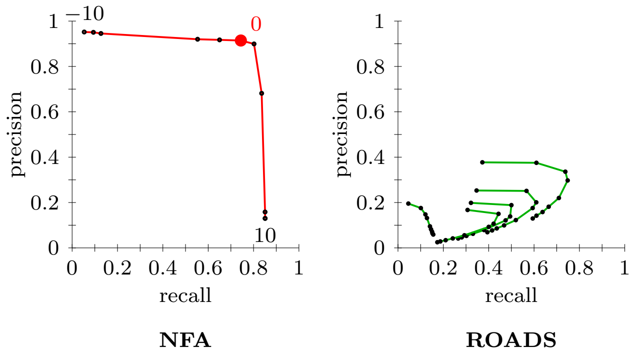


Figure 51: Performance and parameter tuning. We consider a particular case of Fig. 50, that is, synthetic data made of sequences of 20 frames containing 20 real sequences and 160 spurious points on each frame. The average performances in terms of precision/recall is then evaluated for the NFA algorithm (left) and the ROADS algorithm (right), with varying values of the algorithm parameters. For the NFA algorithm, the only parameter is the threshold ε (or, $\log_{10} \varepsilon$, as displayed on the figure), and we can see that the default value $\log_{10} \varepsilon = 0$ is very near to be optimal, as was remarked earlier in the comment of Fig .50. For ROADS, not only the performances are much worse (especially in terms of precision), but they are also quite sensitive to the choice of the maximum speed and maximal smoothness parameters.

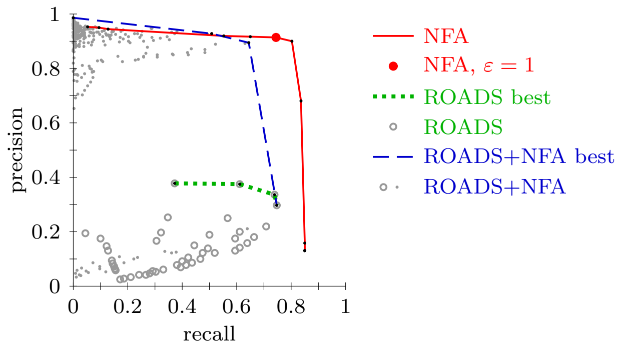


Figure 52: **ROADS output filtered by the NFA algorithm**. We consider the same synthetic data as in Fig. 51, but now add to the comparison of NFA and ROADS algorithms a combination of them that consists in detecting trajectories with ROADS and keeping only those having a NFA under a certain threshold ε. Since each algorithm depends on parameters (1 for NFA, 2 for ROADS, 3 for ROADS+NFA), we explore systematically all parameter values and compute the upper performance envelope (curves named *best*). As we can observe, the major drawback of ROADS (which is its high rate of false detections) can be corrected by NFA filtering, which results in a dramatic increase of precision (up to the level of the NFA algorithm alone). However, this correction does not permit to attain the same level of recall (around 0.75 for NFA, versus 0.6 for ROADS+NFA in the high precision zone). Note also that the mixed ROADS+NFA algorithm would be much more complicated to use than NFA alone, in reason of the 3 parameters that have to be set.

much the recall performances. Note that such a strategy guarantees, thanks to the properties of the NFA criterion (3.1), the control of the number of false detections in random data.

4.4 TRAJECTORIES WITH HOLES

We now examine the performances of the second NFA algorithm (Section 3.3), that is able to handle trajectories with holes. We compare it to ROADS using the same kind of conditions as in Fig. 43 (20 real trajectories, 20 frame, several spurious points added in each frame), except that we now consider incomplete trajectories (20% of the points of the true trajectories are removed before spurious points are added). The conclusions made in the no-holes case remain unchanged (see Fig. 53): the ROADS algorithm detects true trajectory links as well as the NFA algorithm, but at the price of many false detections, whereas the NFA algorithm makes almost no false detection (the precision remains above 0.9, even for 70 spurious points per frame).

4.5 TRAJECTORIES OF REAL-WORLD IMAGES

4.5.1 The *snow* sequence

In this part, we evaluate the relative performances of NFA and ROADS algorithms on a real-world sequence named *snow*. To produce this sequence, we filmed falling snowflakes in front

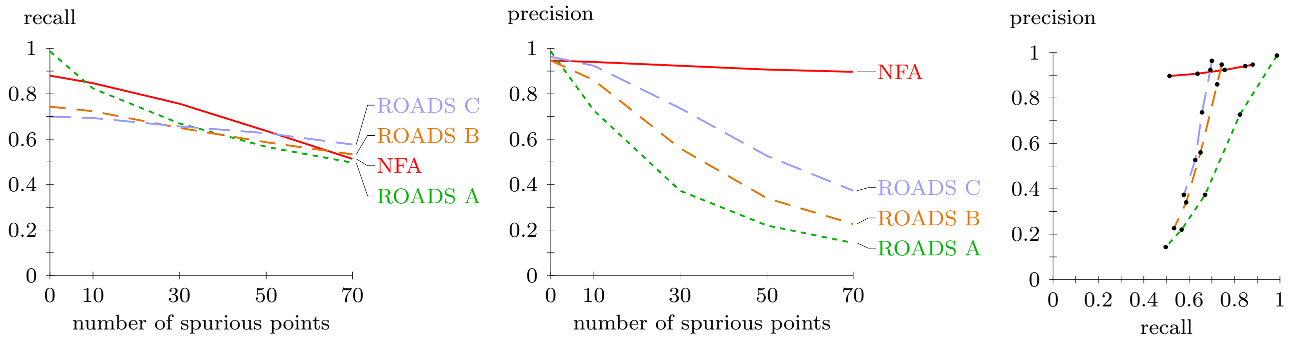


Figure 53: **Influence of spurious points for trajectories with holes.** We generate 20 trajectories spanning the whole sequence (20 frames), and remove randomly 20% percent of the points, before we add a varying number of spurious points (from 0 to 70). On these synthetic data (with 400 repetitions), we estimate the recall (left) and the precision (middle) of the ROADS and NFA algorithms for the "number of correct links" criterion. The obtained results are very similar to those of Fig. 43: the recall values are roughly the same for all algorithms, but only the NFA algorithm manages to maintain a high precision (above 0.9) as the number of spurious points increases, while all ROADS variants make lots of false detections.

of a dark metal door with a high-speed (210 fps) camera, and then subsampled the high speed sequence at 30 fps by taking 1/7 of the original frames. This way, we obtained a classical 30 fps sequence of 40 images, on which we ran a simple point extraction process that we describe below. The high-speed version was used in the same way in order to build a hand-made ground truth for trajectories.

We purposefully used a very simple extraction process to produce data as objectively as possible, without trying to adapt the detection algorithm in a way that would affect (and ease) the tracking part. The snowflakes (but also some stains on the metal door background) were detected in the following way: we smoothed the images using a simple Gaussian kernel, and we computed the mean background image on a few frames of the subsampled (30 fps) sequence. We then thresholded the image differences, processed the result with a morphological closing, and extracted the connected components. For each connected component, we kept the centroid position, rounded to the nearest point on the integer grid, as a trajectory data point.

In the resulting point sequence, many objects were detected as several close points in the sequence (in particular the stains on the background and some big snowflakes). This made it sometimes hard to define the ground truth trajectories. To alleviate this difficulty, we removed all points in the sequence that were in a certain radius of another point (we chose the smallest radius that would resolve almost all ambiguities). An example of detections on one frame of the sequence is displayed on Fig. 54. We finally extracted the ground truth trajectories by hand.

The resulting point sequence is interesting because it presents a mix of difficulties: there are widely varying trajectory types (points in the background that practically do not move, very slow snowflakes with curvy trajectories, very fast snowflakes with almost linear trajectories). There are missing points (missing detections or detections removed because of the windowing process), and a few noise points (but the relatively high detection threshold gave more missing points and fewer noise points).

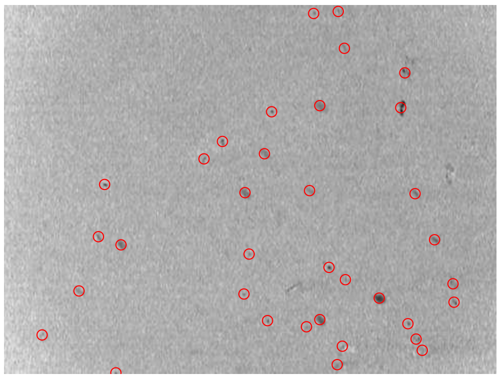


Figure 54: An image of the *snow* sequence (inverted grayscale), with overlaid detections.

Finally, we subsampled the high-speed point sequence by keeping only 1/7 of the frames, and subsampled accordingly the ground truth trajectories. The resulting trajectories containing less than 3 points were eliminated from the ground truth reference, but the corresponding points were kept in the data (thus becoming noise points). The final result of this process (30 fps *snow* point sequence and associated ground truth) is available on the web site `http://www.mi.parisdescartes.fr/~moisan/astre/` The first row of Fig. 59 gives an idea of the ground truth trajectories extracted from the *snow* sequence.

4.5.2 Parameter tuning

There are several parameters to set for ROADS (see Table 3), and they give varying results. Namely, we can set the size s of the time scope (we chose 2, giving the best results), the minimal number p_{min} of consecutive present points for a trajectory to be considered (we chose 1, 3, 5 or 7), the maximal length of interpolation a_{max} before we loose the trajectory (we chose 0, 4, 8 or $+\infty$), the maximal smoothness criterion φ_{max} and the maximal speed d_{max}. The way to choose the best parameters is not obvious, but it appears on Fig. 55 that the most important parameter is the maximum allowed speed d_{max}. The choices $p_{min} = 1$, $a_{max} = 0$ and $\varphi_{max} = 0.6$ are among the best possible for the *snow* point sequence, and would probably achieve reasonable performances on similar sequences too. As concerns the choice of d_{max}, the ground truth value (160) is much too large, and much better results are obtained with $d_{max} = 20$. This fact, that comes from the inability of ROADS to deal with a variety of trajectory speeds at the same time, is analyzed more precisely later. Note that the ground truth value of φ_{max} is 0.58.

On the *snow* sequence, extracting trajectories using the NFA algorithm with holes would return a sequence of trajectories having a value of $\log_{10}(\text{NFA})$ varying from -40 to $+10$, and the optimal precision/recall values would be obtained by thresholding this value with $\log_{10} \varepsilon = +5$ (see Fig. 56). Even without access to the ground truth, finding this value is relatively easy, since one simply has to look for values slightly above the (nearly optimal) default value $\log_{10} \varepsilon = 0$. This strategy works well in all synthetic experiments we considered earlier, and also in the present case of the *snow* sequence. In view of the false detection control offered when $\log_{10} \varepsilon = 0$, such a strategy is probably efficient on most (not to say all) point sequences.

Thus, as we mentioned before, one great interest of the NFA algorithm is that the parameter tuning step is *much* more easier than in other algorithms like ROADS, for which it can be

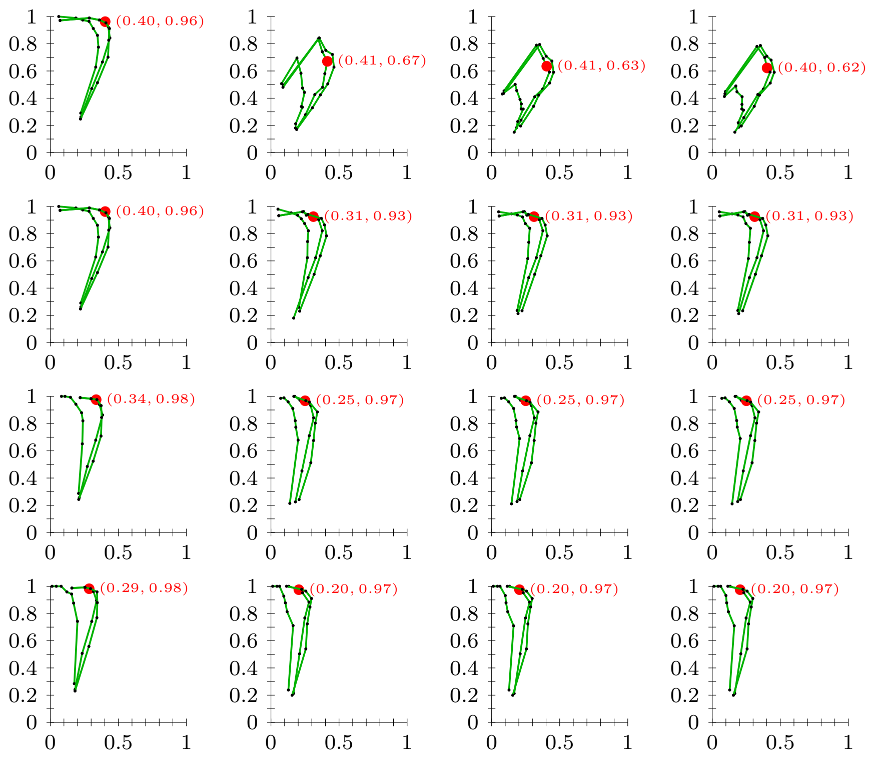


Figure 55: **ROADS parameter tuning on the snow sequence.** We vary all ROADS parameters on the *snow* point sequence, and show the associated performances in the (recall,precision) plane using the available ground truth for that sequence. Each column has a distinct $a_{\max} = 0, 4, 8, +\infty$, and each row has a distinct $p_{\min} = 1, 3, 5, 7$. Each curve corresponds to a different maximal smoothness criterion value $\varphi_{\max} = 0.2, 0.4, 0.6$ and each point of a given curve corresponds to a different maximal speed criterion $d_{\max} = 2, 5, 10, 15, 20, 25, 30, 40, 80$. The big red point corresponds to the parameters $\varphi_{\max} = 0.6$ and $d_{\max} = 20$, that seem to achieve a good precision/recall compromise for all values of $a_{\max}$ and $p_{\min}$. The numbers indicate the corresponding recall and precision.

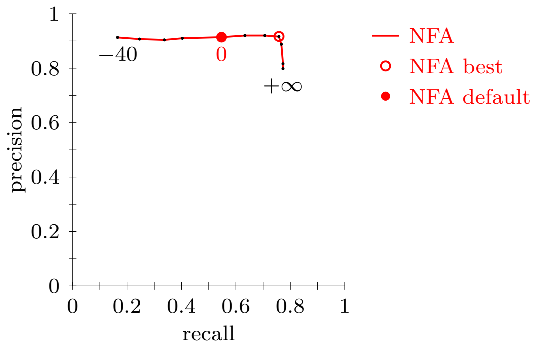


Figure 56: **NFA parameter tuning on the snow sequence.** The performances of the NFA algorithm with holes on the *snow* sequence are represented in the (recall,precision) plane in function of the threshold parameter $\log_{10} \varepsilon$. While the precision remains merely constant, a good recall is obtained by the default value (0) of $\log_{10} \varepsilon$, but the results can be improved by choosing a slightly greater value ($\log_{10} \varepsilon = +5$, that corresponds to the "NFA best" point).

a real burden, especially when dealing with complex data (with unknown ground truth) on which the effect of a parameter change can be very difficult to evaluate. This relative parameter sensitivity is illustrated on Fig 57.

4.5.3 Comparison of ROADS and NFA algorithms

To compare the results obtained by the ROADS and NFA algorithms on the *snow* sequence, we use for each algorithm two different settings: the *default* setting and the *best* setting.

For ROADS, the *default* setting corresponds to $a_{\max} = +\infty$, $p_{\min} = 1$, $d_{\max} = 130$, and $\varphi_{\max} = 0.58$. Note that $\varphi_{\max} = 0.58$ corresponds to the oracle value, that is, the (theoretically unknown) maximum value of φ on the ground truth trajectories. For $d_{\max}$, we chose the value $d_{\max} = 130$ to allow ROADS to detect all the trajectories in the main bulk of trajectories (choosing $d_{\max}$ as the real maximal speed (160) would give worse results). The *best* setting for ROADS was chosen after a careful (and a bit cumbersome) parameter analysis (see Fig .55), that lead to $a_{\max} = 0$, $p_{\min} = 1$, $\varphi_{\max} = 0.6$, and $d_{\max} = 20$.

Concerning the NFA algorithm, the *default* and *best* settings simply correspond to $\log_{10} \varepsilon = 0$ and $\log_{10} \varepsilon = +5$ respectively (see Fig. 56). As for ROADS, the *best* setting was chosen a bit arbitrarily, as one of the best recall values avoiding a significant loss of precision. The strong L-shaped aspect of the precision-recall curves made that choice quite easy (in the sense that other possible choices would not differ much).

The precision-recall performances obtained for ROADS and NFA algorithms (for both *default* and *best* settings) are given in Table 6. In the *best* configuration, the two algorithms attain similar performances in terms of precision, but at the price of a high number of false detections (poor recall) for the ROADS algorithm. Moreover, the *default* parameter setting of ROADS gives very poor results, while the *default* parameter setting of NFA ($\varepsilon = 1$) achieves a better performance than the *best* ROAD settings. These results are analyzed in greater details in Fig. 58, where it appears that the main limiting factor of the ROADS algorithm seems to be its inability to handle simultaneously (that is, with a same set of parameters) trajectories with various lengths and speeds.

Finally, we display the results on Fig. 59.

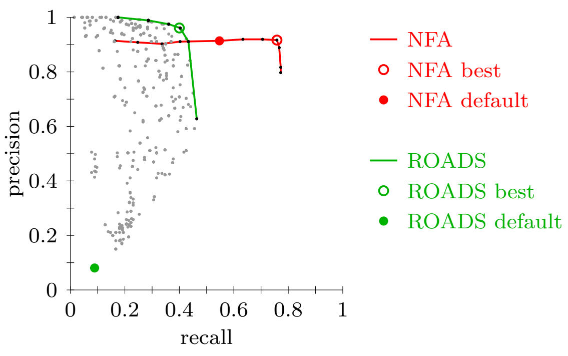


Figure 57: Comparison of NFA and ROADS algorithms on the snow sequence. All results obtained on the *snow* sequence with the NFA (with holes) and ROADS algorithms (both with varying parameters) are represented in the (recall,precision) plane, with a point for each set of parameters (thus, the performance of each algorithm is the curve obtained as the upper-right envelope of its points). We can see not only that ROADS is much more sensitive to the parameter choice than the NFA algorithm, but also that its overall performance in terms of recall is significantly worse, even with an optimal choice of its parameters.

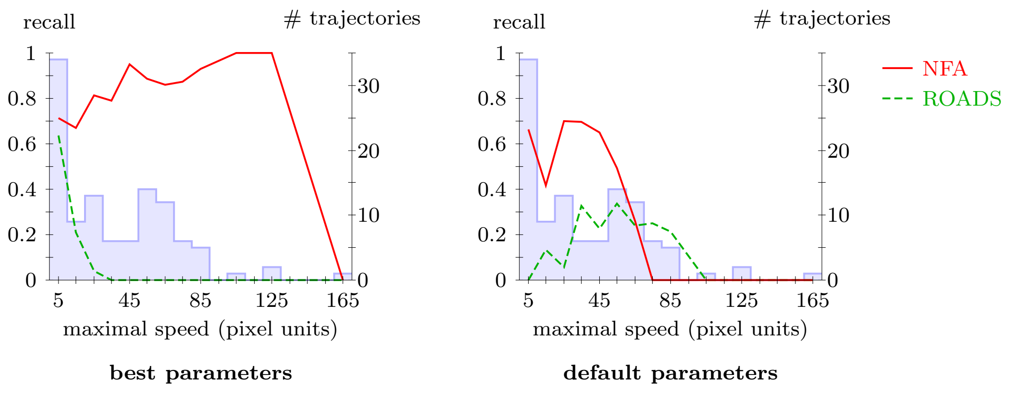
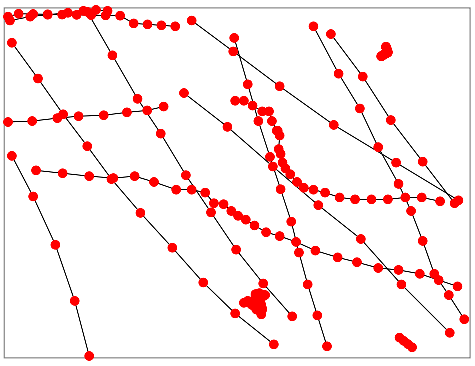
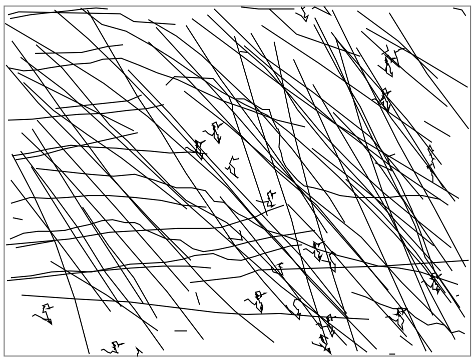


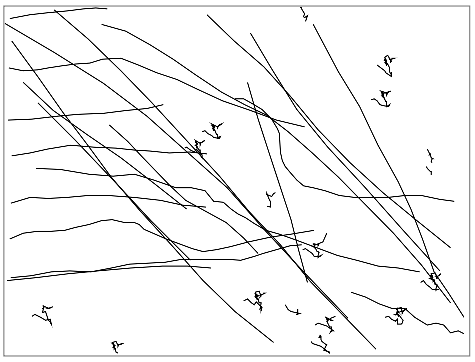
Figure 58: Detailed analysis of ROAD and NFA recall performances. The recall performances of ROADS and NFA algorithms on the *snow* sequence (as given in Table 6) are analyzed in function of the maximal trajectory speed, for both *best* (left) and *default* parameter settings. The possible values of the maximal trajectory speeds are divided into bins (horizontal axis), and the blue histograms indicate the number of corresponding actual trajectories. Then, the recall of each algorithm is analyzed inside each bin, in red for NFA and in dashed green for ROADS. As we can see, the ROADS algorithm does not manage to handle simultaneously trajectories with various speeds: the detection is focused either on trajectories with middle-range speeds (*default* setting), or on very slow trajectories (*best* setting). The NFA algorithm, which combines the trajectory smoothness and length into a single NFA criterion (hence avoiding a speed threshold), does not suffer from this dilemma, as it clearly appears on the left (*best* setting) graph.



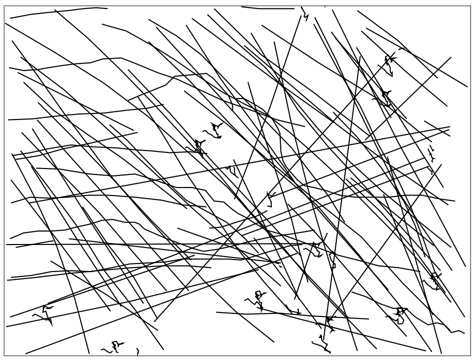
(1a) Sample trajectories from ground truth



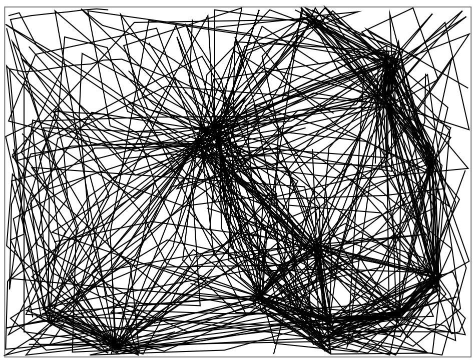
(1b) Ground truth



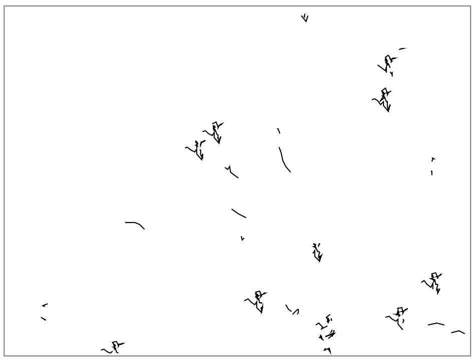
(2a) NFA default



(2b) NFA best



(3a) ROADS default



(3b) ROADS best

Figure 59: The two first figures represent the ground truth for the real data sequence. The first image is a subset of trajectories that have been extracted and whose successive points are represented as dots to give an idea of the objects speeds. The next two figures are the trajectories found by the NFA algorithm when using the best parameter (left) and the default one (right), and the last two figures are the trajectories found by ROADS when using the best parameters (left) and the default one (right). Clearly, the ROADS algorithm makes many false detections with default parameters, and very few detections with best parameters. On the contrary, the NFA algorithm already finds a large part of the real trajectories and almost no false detections (see Fig. 56) with the default parameter, and find many additional trajectories when using the best parameter.

mode	algorithm	recall (fg, bg)	precision
default	ROADS	0.09 (0.19, 0.00)	0.08
	NFA	0.58 (0.48, 0.68)	0.91
best	ROADS	**0.43** (0.22, 0.63)	**0.91**
	NFA	**0.76** (0.79, 0.74)	**0.90**

Table 6: **Performances of ROADS and** NFA **algorithms on the snow sequence.** The ROADS and NFA algorithms are run on the *snow* sequence, both with their *default* and *best* settings, and their performances are analyzed in terms of recall and precision. In order to permit a more accurate analysis, separate recall scores are also computed by considering separately fast foreground (fg) objects (snowflakes) and the almost static background (bg) objects (stains on the background door). As we can see, the comparison is clearly in favor of the NFA algorithm. With the *best* settings, the obtained precision is roughly the same, but the ROADS algorithm is unable to achieve an interesting detection rate on the foreground objects, which results in a poor overall recall.

4.6 CONCLUSION

The NFA algorithms allow the detection trajectories in very high levels of noise when low false detections rates are required. Their only parameter, the NFA threshold, has a natural value $\varepsilon = 1$ that renders these algorithms almost parameterless.

The presentation of the algorithms on two-dimensional data is in no way a limitation, and the NFA algorithms extend smoothly to data in any dimension.

The high computational and memory costs – particularly in the presence of holes – might be mitigated by the easily parallelizable nature of the algorithm, or by developing a set of heuristics to limit the number of trajectories to check.

The NFA criteria might also be used to filter the output of any algorithm, thus filtering out many false detections.

There remains two main limitations to the proposed algorithms. First, the greedy nature of the algorithms does not allow to find a global optimum in space for the trajectories, since they have to be considered in isolation, and second, non-uniform noise will lead to false detections, that might or might not be easy to filter out after processing.

A main contribution of this work, beside giving algorithms to extract trajectories in noisy sequences, is to give an objective criterion to measure the quality of a particular trajectory in a sequence, that might be used as a filter to ameliorate the results of any trajectory detection method.

A possible way to improve detection results would be to use additional information to make the decision, for instance, if the points have intensities. However, we found that adapting the NFA criterion to handle the position and an intensity uniformly is not straightforward, and this might be a future development. It is also possible to develop a criterion for the sum-cost rather than the max-cost of accelerations.

Open-source code for the algorithms is available on the website of the authors, containing both a simple Python implementation and an optimized one in C.

Geometrical composition
inspired by the assignment maps of WRAP

5 The WRAP algorithm

CONTENTS

Extracting the optimal trajectory set that explains a sequence frames containing points is a difficult combinatorial problem – NP-hard in effect, and no algorithm is able to solve it efficiently without approximations.

As we remarked earlier however, the problem can be simplified by dropping some constraints: in the previous chapter, we described the ASTRE algorithm that sequentially detect trajectories, extracting each one in isolation and thus alleviating the computational cost of the simultaneous optimization of all the trajectories. An orthogonal approach consists in tracking all trajectories *at once* but only on a short time span – by optimizing the point correspondences between two frames for example.

We will now focus on the two-frame *point correspondence problem* – assigning to each detected point in a frame its corresponding point in the next frame – which has been a very well studied problem in computer vision. This problem however always seems to be either very simple or very complex: in simple cases (very few objects that move slowly and are far apart from each other), even the most basic algorithm will give the correct answer, whereas in complex cases (many objects moving quickly, close to each other and in presence of spurious detections), close objects do not always correspond to each other, and one needs additional clues to recover the assignment: considering an extended temporal scope to observe the coherence of the object motion on several frames to eliminate spurious measurements for instance, or adding some features to the point, as an object shape, color or texture for example (see Figure 60).

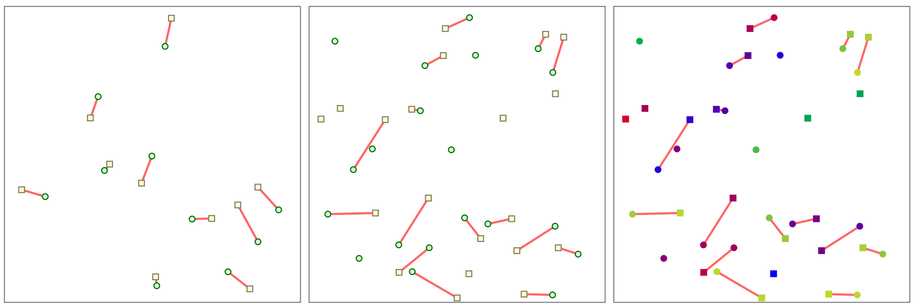


Figure 60: The left figure shows a simple assignment problem (squares and circles represents the points in each image, and the correct pairings are linked by an edge). On this kind of problems, even the most basic correspondence algorithm will usually find the correct underlying assignment when given the points in each frame. The middle figure shows a more complex assignment problem: many points moving quickly and some spurious detections. In this case, additional information is required to correctly solve the problem (one might for instance add the observed color of the objects to the data as in the right picture to disambiguate some choices).

The most obvious candidate algorithm for the point correspondence problem, when no prior on the point displacement is known, is that associating each point to its nearest neighbor. When the motion of the points is centered and Gaussian, the most likely assignment can be recovered efficiently as the one minimizing the sum of squared displacements thanks to the Hungarian algorithm introduced in Kuhn, 1955. In practical applications however, one often has to cope with missing and spurious detections. For instance, Salari and Sethi, 1990 propose to use placeholders for the missing points and use thresholds on the distances to allow spurious detections. Rangarajan and Shah, 1991 use a nearest neighbor approach with a prioritization of the decisions to start by linking points who only have one possible good successor.

In order to better understand the limits of the two-frame point correspondence problem, and to measure the performances of the classical algorithms, we introduce the WRAP criterion – for *Weighted Recall And Precision* – that maximizes the weighted sum $(\text{recall} + \lambda \cdot \text{precision})$ for some $\lambda > 0$, assuming a general probability model of the input data. When λ varies, this criterion defines the optimal curve of a point correspondence algorithm in the (recall, precision) space.

The point correspondence problem and some classical algorithms to solve it are introduced in Section 5.1, followed in section 5.2 by the description of the optimal WRAP criterion and of its behavior on typical use cases, as well as an efficient MCMC-based algorithm to approximate it. In Section 5.3, we compare the performances of the classical algorithms to the WRAP optimal performance, and we study the feasibility of using WRAP as a practical algorithm.

5.1 GENERAL DEFINITIONS AND CLASSICAL APPROACHES FOR THE POINT CORRESPONDENCE PROBLEM

We assume that objects have been detected in the images as points (without features) in the domain $\Omega = [0,1] \times [0,1]$ using a possibly imperfect detector, and the task is to find the correspondences between the points detected in two images representing the same scene with a slight variation in the camera pose or in the objects locations.

Hypothesis 1 (Correspondence uniqueness)**.** *We assume that an object corresponds to at most one detected point, and a detected point corresponds to at most one object (but there might also be some spuriously detected points or some missing detections).*

To alleviate the notations we will now fix the N detections in the first image $s_X = (\mathbf{x}_1, ..., \mathbf{x}_N) \in \Omega^N$ and the M detections in the second image $s_Y = (\mathbf{y}_1, ..., \mathbf{y}_M) \in \Omega^M$.

Definition 7 (Assignment)**.** *An* assignment a *of size* $|a| = k$ *between* s_X *and* s_Y *is a set of* k *pairs* $a = \{i_1 \to j_1, ..., i_k \to j_k\}$*, where* $\forall p,\ 1 \leqslant i_p \leqslant N$ *and* $1 \leqslant j_p \leqslant M$*, such that the pairings are disjoint: for any two distinct indices* p *and* q, $i_p \neq i_q$ *and* $j_p \neq j_q$*. The assignment might sometimes be abusively denoted* $a = \{\mathbf{x}_{i_1} \to \mathbf{y}_{j_1}, ..., \mathbf{x}_{i_k} \to \mathbf{y}_{j_k}\}$ *to make it clear that we are pairing the points* $\{\mathbf{x}_i\}_i$ *to the points* $\{\mathbf{y}_j\}_j$*. The* domain *of the assignment is* $\mathrm{dom}(a) = \{i_1, ..., i_k\}$*, and its* image *is* $\mathrm{im}(a) = \{j_1, ..., j_k\}$*. We denote by* $\mathbb{A}$ *the set of all assignments, and* $\mathbb{A}_k$ *is the set of assignments of size* k*.*

Definition 8 (Pairing costs)**.** *The* pairing costs *between* s_X *and* s_Y *is a set of positive real numbers* $\{c_{ij} \in \mathbb{R}^+\}_{1 \leqslant j \leqslant M}^{1 \leqslant i \leqslant N}$ *that measures the similarity between the points, where a low value of* c_{ij} *indicates a similarity between the detections* $\mathbf{x}_i$ *and* $\mathbf{y}_j$*.*

If we are tracking points slowly moving across an image sequence, the cost of each pairing $\mathbf{x}_i \to \mathbf{y}_j$ will usually be related to the distance $c_{ij} = \|\mathbf{y}_j - \mathbf{x}_i\|$ that the underlying point has traveled between the two frames.

Given such pairing costs $\{c_{ij}\}_{ij}$, the point correspondence problem consists in reconstructing the assignment a such that the detection $\mathbf{x}_i$ corresponds to (the same underlying object as) the detection $\mathbf{y}_{a(i)}$. To assert that a does indeed link the detections corresponding to the same object in both images, it is usually required that it optimizes a global criterion on the costs, some classical examples of which we will now discuss.

5.1.1 Classical approaches for the point correspondence problem in a simple setting

The most commonly found criterion, when there are no spurious nor missing detections (and hence $N = M$), is to find an assignment of size N minimizing the sum of the individual pairing costs, which is known as the *linear assignment problem* (LAP, see Kuhn, 1955):

$$a \in \operatorname*{arg\,min}_{a \in \mathbb{A} \mathrm{st.} |a| = N} \sum_{i=1}^{N} c_{i a(i)}.$$

In the case where the motion of the points is Gaussian and the costs are the squared distances, this would return the most likely assignment.

An alternate possibility is to minimize the maximum of individual pairing costs, which is known as the *bottleneck assignment problem* (BAP, see Gross, 1959 and Garfinkel, 1971):

$$a \in \operatorname*{arg\,min}_{a \in \mathbb{A} \mathrm{st.} |a| = N} \max_{1 \leqslant i \leqslant N} c_{i a(i)}.$$

Since the sum of the pairings cost and the maximal pairing cost will be used often in the next chapters, we give them a simple notation:

Definition 9 (Linear assignment cost)**.** *We denote* $c_{\text{sum}}(a) = \sum_{i \to j \in a} c_{ij}$ *the linear cost of the assignment* a*.*

Definition 10 (Bottleneck assignment cost)**.** *We denote* $c_{\max}(a) = \max_{i \to j \in a} c_{ij}$ *the bottleneck cost of the assignment* a*.*

Efficient algorithms that compute an optimal solution to those problems have been developed. For the linear assignment problem, the most famous are certainly the Hungarian algorithm[Kuhn, 1955] of complexity $\mathcal{O}(N^4)$ and the shortest-path augmenting algorithms (see Jonker and Volgenant, 1987) of complexity $\mathcal{O}(N^3)$, whereas for the bottleneck assignment problem, the threshold algorithm (see Garfinkel, 1971) provides a solution in $\mathcal{O}(N^4)$ time.

5.1.2 Classical approaches in presence of spurious and missing detections

In practical applications, the points detected in an image generally do not correspond uniquely to the points detected in the other because of missing and spurious detections. The assignment problem solvers as presented above are not tailored for such general situations as they look for a one-to-one correspondence between points in the two images, and will therefore find correspondences where there are none, for instance between two (necessarily unrelated) spurious detections.

To mitigate this problem, one might want to use additional knowledge: if there are reasons to discard all pairings that have a cost greater than a certain value c_{thre} (for instance, physical reasons like a limited speed), the number of matching possibilities might be greatly restricted. Similarly, if an estimate of the true number of points detected in both images is available, it might be used to limit the search to the optimal assignments having this particular size.

Thresholded costs

If some pairings can be discarded because of physical reasons for instance, the correspondence problem can be solved in the presence of outliers by transforming it into a *minimal-cost maximal-cardinality assignment problem*, that is, denoting $k_{\max}$ the size of a maximal-cardinality assignment (the largest possible size of an assignment between the two images containing only possible pairings), finding an assignment a of size $k_{\max}$ minimizing

$$a \in \operatorname*{arg\,min}_{a \in \mathbb{A} \text{ st. } |a| = k_{\max}} \sum_{i \in \text{dom}(a)} c_{i a(i)},$$

where the arg min is on assignments a containing only possible pairings.

Given a scalar c_{thre}, the *thresholded-costs assignment problem* is the minimal-cost maximal-cardinality assignment problem where all pairings of cost greater than c_{thre} have been discarded. The classical algorithms solving the linear assignment problem can be adapted to solve the thresholded-costs assignment problem, and there is a similar definition in the case of the bottleneck assignment problem.

Phantom points

Another widespread approach to alleviate the shortcomings of the assignment problem in the presence of spurious or missing detections consists in adding a "phantom point" [Salari and Sethi, 1990] $\tilde{\mathbf{x}}$ in the second image for each point $\mathbf{x}$ in the first image, and conversely a phantom point $\tilde{\mathbf{y}}$ in the first image for each point $\mathbf{y}$ in the second image.

Usually, for a given $c_{\text{thre}} > 0$, the pairing costs of the initial problem are extended to the pairings of the new problem by setting the cost of a pairing involving a phantom point to c_{thre}.

Thus, if a point appears, disappears, or is a spurious detection having no realistic correspondence in the other image (ie. there is no matching point at a distance less than c_{thre}), rather than trying to assign it to any other existing point, this solution will assign it to its phantom for a cost of c_{thre}, effectively discarding it from the assignment (see Figure 61).

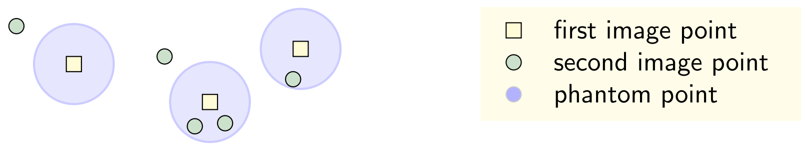


Figure 61: The phantom-point assignment enables us to alleviate some shortcomings of the general assignment problem in presence of spurious and missing points. For each detected point in an image, a "phantom point" is added in the other image, that can be linked to any point for a given cost c_{thre} (for clarity only the phantom points for the first image points have been shown on this figure; they are represented by a disc of area c_{thre}). Thus, if a point in an image is missing in the other, the solution will tend to link the point to its phantom (see the left first image point for example), rather than to an arbitrary point in the other image.

We thus obtain the *phantom assignment problem*: given a cost threshold c_{thre}, find an assignment a minimizing

$$\begin{aligned} a &\in \arg\min_{a \in \mathbb{A}} \sum_{i \in \text{dom}(a)} c_{i a(i)} + (N + M - |a|) \cdot c_{\text{thre}} \\ &= \arg\min_{a \in \mathbb{A}} \sum_{i \in \text{dom}(a)} \left(c_{i a(i)} - c_{\text{thre}}\right) \end{aligned}$$

How do the thresholded-costs and phantom assignment problems relate? Denoting a_t the solution to the thresholded-costs assignment problem with maximal cost c_{thre} and a_p the solution to the phantom assignment problem with phantom costs c_{thre}, we necessarily have $|a_p| \leqslant |a_t|$ (because a_p contains only edges having a cost less than c_{thre} and a_t is such an assignment of maximal cardinality) and $c_{\text{sum}}(a_p) \leqslant c_{\text{sum}}(a_t)$ (because a_t is a candidate solution to the phantom assignment problem). The phantom approach thus permits to remove some edges deemed unrealistic from the assignment using the value of the parameter c_{thre}, adaptively selecting the correct size of a realistic assignment (which might often be less than the maximal-cardinality of an assignment). We illustrate this on a toy example (see Figure 62), where the thresholded-costs assignment will give a maximal-size solution, whereas under some circumstances, the phantom solution will choose to remove a pairing from the assignment.

k-*cardinality assignments*

If an estimate of the true number k of corresponding points in both images (that is, ignoring spurious points and occlusions) is available, one can assume that the correct pairings are those of minimal cost, and that there is exactly k correct pairings. Thus, a good guess at

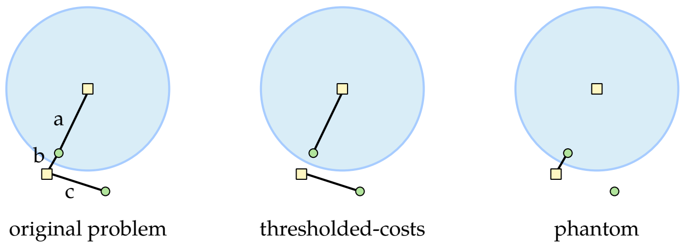


Figure 62: The squares are the point of the first image, the dots are the point of the second image. The blue disc has a radius of c_{thre}, and a, b and c are the costs of the corresponding edges. The illustration above is in the case where $b + c_{thre} < a + c$ (otherwise both solutions are equal to the middle figure).

solving the problem might be to find the solution of the k-*cardinality assignment problem*: find an assignment a of size k minimizing

$$a \in \underset{a \in \mathbb{A} \text{ st. } |a|=k}{\arg\min} \sum_{i \in a} c_{i a(i)}$$

Nearest neighbor

Let us end this brief review of classical approaches to the point correspondence problem with probably the simplest and most widespread of all algorithms: assigning each point to its nearest neighbor in the other image. However simple the idea might be, giving it a mathematical definition that is both clear and has a unique meaning actually proves to be cumbersome. Indeed, if $\mathbf{x}$ is some point in the first image, and $\mathbf{y}$ its nearest neighbor in the second image, it might very well be the case that the nearest neighbor of $\mathbf{y}$ in the first image is not $\mathbf{x}$, breaking the apparent symmetry of the definition. Furthermore, there might actually be more than one point in the first image whose nearest neighbor is $\mathbf{y}$, resulting in an assignment conflict for $\mathbf{y}$.

We therefore give an algorithmic definition to the nearest-neighbor solution, thus avoiding complex mathematical notations to alleviate definition problems: sorting the pairings $\mathbf{x}_i \to \mathbf{y}_j$ by increasing cost c_{ij}, we consider them in order, and we add in turn each pairing $\mathbf{x}_i \to \mathbf{y}_j$ to the assignment if neither $\mathbf{x}_i$ nor $\mathbf{y}_j$ appear in a previously added pairing.

When using this algorithm on data containing spurious and missing detections, one will often restrict the allowed pairings using a threshold on their cost.

5.1.3 Modeling data

In order to understand the performances and the limits of the point correspondence algorithms, we need to define a model of the input data.

We suggested above that in the case of point tracking, the distance between the points might be a good indicator of the fact that they represent the same detected object. However, we might as well choose the squared distance, or any function thereof. Similarly, for some applications the choice of the algorithms parameters might be obvious: in the case of point tracking again, a physical bound on the speed of the objects motion gives a physical meaning to the parameter c_{thre} of the thresholded-costs algorithm. The meaning of this parameter becomes however less obvious when using the phantom assignment algorithm, as there exists an interplay between the roles of c_{thre} as a maximal cost for a pairing and as a penalty for spurious detections. In this case, a proper model of the input data might help us decide which pairing costs and which parameter settings suit the data best.

The model that appeared the most natural to us to generate data pairs of N and M detections consists in drawing uniformly at random N points in the first image, splitting them randomly in two sets representing the correct feature points and the spurious detections, and displacing the feature points slightly to obtain the corresponding detections in the second image. Finally, we complete the second image up to M points by uniformly drawing additional spurious detections. There are at most $\min(N, M)$ points that are paired in both images, and for a given application, we might have a rough idea of the proportion of those $\min(N, M)$ points that are indeed paired, as well as a rough idea of the magnitude of the displacement. We formalize precisely this generative model below:

Definition 11 (Generative model)**.** *Let $\Omega = [0, 1] \times [0, 1]$ be the domain of the images. We are given two parameters, p_R the ratio of correctly detected points, and σ^2 the variance of the displacement magnitude,*

1. *choose an integer K following a binomial distribution $\mathcal{B}(\min(N, M), p_R)$,*
2. *draw $X_1, ..., X_K$ uniformly in Ω,*
3. *for $1 \leqslant i \leqslant K$, displace the point X_i by defining $Y_i = X_i + N_i$, where N_i is a Gaussian variable of null mean and variance σ^2,*
4. *draw $X_{K+1}, ..., X_N$ and $Y_{K+1}, ..., Y_M$ uniformly in Ω.*

The result of the generative model is the tuple $G = (K, \bar{X}, \bar{Y})$ where $\bar{X} = (X_1, ..., X_N)$ and $\bar{Y} = (Y_1, ..., Y_M)$.

Of course, when we are given a pair of point sets, we do not know the order in which the points have been generated in each one (and thus the correspondences), nor the number of real points K that correspond to each other in the images, and we thus introduce an observational model describing this loss of information:

Definition 12 (Observational model)**.** *Assuming $G = (K, \bar{X}, \bar{Y})$ has been generated using the model of definition 11,*

1. *we randomly choose σ_X a permutation of $[1, N]$ and σ_Y a permutation of $[1, M]$,*
2. *we define $S_X = (X_{\sigma_X(1)}, ..., X_{\sigma_X(N)})$ and $S_Y = (Y_{\sigma_Y(1)}, ..., Y_{\sigma_Y(M)})$,*

The result of the observational model is $O = (K, \sigma_X, \sigma_Y, S_X, S_Y)$.

Definition 13 (Generated assignment)**.** *Given $O = (K, \sigma_X, \sigma_Y, S_X, S_Y)$ observed by the above model, we define $A = \{\sigma_X(1) \to \sigma_Y(1), ..., \sigma_X(K) \to \sigma_Y(K)\}$ the* generated assignment *(of size $|A| = K$) between S_X and S_Y.*

Given two point sets s_X and s_Y generated by the above model (realizations of S_X and S_Y), we want to recover the underlying generated assignment a by using our knowledge on the generational model. We now show that we can interpret two classical ways of so doing – the maximum likelihood and the maximum a posteriori approaches – as the solutions to variants of the phantom assignment problem described in the previous section.

5.1.4 The maximum likelihood and maximum a posteriori detectors

Maximum likelihood

Given two point sets s_X and s_Y, the maximum likelihood approach to assignment detection consists in choosing the assignment a that maximizes the likelihood

$$\ell(S_X = s_X, S_Y = s_Y \mid A = a) \quad = \quad \prod_{i \to j \in a} \frac{1}{2\pi\sigma^2} e^{-\|\mathbf{y}_j - \mathbf{x}_i\|^2/2\sigma^2},$$

which, equivalently, is the assignment minimizing

$$\sum_{i \to j \in a} \left(\|\mathbf{y}_j - \mathbf{x}_i\|^2 - 2\sigma^2 \ln \frac{1}{2\pi\sigma^2} \right) \tag{5.1}$$

and is hence the solution of the phantom assignment problem with parameter

$$c_{\text{thre}} = 2\sigma^2 \ln(1/2\pi\sigma^2),$$

where the pairing costs are the squared distances $c_{ij} = \|\mathbf{y}_j - \mathbf{x}_i\|^2$. We note that the value of p_R does not come into play, as the assignments are considered a priori equally probable in the maximum likelihood approach.

Maximum a posteriori

The maximum a posteriori approach makes use of the likelihood of each particular assignment size, and detects the assignment a that maximizes

$$\ell(A = a \mid S_X = s_X, S_Y = s_Y) \quad = \quad \frac{\ell(S_X = s_X, S_Y = s_Y \mid A = a) \cdot \mathbb{P}(A = a)}{\ell(S_X = s_X, S_Y = s_Y)},$$

where $\ell(S_X = s_X, S_Y = s_Y)$ is a constant that we can ignore. Observing that

$$\mathbb{P}(A = a) \quad = \quad \underbrace{\binom{N \wedge M}{|a|} \cdot {p_R}^{|a|} (1 - p_R)^{N \wedge M - |a|}}_{\mathbb{P}(|A| = |a|)} \cdot \underbrace{\frac{|a|!(N - |a|)!(M - |a|)!}{N!M!}}_{\mathbb{P}(A = a \mid |A| = |a|)}$$

this amounts to detecting the assignment a maximizing (for $0 < p_R < 1$)

$$\mathbb{P}(a) \cdot \ell(s_X, s_Y \mid a) \quad \propto \quad (N \vee M - |a|)! \left(\frac{p_R}{2\pi\sigma^2(1 - p_R)} \right)^{|a|} \cdot \exp\left(- \sum_{i \to j \in a} \frac{\|\mathbf{y}_j - \mathbf{x}_i\|^2}{2\sigma^2} \right)$$

where the proportionality constant only depends on the values of N, M and p_R, which amounts again to minimizing over all possible a

$$\sum_{i \to j \in a} \left(\|\mathbf{y}_j - \mathbf{x}_i\|^2 - 2\sigma^2 \ln \frac{p_R}{2\pi\sigma^2(1 - p_R)} \right) - 2\sigma^2 \ln \Big((N \vee M - |a|)! \Big). \tag{5.2}$$

This criterion resembles the phantom assignment problem, with an added penalty that is not increasing linearly with the size of a.

If $p_R = 0$ (ie. all the points are noisy detections), the only possibility is obviously the empty assignment. If $p_R = 1$ (ie. no noise points), the maximum *a posteriori* assignment is the maximum cardinality assignment minimizing the sum of squared distances, and thus it is the assignment found by the linear assignment problem in the case where the costs are the squared distances.

5.1.5 Algorithms

Most of the algorithms we described above, and those we will describe later, share a common building block: an algorithm solving the k-cardinality linear assignment problem, that is, returning $a_0, ..., a_{N \wedge M}$ such that

$$\forall 0 \leqslant k \leqslant N \wedge M, \quad |a_k| = k \quad \text{and} \quad a_k \in \arg\min_{a \in \mathbb{A}_k} \sum_{i \to j \in a} c_{ij}.$$

Such an algorithm is detailed in appendix A that has computational complexity $\mathcal{O}(\min(N, M)NM)$. Similarly, an algorithm solving the k-cardinality bottleneck assignment problem is described in the same appendix and has computational complexity $\mathcal{O}(N^2M^2)$.

Using these algorithms, we obtain immediately the solution $a_{N \wedge M}$ to the general linear (resp. bottleneck) assignment problem. A slight variation of the aforementioned algorithm enables us to disallow some pairings and thus solve the thresholded-costs assignment problem without an increase in complexity. Finally, solving the phantom assignment problem with parameter c_{thre} consists simply in returning the assignment $a_{\hat{k}}$ such that

$$\hat{k} = \arg\min_k \sum_{i \to j \in a_k} c_{ij} - k \cdot c_{\text{thre}}.$$

5.2 WRAP (WEIGHTED RECALL AND PRECISION)

In order to measure the performances of the assignment algorithms and to define an optimal bound on those performances, we use the precision and the recall criteria: given a pair of point sets for which the real underlying assignment is a, an algorithm detecting the assignment $\tilde{a}$ has a recall and a precision defined by

$$\text{recall} = \frac{\text{\# of correct pairings}}{\text{\# of real pairings}} \quad \text{and} \quad \text{precision} = \frac{\text{\# of correct pairings}}{\text{\# of found pairings}},$$

where a pairing is real if it belongs to a, is found if it belongs to $\tilde{a}$ and is correct if it belongs to $a \cap \tilde{a}$. The recall measures how much of the real assignment has been detected by the method, and the precision tells whether most of the extracted pairings are correct or not. More precisely, and extending this definition to empty assignments,

$$r(\tilde{a}, a) = \mathbb{1}_{a=\varnothing}\mathbb{1}_{\tilde{a}=\varnothing} + \mathbb{1}_{a \neq \varnothing}\frac{|a \cap \tilde{a}|}{|a|}, \quad p(\tilde{a}, a) = \mathbb{1}_{\tilde{a}=\varnothing}\mathbb{1}_{a=\varnothing} + \mathbb{1}_{\tilde{a} \neq \varnothing}\frac{|a \cap \tilde{a}|}{|\tilde{a}|}.$$

Using the fairly general data generation model described in the previous section, and this performance measure, we are now able to give a quantified meaning to our intuition that the point correspondence problem is always too simple or too complex. By comparing the performances of the most local greedy algorithm like the Nearest-Neighbor, and those of the global Maximum-Likelihood criterion for instance, we observe in Figure 63 that no matter the complexity of the underlying assignment problem (as measured by the variance of the point displacement), both criteria have essentially the same performances, and it seems thus mostly useless to use a global criterion.

To study whether those algorithms are indeed optimal or not, we now introduce the optimal WRAP criterion, that will give us a theoretical bound on the performances of the point correspondence algorithms.

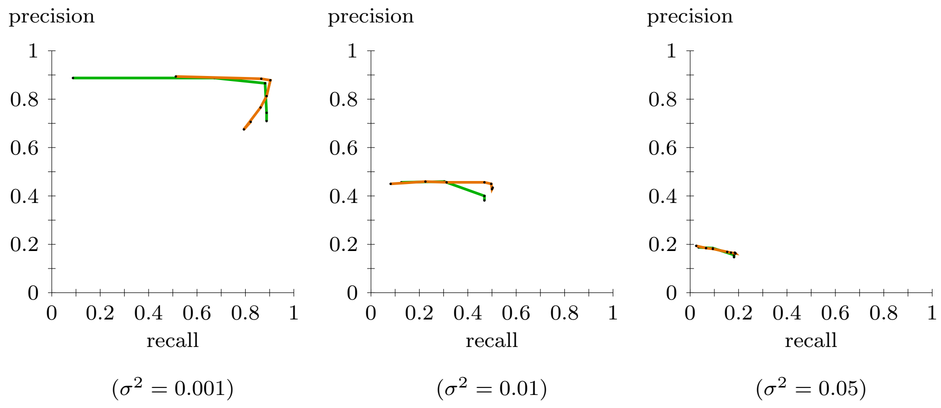


Figure 63: (**Equivalence of algorithms performances**) Mean precision and recall of the Maximum Likelihood (brown) and Nearest-Neighbor (green) algorithms on the generative model with parameters $N = M = 20$, $p_R = 0.8$ and various values of σ^2. We vary the algorithms parameters c_{thre} of NN and σ^2 of ML, and the values are averages of 500 runs. We observe that no matter the complexity of the problem (the variance of the point displacement amplitude), the global ML algorithm and the greedy local NN algorithm yield similar performances. The performances of the Maximum A Posteriori algorithm are equivalent.

5.2.1 The WRAP optimal criterion

We will now always assume that the input data has been obtained through the generative and observational models of definitions 11 and 12: we remind the reader that the generative model results in a tuple $G = (K, \bar{X}, \bar{Y})$ describing the points in the assignment as the first K points of the sequences $\bar{X}$ and $\bar{Y}$, the other points being spurious detections ; the observational model simulates the loss of information by transforming this tuple into $O = (K, \sigma_X, \sigma_Y, S_X, S_Y)$, where S_X and S_Y are shuffled versions of the points in each point set, obtained using the permutations σ_X and σ_Y ; and finally the generated assignment is $A = \{\sigma_X(1) \to \sigma_Y(1), ..., \sigma_X(K) \to \sigma_Y(K)\}$.

Given the observed points $s_X = (\mathbf{x}_1, ..., \mathbf{x}_N)$ and $s_Y = (\mathbf{y}_1, ..., \mathbf{y}_M)$ in both images – a realization of S_X and S_Y, we are able to compute for each possible assignment a the probability that it is indeed the realization of the real underlying generated assignment A. Any completely specified algorithm running on this data would return an assignment $\tilde{a}$, and we can now compute the expected recall and precision of the algorithm on all possible generated data that results in the *exact same* observed points (from now on in this section, expectations are computed on observations O of generated data G that result exactly in the observed point sets s_X and s_Y, and we will denote $\mathbb{E}[X]_{s_X,s_Y} = \mathbb{E}[\delta_{S_X=s_X,S_Y=s_Y} \cdot X]_{(G,O)}$)

$$\begin{aligned}
\mathbb{E}[r(\tilde{a}, A)]_{s_X,s_Y} &= \sum_{a \in \mathbb{A}} \ell(A = a \mid S_X = s_X, S_Y = s_Y) \cdot r(\tilde{a}, a), \text{ and} \\
\mathbb{E}[p(\tilde{a}, A)]_{s_X,s_Y} &= \sum_{a \in \mathbb{A}} \ell(A = a \mid S_X = s_X, S_Y = s_Y) \cdot p(\tilde{a}, a),
\end{aligned}$$

Now, assuming the realization of the input data is still fixed, let S be the (finite) set of all such points $\mathbb{E}[(r(\tilde{a}, A), p(\tilde{a}, A)]_{s_X,s_Y}$. Using Lagrange multipliers, we know that maximizing r when we fix the value of p to a given α is equivalent to maximizing $r + \lambda p$ for a certain λ (that depends on the value of α). This essentially corresponds to drawing a line (whose orientation depends on λ) above the point set, and to pull it towards the origin until it crosses a point (see figure 64). Thus, the pairs (r, p) that are obtained when maximizing $r + \lambda p$ as λ varies in $\mathbb{R}_+$ define a boundary on the pairs that can be reached by any assignment algorithm.

Of course, the WRAP criterion is only able to attain the points on the hull, and not any arbitrary location on the segments that join them, but a simple modification of the WRAP criterion to a probabilistic criterion would enable us to do so: let λ_1 and λ_2 be two values of λ such that $(r_{\lambda_1}, p_{\lambda_1})$ and $(r_{\lambda_2}, p_{\lambda_2})$ are two consecutive points on the hull ; by choosing a real number $\theta \in [0, 1]$, and defining a probabilistic algorithm that returns with probability θ the assignment detected by WRAP with the parameter λ_1, and with probability $1 - \theta$ the assignment detected by WRAP with the parameter λ_2, we are obviously able to attain an average performance at any location on the segment joining $(r_{\lambda_1}, p_{\lambda_1})$ and $(r_{\lambda_2}, p_{\lambda_2})$.

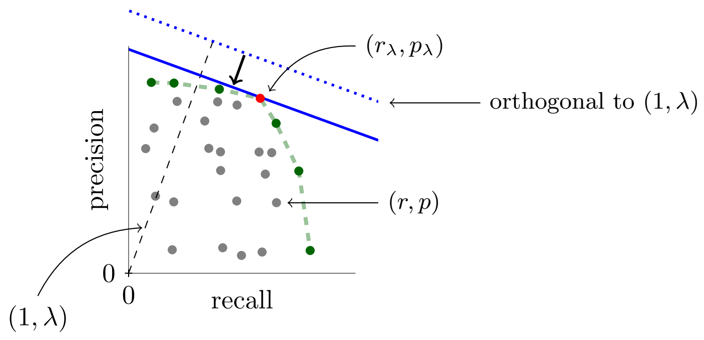


Figure 64: Given observed data s_X and s_Y generated using the generative model of definition 11, the points (r, p) displayed above represent the possible expected recall and precision of any assignment detection algorithm: $r = \mathbb{E}[r(\tilde{a}, A)]_{s_X, s_Y}$ and $p = \mathbb{E}[p(\tilde{a}, A)]_{s_X, s_Y}$ where $\tilde{a}$ is the detection made by the algorithm on s_X and s_Y, and the expectation is taken on all the real underlying generated assignments A that can explain s_X and s_Y. The points on the "hull" (green dashed line) are optimal in the sense that for any point in the set, there is a point in the hull having both a larger recall and a larger precision. They correspond to the set of points $\{\mathbb{E}[(r(\tilde{a}_\lambda, A), p(\tilde{a}_\lambda, A)]_{s_X, s_Y}\}_{\lambda > 0}$, where $\tilde{a}_\lambda$ maximizes the expected value $\mathbb{E}[r(\tilde{a}_\lambda, A) + \lambda p(\tilde{a}_\lambda, A)]_{s_X, s_Y}$ – that is, the dot product of $(1, \lambda)$ and $(\mathbb{E}[r(\tilde{a}_\lambda, A)]_{s_X, s_Y}, \mathbb{E}[p(\tilde{a}_\lambda, A)]_{s_X, s_Y})$. Geometrically, the point $(r_\lambda, p_\lambda) = (\mathbb{E}[r(\tilde{a}_\lambda, A)], \mathbb{E}[p(\tilde{a}_\lambda, A)]_{s_X, s_Y})$ is the first one that is crossed by a line perpendicular to the vector $(1, \lambda)$ placed above the points, when that line is pulled toward the origin.

This idea can be translated mathematically to yield the optimal WRAP criterion that describes this boundary: for any $\lambda > 0$, the line perpendicular to the vector $(1, \lambda)$ crosses the point (r_λ, p_λ) when it is pulled towards the origin, where r_λ and p_λ are the recall and precision of the assignment $\tilde{a}_\lambda$ maximizing the dot product of $(1, \lambda)$ and $(\mathbb{E}[r(\tilde{a}_\lambda, A)]_{s_X, s_Y}, \mathbb{E}[p(\tilde{a}_\lambda, A)]_{s_X, s_Y})$; in other words, the "hull" of the set S is the set of points $\{\mathbb{E}[(r(\tilde{a}_\lambda, A), p(\tilde{a}_\lambda, A)]_{s_X, s_Y}\}_{\lambda > 0}$, where $\tilde{a}_\lambda$ maximizes the expected value of the WRAP (*Weighted Recall And Precision*) criterion

$$\mathbb{E}_\lambda(\tilde{a}_\lambda) = \mathbb{E}[r(\tilde{a}_\lambda, A) + \lambda \cdot p(\tilde{a}_\lambda, A)]_{s_X, s_Y}. \tag{5.3}$$

By both optimizing for the recall and the precision, WRAP is thus an optimal criterion, in the sense that for any algorithm whose average performance on data sets generated using the generative and observational models is (r, p), there exists a value λ such that the WRAP algorithm with parameter λ would subsume this result: it would yield an average performance $r_\lambda \geqslant r$ and $p_\lambda \geqslant p$.

5.2.2 The WRAP algorithm

In order to define an algorithm to compute the WRAP criterion, let us rewrite it in the form of a sum cost on the pairings in $\tilde{a}_\lambda$:

$$\begin{aligned}
\mathbb{E}_\lambda(\tilde{a}_\lambda) &= \mathbb{E}[r_\lambda(\tilde{a}_\lambda, A) + \lambda p_\lambda(\tilde{a}_\lambda, A)]_{s_X, s_Y} \\
&= \mathbb{E}\left[\mathbf{1}_{A=\varnothing}\mathbf{1}_{\tilde{a}_\lambda=\varnothing} + \mathbf{1}_{A\neq\varnothing}\frac{|\tilde{a}_\lambda \cap A|}{|A|} + \lambda\left(\mathbf{1}_{A=\varnothing}\mathbf{1}_{\tilde{a}_\lambda=\varnothing} + \mathbf{1}_{\tilde{a}_\lambda\neq\varnothing}\frac{|\tilde{a}_\lambda \cap A|}{|\tilde{a}_\lambda|}\right)\right]_{s_X, s_Y} \\
&= \mathbf{1}_{\tilde{a}_\lambda=\varnothing} \cdot (1+\lambda)\mathbb{E}[\mathbf{1}_{A=\varnothing}]_{s_X, s_Y} + \\
&\quad \mathbf{1}_{\tilde{a}_\lambda\neq\varnothing}\mathbb{E}\left[\mathbf{1}_{A\neq\varnothing}\sum_{p\in\tilde{a}_\lambda}\frac{\mathbf{1}_{p\in A}}{|A|} + \frac{\lambda}{|\tilde{a}_\lambda|}\sum_{p\in\tilde{a}_\lambda}\mathbf{1}_{p\in A}\right]_{s_X, s_Y} \\
&= \mathbf{1}_{\tilde{a}_\lambda=\varnothing} \cdot (1+\lambda)\mathbb{E}[\mathbf{1}_{A=\varnothing}]_{s_X, s_Y} + \\
&\quad \sum_{p\in\tilde{a}_\lambda}\mathbb{E}\left[\mathbf{1}_{A\neq\varnothing}\frac{\mathbf{1}_{p\in A}}{|A|}\right]_{s_X, s_Y} + \frac{\lambda}{|\tilde{a}_\lambda|}\sum_{p\in\tilde{a}_\lambda}\mathbb{E}\left[\mathbf{1}_{p\in A}\right]_{s_X, s_Y} \\
&= \mathbf{1}_{\tilde{a}_\lambda=\varnothing} \cdot (1+\lambda)\mathbb{E}[\mathbf{1}_{A=\varnothing}]_{s_X, s_Y} + \sum_{p\in\tilde{a}_\lambda}\left(r_p + \frac{\lambda}{|\tilde{a}_\lambda|}p_p\right)
\end{aligned}$$

where

$$\begin{aligned}
r_p &= \mathbb{E}\left[\mathbf{1}_{A\neq\varnothing}\frac{\mathbf{1}_{p\in A}}{|A|}\right]_{s_X, s_Y} = \sum_{a\ni p}\frac{1}{|a|}\cdot\ell(A=a \mid S_X=s_X, S_Y=s_Y), \\
p_p &= \mathbb{E}\left[\mathbf{1}_{p\in A}\right]_{s_X, s_Y} = \sum_{a\ni p}\ell(A=a \mid S_X=s_X, S_Y=s_Y),
\end{aligned} \tag{5.4}$$

and (using lighter notations)

$$\ell(a \mid s_X, s_Y) \quad \propto \quad {p_R}^{|a|}(1-p_R)^{N\wedge M-|a|}\cdot\frac{(N\vee M-|a|)!}{(N\vee M)!}\cdot\prod_{i\to j\in a}\frac{1}{2\pi\sigma^2}e^{-\|\mathbf{y}_j-\mathbf{x}_i\|^2/2\sigma^2}.$$

Maximizing the expectation $\mathbb{E}_\lambda(\tilde{a}_\lambda)$ thus amounts to solving $\min(N, M)$ optimization problems $\{P_k\}_{1\leqslant k\leqslant N\wedge M}$ where the solution of the problem P_k is the assignment of size k between s_X and s_Y of maximal sum cost, and the cost of each pairing $p = (i \to j)$ is $c^k_{ij} = r_p + (\lambda/k)p_p$ (see Algorithm 8).

The routine `maximum_cost_kcard_assignment` solving the optimization problems $\{P_k\}_k$ is an immediate modification of the k-cardinality (minimum-cost) linear assignment problem solver described in appendix A.

Remark 2 (Modified precision for simpler computations)**.** *We could replace the precision*

$$p(\tilde{a}, a) = \frac{\textit{\# of correct pairings}}{\textit{\# of found pairings}} = \mathbf{1}_{\tilde{a}=\varnothing}\mathbf{1}_{a=\varnothing} + \mathbf{1}_{\tilde{a}\neq\varnothing}\frac{|a \cap \tilde{a}|}{|\tilde{a}|}$$

by the variant

$$p'(\tilde{a}, a) = 1 - \frac{\textit{\# of incorrect pairings}}{N \wedge M} = 1 - \frac{|a| - |a \cap \tilde{a}|}{N \wedge M},$$

```
Algorithm: WRAP
input  : $s_X, s_Y$ the points detected in each image
input  : $p_R, \sigma^2$ the parameters of the generative model
input  : $\lambda$ the WRAP parameter
output: $\tilde{a}_\lambda$ the optimal assignment for parameter $\lambda$

$\mathbb{A} \leftarrow$ set of assignments between $s_X$ and $s_Y$

foreach pairing p between $s_X$ and $s_Y$ do
    $r_p \leftarrow \sum_{a \in \mathbb{A} \text{ s.t. } a \ni p} \frac{1}{|a|} \ell(a \mid s_X, s_Y)$
    $p_p \leftarrow \sum_{a \in \mathbb{A} \text{ s.t. } a \ni p} \ell(a \mid s_X, s_Y)$
end

for $k = 1 \to \min(N, M)$ do
    $\{c_p^k\}_p = \{r_p + (\lambda/k) p_p\}_p$
    $a_k \leftarrow$ maximum_cost_kcard_assignment($k, \{c_p^k\}_p$)
    $c_k \leftarrow \sum_{p \in a_k} c_p^k$
end

if $\max_k c_k < (1+\lambda)\ell(A = \varnothing \mid s_X, s_Y)$ then
    $\tilde{a}_\lambda \leftarrow \varnothing$
else
    $\hat{k} \leftarrow \arg\max_k c_k$
    $\tilde{a}_\lambda \leftarrow a_{\hat{k}}$
end

return $\tilde{a}_\lambda$
```

Algorithm 8: Computation of the optimal WRAP assignment. The computation of the coefficients r_p and p_p is exact, making the algorithm unpractically slow when the size of the point sets s_X and s_Y becomes too large.

in which case the expression of $\mathbb{E}_\lambda(\tilde{a}_\lambda)$ *becomes*

$$\mathbb{E}_\lambda(\tilde{a}_\lambda) \quad = \quad \lambda\mathbb{E}[1]_{s_X,s_Y} + \mathbf{1}_{\tilde{a}_\lambda=\varnothing} \cdot \mathbb{E}[\mathbf{1}_{A=\varnothing}]_{s_X,s_Y} + \sum_{p\in\tilde{a}_\lambda} \mathbb{E}\left[\mathbf{1}_{A\neq\varnothing}\frac{\mathbf{1}_{p\in A}}{|A|} - \frac{\lambda}{N\wedge M}\mathbf{1}_{p\notin A}\right]_{s_X,s_Y}$$

and results in a somewhat less complex algorithm requiring only the computation of one *maximum cost assignment problem, since the costs of the pairings no longer depend on the size of* $\tilde{a}_\lambda$. *We will however not use this variant, as the original definition of the precision is the most widely accepted one.*

5.2.3 Behavior of WRAP

To better understand what makes WRAP an optimal criterion, let us study its behavior on some simple cases, and compare it with the behavior of the classical approaches like ML and MAP.

1 point vs. k *points*

First, in the simple case where both point sets contain exactly one point, $s_X = \{A\}$ and $s_Y = \{a\}$, the only two possible assignments are $\{\varnothing, (A \to a)\}$, and the corresponding values of $\mathbb{E}_\lambda$ are

$$\begin{aligned} \mathbb{E}_\lambda(\varnothing) &= (1+\lambda)\cdot\ell(A=\varnothing \mid s_X, s_Y), \quad \text{and} \\ \mathbb{E}_\lambda(A \to a) &= (1+\lambda)\cdot\ell(A=\{1\to 1\} \mid s_X, s_Y). \end{aligned}$$

Thus, $\tilde{a}_\lambda$ maximizes $\mathbb{E}_\lambda$ whenever it maximizes $\ell(A = \tilde{a}_\lambda \mid s_X, s_Y)$; in other words – and not surprisingly – λ has no effect, and WRAP and MAP have exactly the same behavior in this case, and so does ML up to a multiplicative factor. More precisely, WRAP and MAP associate the two points if and only if they are separated by a distance $\delta \leqslant (2\sigma^2 \ln \frac{p_R}{2\pi\sigma^2(1-p_R)})^{1/2}$, and ML associates them if and only if $\delta \leqslant (-2\sigma^2 \ln 2\pi\sigma^2)^{1/2}$.

Generally, the WRAP and MAP behavior are the same for all the assignments between one point in the first image and k points in the second image (or the converse).

2 points vs. 2 points

When the point sets both contain several points however, the WRAP algorithm adopts a much more complex behavior, by taking into account the fact that pairings can belong to several different assignments or variable probability. This allows WRAP in essence to define which pairings have inherently more risk – are more ambiguous – than others, and choose whether or not to detect them.

Let us observe what happens when both sets contain two points $s_X = \{A, B\}$ and $s_Y = \{a, b\}$, using abbreviated notations:

$$\begin{aligned}
\mathbb{E}_\lambda(\varnothing) &= (1+\lambda)\cdot\ell(A=\varnothing)\\
\mathbb{E}_\lambda(A\to a) &= (1+\lambda)\cdot\ell(A=\{A\to a\}) + (1/2+\lambda)\cdot\ell(A=\{A\to a, B\to b\})\\
\mathbb{E}_\lambda(A\to b) &= (1+\lambda)\cdot\ell(A=\{A\to b\}) + (1/2+\lambda)\cdot\ell(A=\{A\to b, B\to a\})\\
\mathbb{E}_\lambda(B\to a) &= (1+\lambda)\cdot\ell(A=\{B\to a\}) + (1/2+\lambda)\cdot\ell(A=\{A\to b, B\to a\})\\
\mathbb{E}_\lambda(B\to b) &= (1+\lambda)\cdot\ell(A=\{B\to b\}) + (1/2+\lambda)\cdot\ell(A=\{A\to b, B\to a\})\\
\mathbb{E}_\lambda(A\to a, B\to b) &= (1+\lambda/2)\cdot\left[\ell(A=\{A\to a\}) + \ell(A=\{B\to b\})\right]\\
&\qquad +(1+\lambda)\cdot\ell(A=\{A\to a, B\to b\})\\
\mathbb{E}_\lambda(A\to b, B\to a) &= (1+\lambda/2)\cdot\left[\ell(A=\{A\to b\}) + \ell(A=\{B\to a\})\right]\\
&\qquad +(1+\lambda)\cdot\ell(A=\{A\to b, B\to a\})
\end{aligned}$$

As is obvious from the criterion $r+\lambda p$ that we optimize, small values of λ will favor large assignments to increase the recall, while large values of λ will favor smaller assignments. We now detail the above equations in the case where $\lambda = 0$, $\lambda \gg 1$ and $\lambda = 1$ to show this clearly.

We assume here for the clarity of the explanation that the configuration of the points in both sets is such that only the assignments $\{A\to a\}$ and $\{A\to a, B\to b\}$ are possible solutions of the problem, the other assignments having a smaller $\mathbb{E}_\lambda$ value, and we only discuss when WRAP will include one or two pairings in the optimal solution.

- If $\lambda = 0$, we have:

$$\begin{aligned}
\mathbb{E}_0(A\to a) &= \ell(A=\{A\to a\}) + \frac{1}{2}\cdot\ell(A=\{A\to a, B\to b\})\\
\mathbb{E}_0(A\to a, B\to b) &= \ell(A=\{A\to a\}) + \ell(A=\{B\to b\}) + \ell(A=\{A\to a, B\to b\})
\end{aligned}$$

  and since $\mathbb{E}_0(A\to a, B\to b) \geqslant \mathbb{E}_0(A\to a)$, WRAP will always detect both pairings when they are more likely than the empty assignment, hence yielding a high recall.

- If $\lambda \gg 1$, we have:

$$\begin{aligned}
\mathbb{E}_\lambda(A\to a) &= \lambda\cdot\left[\ell(A=\{A\to a\}) + \ell(A=\{A\to a, B\to b\})\right] + \mathcal{O}_{\lambda\to+\infty}(1)\\
\mathbb{E}_\lambda(A\to a, B\to b) &= \lambda\cdot\left[\frac{1}{2}\ell(A=\{A\to a\}) + \frac{1}{2}\ell(A=\{B\to b\}) + \ell(A=\{A\to a, B\to b\})\right]\\
&\quad +\mathcal{O}_{\lambda\to+\infty}(1)
\end{aligned}$$

  and WRAP will tend to detect only one pairing when λ grows very large, as either $\ell(A = \{A\to a\})$ or $\ell(A=\{B\to b\})$ will be greater than $\frac{1}{2}\ell(A=\{A\to a\}) + \frac{1}{2}\ell(A=\{B\to b\})$, thus yielding a high precision.

- If $\lambda = 1$, we have:

$$\begin{aligned}
\mathbb{E}_1(A\to a) &= 2\cdot\ell(A=\{A\to a\}) + \frac{3}{2}\cdot\ell(A=\{A\to aB\to b\})\\
\mathbb{E}_1(A\to a, B\to b) &= \frac{3}{2}\cdot\left[\ell(A=\{A\to a\}) + \ell(A=\{B\to b\})\right] + 2\cdot\ell(A=\{A\to a, B\to b\})
\end{aligned}$$

and WRAP will try to find a compromise between detecting one or two pairings ; more precisely, $\mathbb{E}_1(A \to a) \geqslant \mathbb{E}_1(A \to a, B \to b)$ if and only if $\ell(A = \{A \to a, B \to b\}) \leqslant \ell(A = \{A \to a\}) - 3 \cdot \ell(A = \{B \to b\})$. Thus, if $\ell(A = \{B \to b\}) \geqslant \frac{1}{3}\ell(A = \{A \to a\})$, WRAP will prefer to detect both pairings to maximize $\mathbb{E}[r + p]$.

- And generally for any $\lambda > 0$, $\mathbb{E}_\lambda(A \to a) \geqslant \mathbb{E}_\lambda(A \to a, B \to b)$ if and only if

$$\ell(A = \{A \to a, B \to b\}) \leqslant \lambda \cdot \ell(A = \{A \to a\}) - (2 + \lambda) \cdot \ell(A = \{B \to b\})$$

and thus, the pairings that WRAP will choose depends on the value of $\ell(A = \{A \to a, B \to b\})$ in relation to a combination of the values of $\ell(A = \{A \to a\})$ and $\ell(A = \{B \to b\})$.

Visualizing the effect of λ

As we marked above, optimizing the expected value $r + \lambda p$ can be interpreted as maximizing the expected projection of the point (r, p) on the vector $(1, \lambda)$. We can visualize this by generating random assignment problems that we solve using the WRAP algorithm, and project the solution on the (recall, precision) plane for different values of λ (see Figure 65).

We observe very concretely that, as expected, the WRAP criterion tends to detect high-precision assignments when λ is large, and high-recall assignments when it is small.

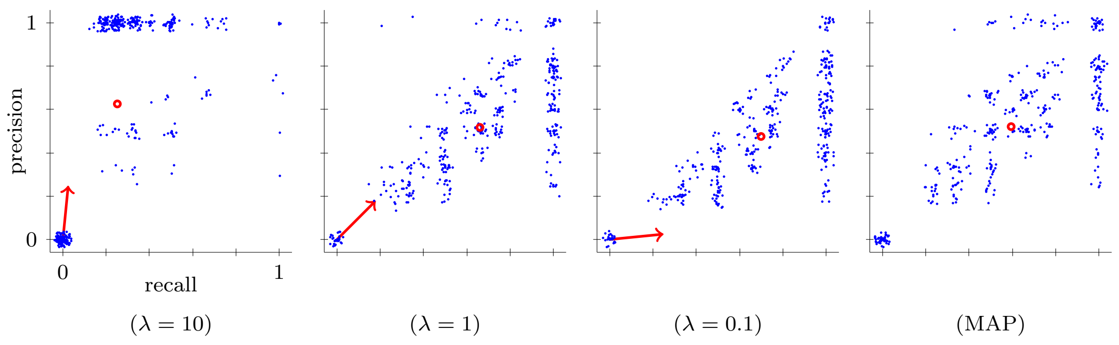


Figure 65: We generated 400 random assignment problems using the generative model where $N = M = 7$, $p_R = 0.5$ and $\sigma^2 = 10^{-2}$, we recovered the corresponding assignments using the WRAP algorithm with $\lambda = 10$ (left), $\lambda = 1$ (middle), $\lambda = 0.1$ (right) and the exact parameters p_R and σ^2, and we plot the recall and precision in each case. Note that the points have been randomized by adding a small displacement to their coordinates to show their density. The red circle shows the mean recall and precision, and the vector shows the direction of $(1, \lambda)$. The WRAP algorithm maximizes the expected value of $r + \lambda \cdot p$, that is, the projection of the mean on the vector $(1, \lambda)$. We observe that most of the projected assignments are in the upper left part when λ is high (high precision, low recall), and in the lower right part when λ is small (low precision, high recall). As a reference, we show in gray the recall and precision obtained using the MAP algorithm with the correct p_R and σ^2 (they are the same points in all images).

5.2.4 WRAP assignment maps

Although the behavior of WRAP is easy to understand in very simple cases, it becomes more difficult to predict when the number of points increase, or when there are ambiguities leading to many possible assignments to be candidate solutions, rather than just two as we assumed when discussing the effect of λ for simplicity in the previous section.

To better understand the behavior of the WRAP criterion in more general cases, it is useful to observe concrete results on several point configurations and to compare it to the MAP criterion. To get a sense of the changes in the decisions as the point configurations are locally modified, we display "assignment maps" where all the points but one are fixed, and observe how the WRAP and MAP decisions evolve as the free point moves in the image.

For example, the assignment map of Figure 66 compares the behavior of the WRAP algorithm for several values of λ, and for two points in each image. The two points detected in the first image are the red dots A and B, and the first point detected in the second image is the blue dot a ; Those points are fixed, while the second point of the second image – denoted b – is free, and for each of its possible positions on a discrete grid, the color of the map represents the corresponding found assignment. Keep in mind that there *is no ground truth* for the assignment in those assignment maps. Although we assume that the two point sets have been generated using the generative model, we don't know the true underlying generated assignment. Rather, WRAP assumes that any possible underlying generated assignment could be the real one, each having a certain probability of being the real one that can be computed from the position of the points. The goal of WRAP is not to find the assignment maximizing the expectation of a function of the recall and the precision.

We observe here that, as predicted, a small value for λ will always yield an assignment containing a maximal number of pairings, while a large value will always yield an assignment containing only one pairing. When $\lambda = 1$, the WRAP and MAP behavior are essentially similar. In this case, the area around B where WRAP chooses to include the two pairings is slightly larger than that of MAP, but this might no longer be the case for a slightly higher value of λ.

In Figure 67, we observe the behavior of WRAP for several values of λ as the point a comes closer to A. We note several interesting features.

First, no matter the value of λ, the area of the images where the WRAP criterion makes a detection rather than returning the empty assignment is essentially the same – although as we noted above, a small λ favors detections with many pairings, and a large λ favors detections with only one pairing.

When λ is small, and when a and A are close (last row), the blue disc representing the assignment $\{A \to a, B \to b\}$ starts enclosing the red region $\{A \to b, B \to a\}$, and in some cases – for instance when the point b is in the position denoted by c – the most likely assignment in the probabilistic generative model is the latter, but WRAP will still select the former. This can be explained because *individually*, the pairing $A \to a$ is still more likely than $A \to c$, and is thus chosen to favor a (more likely) real detection, but since the precision is not relevant, we can still add the pairing $B \to c$ in case it also happens to be real.

When λ is very large, other interesting phenomena appear. First, notice the change that happens when b takes the positions d and e (third line, last column). When b is far from B, at location b for instance, WRAP naturally pairs $\{A \to a\}$ only. But when a stays in place and b moves closer to B (at location e for instance), WRAP will not add the pairing $B \to b$ to the assignment, but detect this pairing alone – because it is very likely, and thus helps improve the precision. If we now observe what happens when b takes the position f in the last row, we see that WRAP counter-intuitively detects $\{B \to a\}$ rather than $\{A \to a\}$, although it is less likely. This can be explained because WRAP detects an ambiguity: it does not know whether f is a spurious detection, or is an abnormally high motion of one of the points ; and in this latter case, it most likely comes from A. Thus in order to favor the precision, WRAP decides to leave this ambiguous pairing out of the assignment, but it also decides that, since if it is the

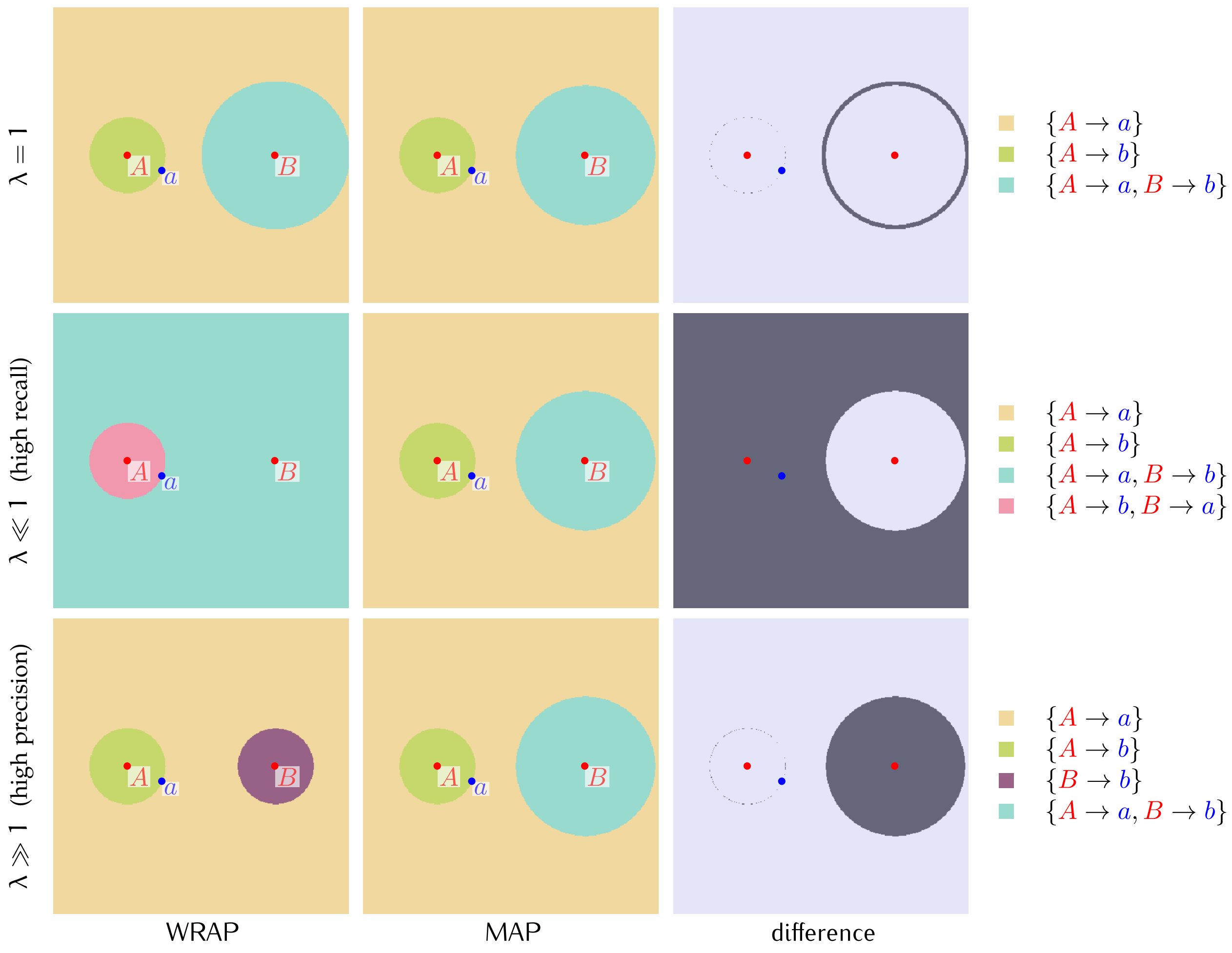


Figure 66: (**Effect of** λ) Assignment maps of WRAP and MAP where the computations have been made assuming that the generative model has parameters $p_R = 0.5$ and $\sigma^2 = 0.01$, for various values of λ. The two red dots A and B are the points in the first image, the blue dot a is the first point of the second image, and we visualize how the assignments change when the second point of the second image b moves across image, by displaying the assignment found for each possible position. The first WRAP map is read this way: when b is in the green disc around A, and thus closer from it than a, WRAP pairs $\{A \to b\}$, when b is in the blue disc, WRAP pairs $\{A \to a, B \to b\}$, and when b is outside these discs, WRAP pairs $\{A \to a\}$. The difference figure shows in dark gray the regions where the criteria differ. We observe that, as predicted, when λ is very small, WRAP tends to always detect both pairings to favor the recall, when λ is very large, WRAP tends to detect only one pairing to favor the precision, and when $\lambda = 1$, there is a compromise between detecting only one or both pairings. In this latter case we observe in the area for which the two pairings are detected is slightly larger for WRAP than for MAP, but this might no longer be the case for a slightly higher value of λ.

case that f comes from one of the point it must be A, it must also be less risky to assume that a comes from B.

The assignment maps of Figure 68 and 69 show the behavior of WRAP ($\lambda = 1$) when the parameter p_R of the generative model varies, and the behavior of WRAP when A and B are close, and a moves closer to A.

In particular, we observe that the behavior of WRAP evolves continuously, while the changes for MAP seem to happen by steps. Indeed, between the first and third row of Figure 69, the detection areas for MAP have barely changed, and only the assignment detected around B has evolved. In the meantime in the WRAP maps, the purple area around B has gradually shrunk, while the blue area has continuously expanded, to finally become even larger than the corresponding area in the MAP assignment map. This is shown clearly in Figure 70, where the areas of each assignment region is displayed as the point a of Figure 69 moves from the first row to the last.

Figure 71 shows examples of point configurations on which the WRAP criterion and the classical algorithms like MAP exhibit major differences. In many instances in those maps, WRAP makes a seemingly counter-intuitive choice because it takes into account the ambiguity in order to maximize both the recall and the precision. Note that in this case, the free point is e. Let us discuss for example the first row. If e is in the purple region around A, the pairing $\{A \to e\}$ is detected, but if e moves farther to the brown area for instance, this has not the expected effect of detecting less pairings, but *more*. In essence, WRAP says: if e is very close to A, I detect only this pairing in order to increase the precision, but if it gets farther, there is an ambiguity, because A might equally likely be displaced to e, b or a, and since I don't want to avoid making a detection because I want to optimize the recall, but I am not sure about my decision, I might as well add another decision which I am not sure of, as $\{B \to c\}$.

In other cases, the argument is essentially the same: for instance, still when e is in the brown area around A, one would expect that WRAP would detect the pairing $B \to b$ rather than $B \to c$, as the former is more likely in the probabilistic generative model. However, there is an ambiguity on the real motion of the point A, and although in the brown region it is more likely that A has moved to e, it could as well have moved to b or to a ; but certainly not c. It is thus less risky to assume that c comes from B. The same argument explains why in the second row, WRAP detects $\{A \to a, B \to c\}$ rather than $\{A \to a, B \to b\}$ when e is far from the other points.

Many other examples of the WRAP ambiguity handling can be found in the maps, and it is left as an amusement to the reader to find and explain them.

Overall, we see that the WRAP criterion tends to be able to arbitrate finely between ambiguities, in order to make the best decision, leading to more complex assignment maps that evolve gradually when the point locations are varying, while MAP tends to make simpler predictions, that are nonetheless close to those of WRAP.

5.2.5 WRAP computation using MCMC

Of course, the WRAP algorithm requires the computation of the weights $\{(r_p, p_p)\}_p$ from equation 5.4, and this is intractable in practice, as it takes $\mathcal{O}(|\mathbb{A}|)$ time, where $\mathbb{A}$ is the set of assignments between s_X and s_Y, which is very large for typical values of N and M (see Table 7).

It seems difficult to compute these coefficients efficiently, as they closely resemble the computation of a permanent – which is a #P-complete problem.

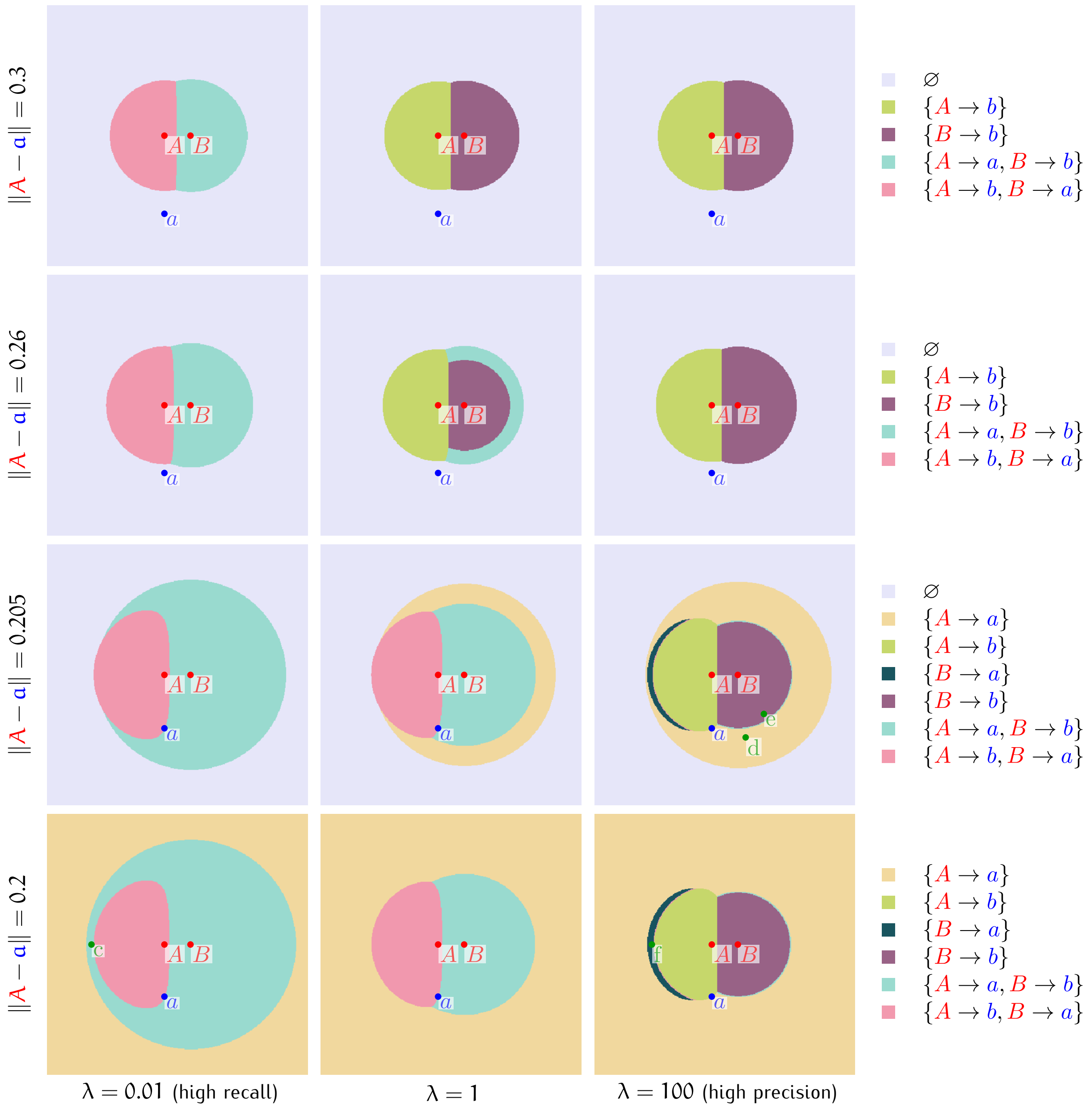


Figure 67: (**Effect of** λ) We observe the behavior of the WRAP algorithm for a generative model with parameters $p_R = 0.5$ and $\sigma^2 = 0.01$, when the configuration of the points varies, for various values of the parameter λ. As noted before, the WRAP criterion tends to detect many pairings when λ is small and fewer when λ is high, but it is interesting to note that the area where WRAP makes a detection stays essentially the same no matter the value of λ. We note other interesting effects taking place when b occupies some locations pinpointed by green labels. When λ is small, WRAP prefers to detect $\{A \to a, B \to b\}$ than $\{A \to b, B \to a\}$ although the latter is more likely in the probabilistic generative model, because it favors the recall (position c). When λ is large, WRAP will favor the precision in two counter-intuitive ways: when b occupies the position marked by d, WRAP detects the pairing $A \to a$, but when it occupies the position e, WRAP does not detect the two pairings as one would expect, but only the pairing $B \to b$. Finally, when b occupies the position f, WRAP detects $\{B \to a\}$ rather than $\{A \to a\}$, because f might be a statistically abnormal motion of the point A, and it is thus less risky to assume that a corresponds to B.

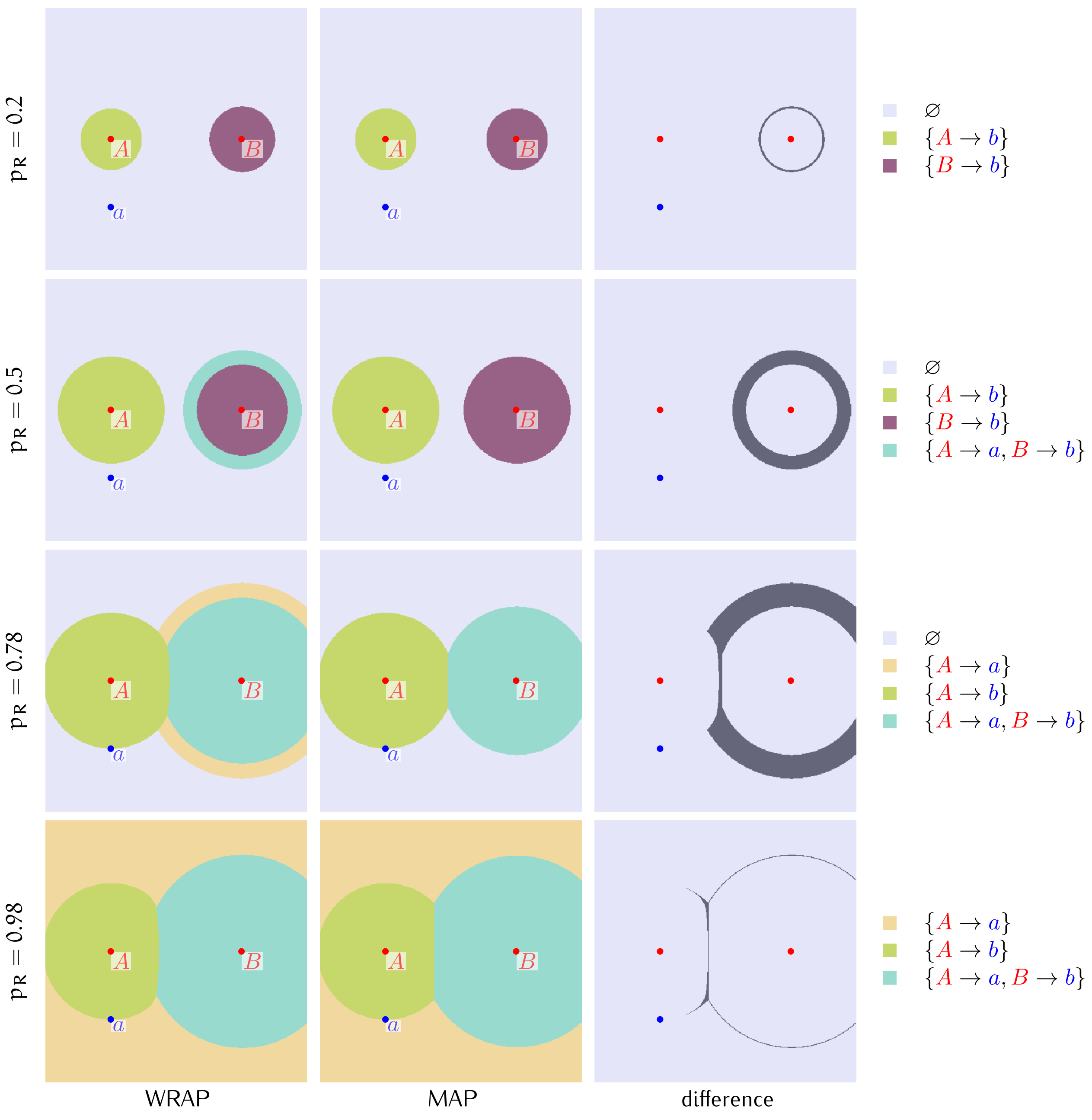


Figure 68: (**Effect of** p_R) Assignment maps for WRAP ($\lambda = 1$) and MAP where $\sigma^2 = 0.01$, as p_R varies. The changes of WRAP are continuous and gradual with respect to $\|A - a\|$, while the changes of MAP happen abruptly for specific values of $\|A - a\|$;

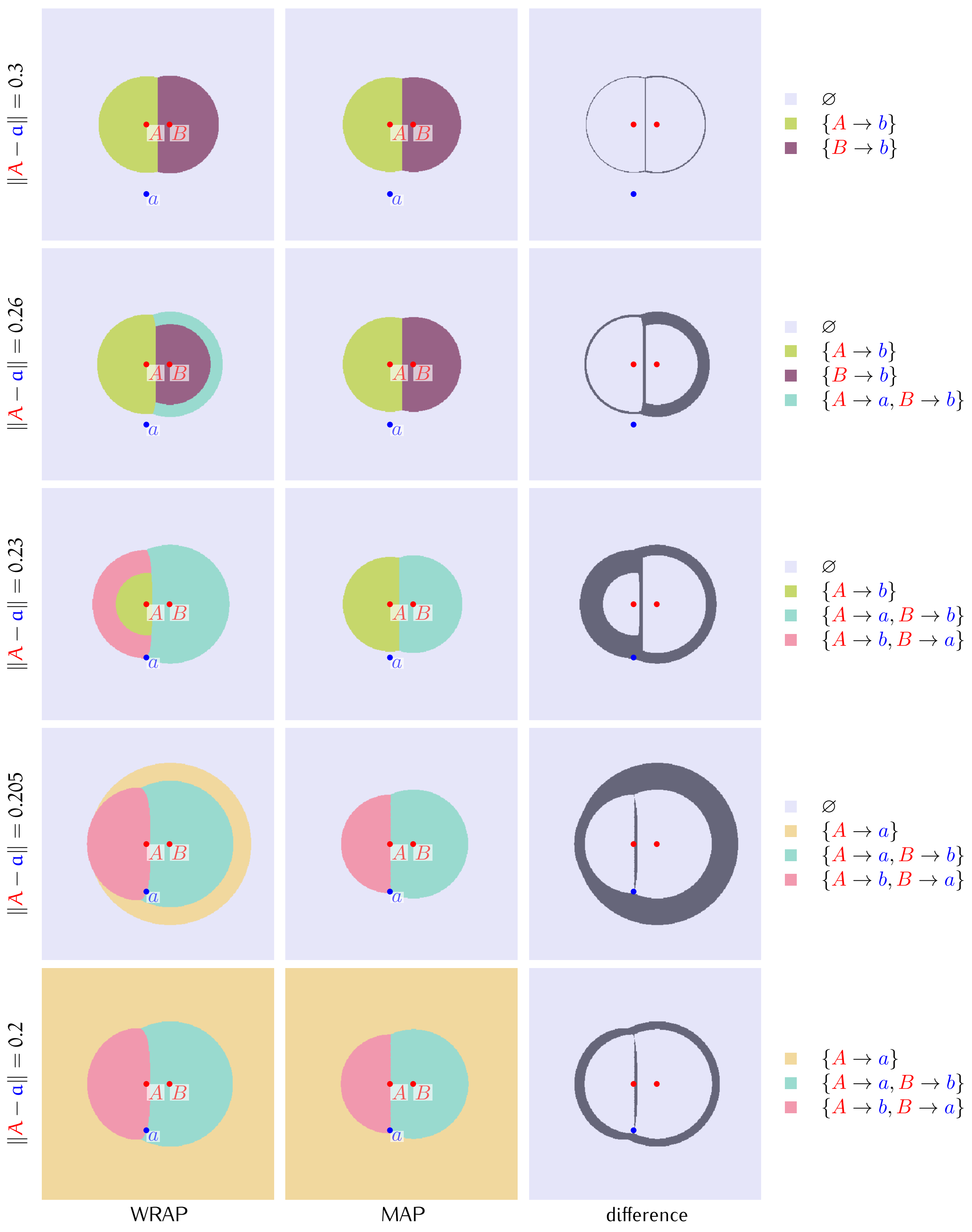


Figure 69: (**Assignment ambiguities**) Assignment maps for WRAP ($\lambda = 1$) and MAP where $p_R = 0.5$ and $\sigma^2 = 0.01$, as a moves closer to A. The changes of WRAP are continuous and gradual with respect to $\|A - a\|$, while the changes of MAP happen abruptly for specific values of $\|A - a\|$;

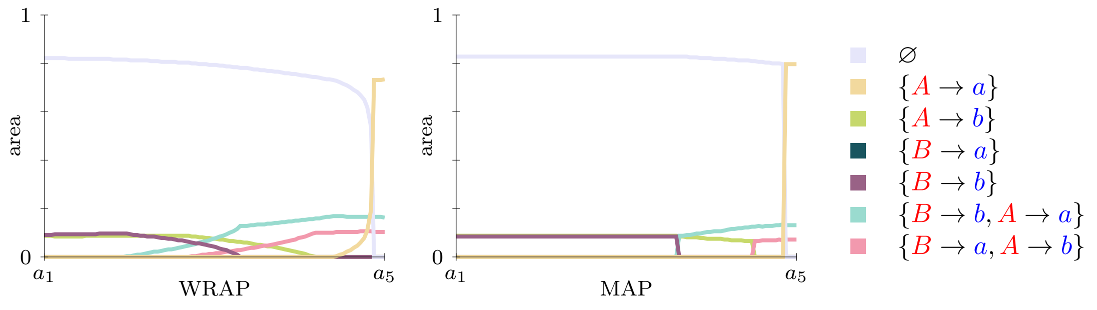


Figure 70: (**Continuous and discrete evolution of the assignment maps**) Evolution of the area of each assignment region for WRAP ($\lambda = 1$) and MAP where $p_R = 0.5$ and $\sigma^2 = 0.01$, as a moves closer to A (see Figure 69). The changes of WRAP are continuous and gradual, while the changes of MAP happen at discrete points in time.

N	2	4	8	16	32
$\lvert\mathbb{A}\rvert$	7	209	$\approx 1.4 \times 10^6$	$\approx 6.2 \times 10^{15}$	$\approx 1.7 \times 10^{39}$

Table 7: Number of assignments $\lvert\mathbb{A}\rvert = \sum_{k=0}^{N} \binom{N}{k}\binom{N}{k} k!$ between two sets containing N points.

However, noting that only a few assignments have a high probability and contribute to almost all of the probability mass (see table 8), it might be possible to obtain a good approximation of the coefficients by enumerating only the high probability assignments.

$r =$	0.95	0.99	0.999	0.9999	0.99999
$n =$	259 (257)	496 (491)	965 (964)	1617 (1634)	2456 (2474)
$n/\lvert\mathbb{A}\rvert \approx$	0.2%	0.4%	0.7%	1.2%	1.9%
$u =$	106 (95)	218 (194)	463 (403)	818 (698)	1298 (1083)
$u/\lvert\mathbb{A}\rvert \approx$	0.1%	0.2%	0.4%	0.6%	1.0%

Table 8: Displayed is the mean (standard variation) number n of the highest-probability assignments required to obtain various ratios r of the total probability mass for a problem with $N = M = 7$, $p_R = 0.5$ and $\sigma^2 = 10^{-2}$ (the total number of possible assignments in this case is $\lvert\mathbb{A}\rvert = 130\,922$). These values are averages computed on 500 random problems. As a reference, we also display the mean (standard deviation) number u of the highest-probability assignments required to obtain those ratios when the points detected in both images have been drawn uniformly at random (the same parameters p_R and σ^2 have been used to compute the probabilities). Although the standard variation is relatively high, the mass of the probability is still concentrated in a few of the highest probability assignments.

In the related case of the computation of a permanent, some recent work propose to approximate the value of the permanent in polynomial time using a Markov Chain Monte-Carlo algorithm [Jerrum, Sinclair, and Vigoda, 2004]. Although their work does not apply immediately to our specific problem, we will use a similar approach to compute the WRAP coefficients.

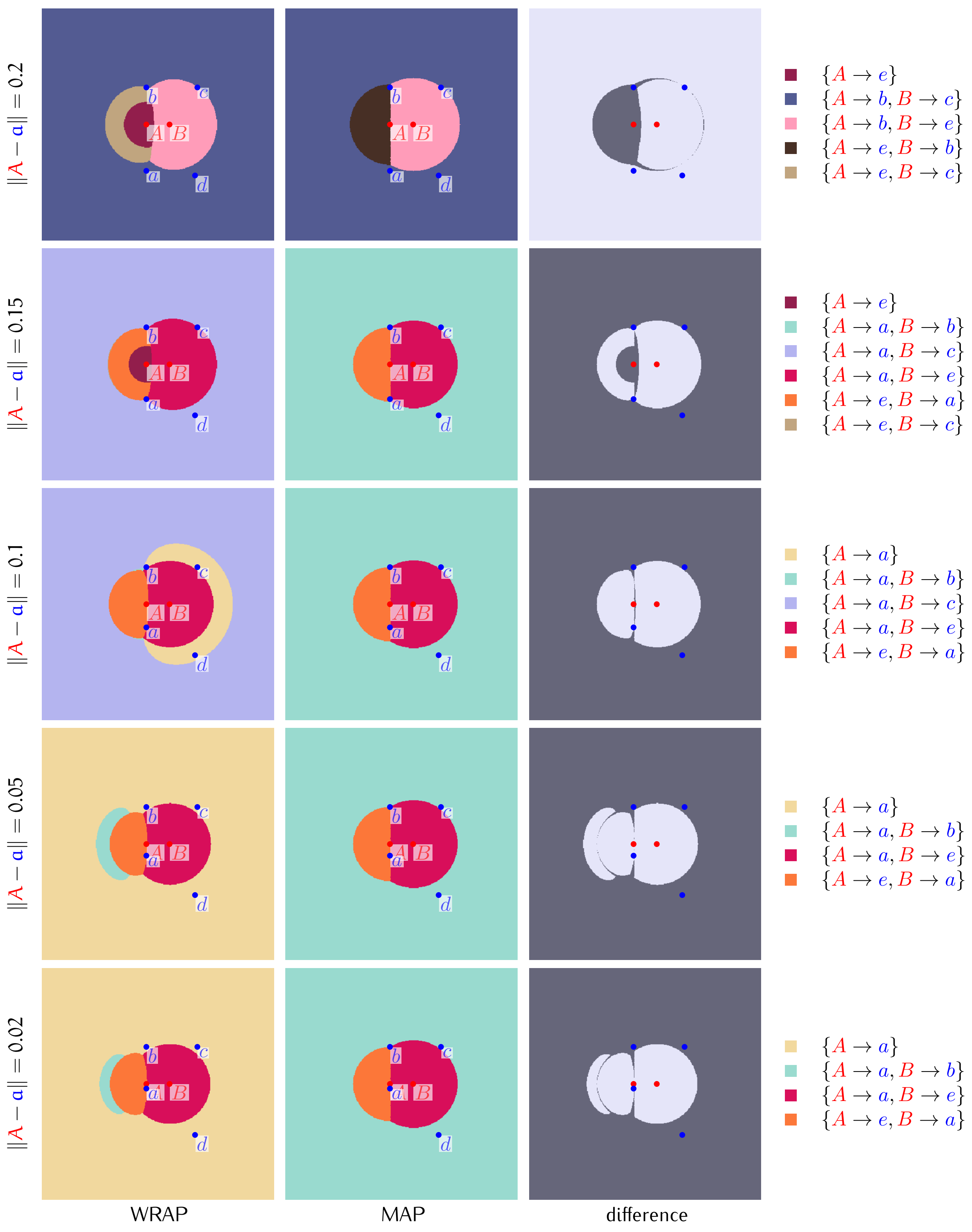


Figure 71: Assignment maps for WRAP and MAP between two point sets containing two points (A and B) and five points (a, b, c, d and the point e that defines the color of the map as it moves over the image area) where $p_R = 0.8$, $\sigma^2 = 0.01$ and $\lambda = 1$, as a moves toward A. In many instances in those maps, WRAP makes a seemingly counter-intuitive choice because it takes into account the ambiguity in order to maximize both the recall and the precision. Note that in this case, the free point is e. For instance, in the first row, one would expect WRAP to detect the pairing $B \to b$ rather than $B \to c$ when e is in the brown region around A ; but there is an ambiguity on the real displacement of the point A, and, although it might very well be the case that the real pairing is $A \to a$, it is most certainly not the case that A has moved to c, and thus, WRAP considers it less risky to propose the pairing $B \to c$.

We remind the reader that $\tilde{a}_\lambda$ maximizes

$$\mathbb{E}_\lambda(\tilde{a}_\lambda) = \mathbf{1}_{\tilde{a}_\lambda=\varnothing} \cdot (1+\lambda)\mathbb{E}[\mathbf{1}_{A=\varnothing}]_{s_X,s_Y} + \sum_{p\in\tilde{a}_\lambda} \left(r_p + \frac{\lambda}{|\tilde{a}_\lambda|} p_p \right)$$

where $r_p = \mathbb{E}\left[\mathbf{1}_{A\neq\varnothing}\frac{\mathbf{1}_{p\in A}}{|A|}\right]_{s_X,s_Y}$ and $p_p = \mathbb{E}\left[\mathbf{1}_{p\in A}\right]_{s_X,s_Y}$.

The WRAP algorithm computes $\tilde{a}_\lambda$ by first computing the values of r_p and p_p for all pairing p between s_X and s_Y and then solving a series of fixed cardinality maximum-cost assignment problems, and comparing their cost to $(1+\lambda)\mathbb{E}[\mathbf{1}_{A=\varnothing}]_{s_X,s_Y}$.

Here, rather than computing exactly the values of $\mathbb{E}[\mathbf{1}_{A=\varnothing}]_{s_X,s_Y}$, $r_p = \mathbb{E}\left[\mathbf{1}_{A\neq\varnothing}\frac{\mathbf{1}_{p\in A}}{|A|}\right]_{s_X,s_Y}$ and $p_p = \mathbb{E}\left[\mathbf{1}_{p\in A}\right]_{s_X,s_Y}$, which is very costly, we will approximate them by sampling from the assignments distribution using the Metropolis-Hastings algorithm [Metropolis *et al.*, 1953, Hastings, 1970]. More precisely, we will approximate the values of $\mathbb{E}[\mathbf{1}_{A=\varnothing}]_{s_X,s_Y}/\mathbb{E}[1]_{s_X,s_Y}$, $r_p/\mathbb{E}[1]_{s_X,s_Y}$ and $p_p/\mathbb{E}[1]_{s_X,s_Y}$ which are proportional to the values we are looking for, by sampling assignments A from the distribution

$$\ell_0(A=a) = \frac{(N\vee M - |a|)!\left(\frac{p_R}{1-p_R}\right)^{|a|} \cdot \prod_{i\to j\in a}\frac{1}{2\pi\sigma^2}e^{-\|\mathbf{y}_j-\mathbf{x}_i\|/2\sigma^2}}{\sum_{a\in\mathbb{A}}(N\vee M - |a|)!\left(\frac{p_R}{1-p_R}\right)^{|a|} \cdot \prod_{i\to j\in a}\frac{1}{2\pi\sigma^2}e^{-\|\mathbf{y}_j-\mathbf{x}_i\|/2\sigma^2}}$$

where the sum is taken on all the assignments a between s_X and s_Y. We thus have

$$\frac{\mathbb{E}_\lambda(\tilde{a}_\lambda)}{\mathbb{E}[1]_{s_X,s_Y}} = \begin{cases} \ell_0(A=\varnothing) & \text{if } \tilde{a}_\lambda = \varnothing \\ \sum_{a\in\mathbb{A}}\ell_0(A=a)\sum_{p\in\tilde{a}_\lambda}\left(\mathbf{1}_{a\neq\varnothing}\frac{\mathbf{1}_{p\in a}}{|a|} + \frac{\lambda}{|\tilde{a}_\lambda|}\mathbf{1}_{p\in a}\right) & \text{otherwise} \end{cases}$$

The Metropolis-Hastings algorithm assumes given a probability function $Q(a\to\cdot)$ such that $Q(a\to a')$ is the probability of a transition from a to a', as well as an initial assignment a^0, and defines a Markov Chain $\{a^k\}_k$ by iterating the following steps:

1. draw $a' \sim Q(a^k \to \cdot)$
2. let $\rho = \dfrac{\ell_0(A=a')}{\ell_0(A=a^k)}\dfrac{Q(a'\to a^k)}{Q(a^k\to a')}$
3. accept the transition with probability $\min(r,1)$
4. if the transition is accepted, define $a^{k+1} = a'$, otherwise let $a^{k+1} = a^k$.

For integrable functions h, we then know that

$$\frac{\mathbb{E}[h(A)]_{s_X,s_Y}}{\mathbb{E}[1]_{s_X,s_Y}} = \lim_{N\to+\infty}\frac{1}{N}\sum_{1\leqslant k\leqslant N} h(a_k),$$

and in practice, we will "burn" a small number b of steps to avoid transitional effects and approximate the expectation after $(b+N)$ transition steps by the formula

$$\frac{\mathbb{E}[h(A)]_{s_X,s_Y}}{\mathbb{E}[1]_{s_X,s_Y}} \approx \frac{1}{N}\sum_{b+1\leqslant k\leqslant b+N} h(a_k)$$

for some large N. In the present case we use the family of functions

$$\begin{aligned} h_{\varnothing}(a) &= \mathbb{1}_{a=\varnothing} \\ h_{r_p}(a) &= \mathbb{1}_{a\neq\varnothing}\frac{\mathbb{1}_{p\in a}}{|a|} \qquad \text{for each pairing } p \text{ between } s_X \text{ and } s_Y \\ h_{p_p}(a) &= \mathbb{1}_{p\in a} \qquad \text{for each pairing } p \text{ between } s_X \text{ and } s_Y \end{aligned}$$

In Dellaert *et al.*, 2003, the authors propose several ways to define such a transition probability Q in a setting where assignments have a fixed length. We here follow their construct, extending it to the case of variable-length assignments. For a given assignment a between s_X and s_Y, define

$$\begin{aligned} w_{ij} &= \|\mathbf{y}_j - \mathbf{x}_i\|^2/2\sigma^2, \quad \text{and} \\ q_a(i,j) &= \begin{cases} e^{-w_{ij}}/\sum_{j'\neq a(i)} e^{-w_{ij'}} & \text{if } i \in \operatorname{dom}(a) \text{ and } j \neq a(i) \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

Definition 14 (Free points, paired points)**.** *If a is an assignment, we will denote $F_a(s_X)$ (resp. $P_a(s_X)$) the set of free (resp. paired) points of s_X in the assignment, that is, the set of points from s_X not appearing (resp. appearing) in a, and we define similarly $F_a(s_Y)$ and $P_a(s_Y)$.*

Hypothesis 2 (Number of points in the data)**.** *We assume that $\min(N, M) > 1$ for the following definitions to be correct, but in no way is this a limitation, as the problem is solved easily if N or M contains only one point.*

We now present two possible MCMC models, M_0 and M_1, to sample the assignments distribution.

Model M_0

Let a be the current assignment, we define the transition to a' in the following way:

- with probability $\mathbb{1}_{|a|>1}/2$, we permute two pairings: we choose two distinct indices i_0 and i_1 uniformly in $P_a(s_X)$, and we replace $\{(i_0 \to j_0), (i_1 \to j_1)\}$ with $\{(i_0 \to j_1), (i_1 \to j_0)\}$.

- with probability $(1 - \mathbb{1}_{|a|>1}/2) \cdot k/(N \wedge M)$ we remove a pairing: we choose i uniformly in $P_a(s_X)$, and we remove the pairing $(i \to a(i))$.

- with probability $(1 - \mathbb{1}_{|a|>1}/2) \cdot (1 - k/(N \wedge M))$ we add a pairing: we choose i uniformly in $F_a(s_X)$, we draw j from the distribution $j \sim e^{-w_{ij}}/\sum_{j'\in F_a(s_Y)} e^{-w_{ij'}}$, and we add the pairing $(i \to j)$.

We only need to compute the ratio r in each case. If a' is obtained from a by adding the pairing $(i \to j)$, we obtain

$$\begin{aligned}
\rho &= \frac{\ell_0(A = a')}{\ell_0(A = a)} \frac{Q(a' \to a)}{Q(a \to a')} \\
&= \frac{1}{N \vee M - |a|} \frac{p_R}{1 - p_R} \frac{1}{2\pi\sigma^2} e^{-w_{ij}} \, . \\
&\quad \frac{1 - \mathbb{1}_{|a|+1>1}/2}{1 - \mathbb{1}_{|a|>1}/2} \frac{\frac{|a|+1}{N \wedge M}}{1 - \frac{|a|}{N \wedge M}} \cdot \frac{1}{|a|+1} \cdot (N - |a|) \frac{\sum_{j' \in F_a(s_Y)} e^{-w_{ij'}}}{e^{-w_{ij}}} \\
&= \frac{p_R}{1 - p_R} \frac{1}{2\pi\sigma^2} \cdot (1 - \mathbb{1}_{|a|=1}/2) \frac{1}{M - |a|} \cdot \sum_{j' \in F_a(s_Y)} e^{-w_{ij'}}
\end{aligned}$$

Similarly, if a' is obtained from a by removing the pairing $(i \to j)$, by exchanging a' and a in the above equation and taking the inverse, we have

$$\begin{aligned}
\rho &= \frac{1 - p_R}{p_R} (2\pi\sigma^2) \cdot \frac{1}{1 - \mathbb{1}_{|a'|=1}/2} (M - |a'|) \cdot \frac{1}{\sum_{j' \in F_{a'}(s_Y)} e^{-w_{ij'}}} \\
&= \frac{1 - p_R}{p_R} (2\pi\sigma^2) \cdot \frac{1}{1 - \mathbb{1}_{|a|=2}/2} (M - |a| + 1) \cdot \frac{1}{e^{-w_{ij}} + \sum_{j' \in F_a(s_Y)} e^{-w_{ij'}}}
\end{aligned}$$

Finally, if a' is obtained from a by permuting $(i_0 \to j_0)$ and $(i_1 \to j_1)$, we have

$$\rho = \frac{e^{-w_{i_0 j_1}} e^{-w_{i_1 j_0}}}{e^{-w_{i_0 j_0}} e^{-w_{i_1 j_1}}}.$$

The irreducibility property of the transition kernel Q is obvious, since we can transition from any assignment a to any other assignment a' in a finite number of steps with a non-null probability by removing all the pairings from a, and then adding all the pairings from a'.

Model M_1

To have the Markov chain converge faster to the expected distribution, a possible optimization is to choose the edge permutations more cleverly, as noted in ivi. We regard the assignment a as a directed bipartite graph $\mathcal{G}_a$ where the nodes are the points in each image and the edges link the points i in the first image and j in the second image, directed from $i \to j$ if $(i \to j)$ is *not* in a, and directed from $j \to i$ otherwise. Let C be an oriented cycle in this graph. If a' is the assignment corresponding to the graph $\mathcal{G}_a$ where all the directions of the edges on the cycle have been reversed, we obtain a new assignment of the same cardinality, where the pairings of a have been permuted. We use this property to define a new model similar to the previous one except for the permutation of the edges, that is now defined as:

1. choose i_0 uniformly in $P_a(s_X)$
2. draw $j_0 \in P_a(s_Y) \setminus a(i_0)$ from the distribution $j_0 \sim e^{-w_{i_0 j_0}} / \sum_{j' \in P_a(s_Y) \setminus a(i_0)} e^{-w_{i_0 j'}}$,
3. choose $i_1 = a^{-1}(j_0)$,
4. iterate to construct the path $\pi = i_0 \to j_0 \to i_1 \to ...$ until a cycle is formed

we thus obtain the path $\pi = T + C$, where $+$ denotes the concatenation of paths, T is called the transient path and C is the cycle. We will denote $C_X = s_X \cap C$.

If a' is obtained from a by permuting the pairings along the cycle C, we have

$$\begin{aligned}
\rho &= \frac{\ell_0(A = a')}{\ell_0(A = a)} \frac{Q(a' \to a)}{Q(a \to a')} \\
&= \prod_{i \in C_X} \frac{e^{-w_{ia'(i)}}}{e^{-w_{ia(i)}}} \cdot \prod_{i \in C_X} \frac{e^{-w_{ia(i)}} / \sum_{j' \in P_{a'}(s_Y) \setminus a'(i)} e^{-w_{ij'}}}{e^{-w_{ia'(i)}} / \sum_{j' \in P_a(s_Y) \setminus a(i)} e^{-w_{ij'}}} \cdot \underbrace{\frac{\sum_{t \in T_a(C)} \mathbb{P}(t)}{\sum_{t \in T_{a'}(C)} \mathbb{P}(t)}}_{=1} \\
&= \prod_{i \in C_X} \frac{\sum_{j' \in P_a(s_Y) \setminus a(i)} e^{-w_{ij'}}}{\sum_{j' \in P_{a'}(s_Y) \setminus a'(i)} e^{-w_{ij'}}}
\end{aligned}$$

where $T_a(C)$ is the set of transient paths in $\mathcal{G}(a)$ that end on a point of the cycle C (and contain only one point in C). Since the pairings not appearing in C are the same in a and a', the transient paths leading to C are the same and have the same probability, the ratio of the sums over T_a and $T_{a'}$ is equal to one.

Comparison of MCMC models

Table 9 displays the acceptance rates (proportion of accepted transitions) for both transition models. The transition model M_1 exhibits a consistently higher acceptance rate than the transition model M_0, yet this does not necessarily indicates that the MCMC chains will converge faster. In both cases, the acceptance rates seem reasonable to us.

$p_R =$	0.25	0.5	0.75
$a_r(M_0) =$	0.25	0.29	0.16
$a_r(M_1) =$	0.28	0.39	0.31

Table 9: Acceptance rates a_r for both models M_0 and M_1 for different values of the parameter p_R used in the generation of the problems and as a parameter for the MCMC algorithms, with $N = M = 10$ and $\sigma^2 = 0.01$. The values have been obtained as averages of 50 runs, where each run consists in 50 000 "burned" transitions and the acceptance rate is computed on the next 100 000 transitions. The acceptance rate is consistently higher for the model M_1, but this does not imply that the model is better suited. Both models have reasonable acceptance rates.

We then study the convergence speed of both models. For a small number of points in each image it is feasible to compute the exact WRAP coefficients $\{p_p\}_p$ (for instance, see equation 5.4), and it is hence possible to show the decrease of the average error for each model (see Figure 72). For large number of points, we instead display the decrease in variance of the coefficient estimation (see Figure 73). In both cases, the MCMC models behave essentially the same, and marking that the M_0 model has a slightly lower computational overhead, we recommend using the latter.

It is interesting to note that in Dellaert *et al.*, 2003, the authors found that a model similar to M_1 in the case where the number of pairings was fixed greatly improved the performances, while it has no significant effect when the number of pairings is allowed to change.

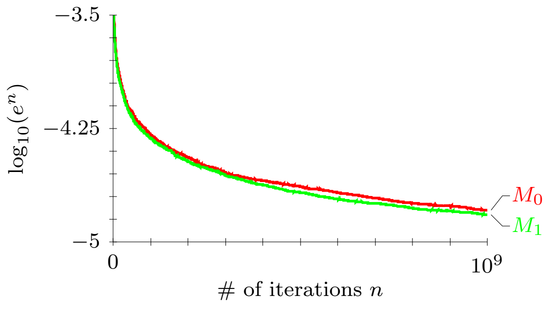


Figure 72: (**Small number of points, approximation error**) We generated 16 problems with $N = M = 7$, $p_R = 0.5$ and $\sigma^2 = 0.01$, and we ran each MCMC model 10 times on each problem for 10^9 steps (and 500 000 burned initial steps). For each step, we compute the criterion $e^n = \sum_p |p_p^n - p_p|/NM$ where the $\{p_p\}_p$ are the exact coefficients corresponding to equation 5.4, and the $\{p_p^n\}_p$ are the approximation obtained at iteration n. The values displayed are the averages over all problems and all chains for each problem of $\log_{10} e^n$. There is only a slight increase in convergence speed when using M_1, but this might often be offset by the fact that M_0 is computationally faster, and we would hence rather recommend using M_0.

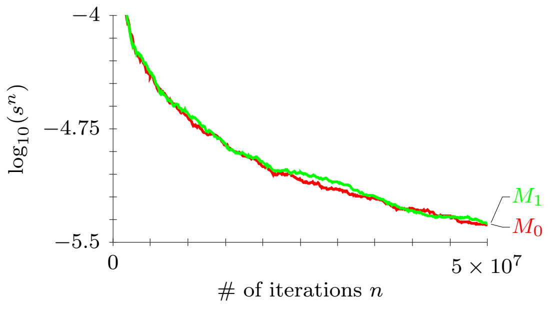


Figure 73: (**Large number of points, decrease in variance**) When the number of points is high, it is no longer feasible to compute the exact solution of WRAP, and to study the convergence of the MCMC chains, we adopt the following protocol. We generate 20 problems with the normal generative model with parameters $N = M = 20$, $p_R = 0.5$ and $\sigma^2 = 0.005$. We run 10 chains on each problem with each MCMC model for 50 000 000 steps (burning the first 1 000 000 steps) and we record at each step n the value of $c^n = \sum_p p_p^n$ where the $\{p^n\}_p$ are the approximations of the coefficients corresponding to equation 5.4 obtained at step n. For each MCMC model, we display at each step n the logarithm of the value s^n which is the mean on all the problems of the variance on all the chains of c^n. The variance decreases at the same rate for both models, and therefore again, marking that the computational complexity of model M_0 is slightly advantageous, we recommend using the latter.

5.3 EXPERIMENTS

5.3.1 Performances of classical algorithms

We noted earlier by comparing the results of a very simple greedy local algorithm and a global one, respectively the Nearest-Neighbor and the Maximum-Likelihood, that the problem of the point correspondence without features seems to be either very simple or very complex: on simple cases, even the simplest algorithm performs almost optimally, while on complex cases, global algorithms also have very low performances.

Now that we can compute the theoretical limit of point correspondence algorithms using the WRAP criterion, we can answer the question of whether those algorithms are indeed close to the optimum: we generate very simple, moderate and difficult assignment problems using our generative model, where the difficulty is defined in terms of the mean displacement of a point between the two images, and we plot the WRAP curve as λ varies, as well as the curves obtained using NN and ML as their own parameter varies (figure 74).

Note that, although we compared WRAP with MAP in the previous section because they have a similar set of parameters (p_R and σ^2, and WRAP also has the parameter λ), we will use ML here because we are interested in the locations that the algorithms can attain in the recall and precision space, and it will be easier to visualize them for ML because it has only one parameter, and the attainable locations are thus a curve.

We observe that although NN and ML have comparable performances, and are able to attain the point of maximum recall – and in this case attain it with the maximal possible precision – they lie well below the optimal curve, and more importantly, they are not able to make a compromise between the recall and the precision, as their curves are essentially flat. In other words, it is not possible to tune the parameters to increase the precision at the expense of the recall – although this might be a desirable feature in some applications, and we see that it is theoretically possible, as attested by the optimal WRAP curve.

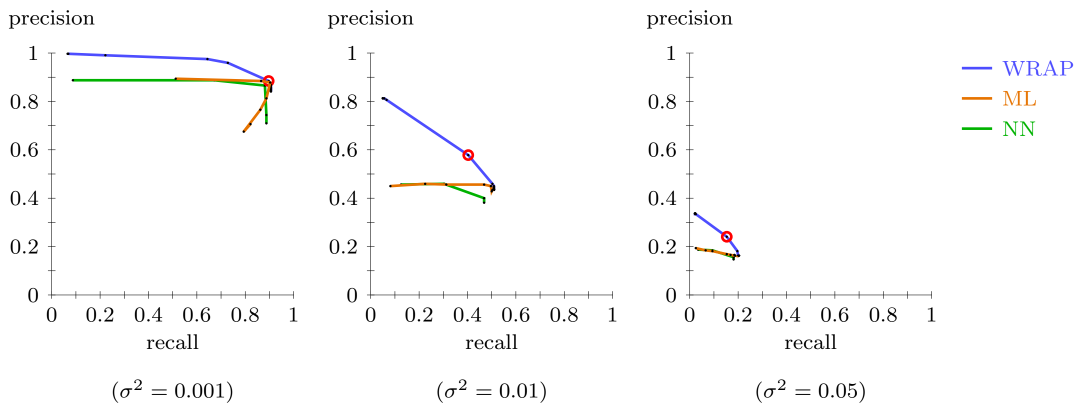


Figure 74: (**Performance of classical algorithms**) Mean precision and recall of the ML (brown) and NN (green) algorithms as their parameters σ^2_{ML} and c_{thre} vary, compared to the WRAP (blue) optimum on data generated by the generative model with parameters $N = M = 20$, $p_R = 0.8$ and various values of σ^2. The values are averages of 500 runs, and the circled point corresponds to the WRAP performances for $\lambda = 1$. No matter the complexity of the problem as measured by σ^2, ML and NN have similar performances, and are able to attain the maximal possible recall – yet they lie well below the optimal curve and cannot make a compromise to improve the precision at the expense of the the recall, since their curves are essentially flat.

In this sense, the main weakness of most approaches to the point correspondence problem is that they make decisions based on whether a pairing $A \to B$ is very likely or belong to a very likely assignment, but they do not take into account the existence of ambiguities – it might for instance be the case that the pairing $A \to C$ is also very likely, in which case WRAP would avoid such a risky detection, because it would lower the criterion $r + \lambda p$.

Figure 75 shows an example of ambiguity that is detected by WRAP: when b occupies the position marked by b', WRAP does not know how to decide between the fact that b is a spurious detection, or that it is indeed one of the point A or B that has moved away exceptionally far. Thus, WRAP decides to leave the point out of the assignment, but still chooses to make a detection – and since the probability that a comes from A is only slightly higher than the probability that it comes from B, but the probability that b comes from B is much lower than that it comes from A, WRAP assumes that it is less risky to pair B to a.

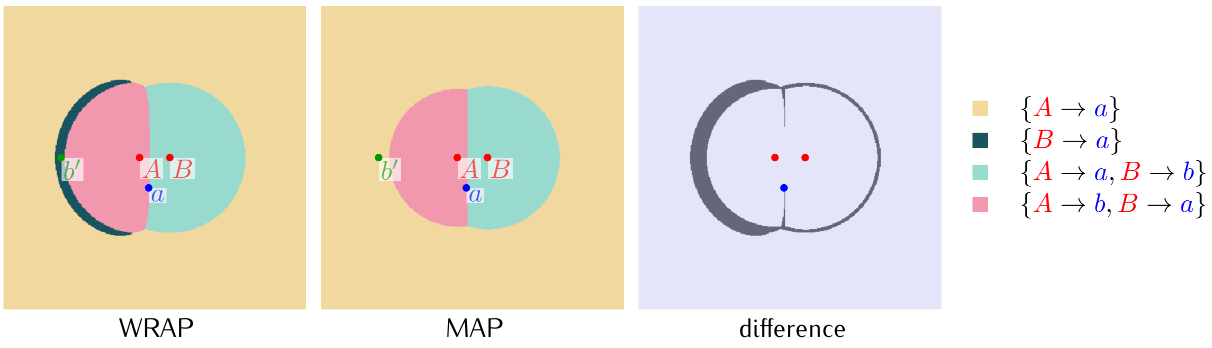


Figure 75: (**Assignment ambiguity**) WRAP is able to detect ambiguities in an assignment and thus make recall and precision compromises: for instance, when b occupies the position marked by b', one would expect WRAP to pair $A \to a$, but it pairs $B \to a$ instead, even though this pairing is less likely in the generative model, because WRAP does not know whether b is a spurious point or whether it is actually the point A that was displaced exceptionally far, and thus decides that it is less risky to assume that a comes from B. Such a strategy results in a greater expectation for the weighted recall and precision measure.

This is even more obvious when the input data is drawn in such a way as to force ambiguities: for instance, we draw some of the points uniformly in a small disc in the center of the image, and the rest of the points uniformly outside this disc – and use the standard generative model to displace the points and generate the second image (see Figure 76). Figure 77 shows the results of NN and ML on these data, and they are still farther from the optimum. We also mark that the point corresponding to $\lambda = 1$ on the WRAP curve (the point circled in red) has decreased in recall if we compare it to the case with less ambiguities. The WRAP approach makes fewer detections when there are ambiguities, resulting in a much higher precision.

5.3.2 Modified nearest neighbor algorithm

This prompts us to see whether a very simple modification of the Nearest-Neighbor algorithm taking into account assignment ambiguities would fare as well as the WRAP optimal algorithm.

We propose the NN_r algorithm that uses a measure of the credibility, for a pairing $\mathbf{x}_i \to \mathbf{y}_j$, that either $\mathbf{x}_i$ or $\mathbf{y}_j$ has another credible alternative pairing:

$$t(i,j) = \min\left(\frac{c_{ij}}{\min_{j' \neq j} c_{ij'}}, \frac{c_{ij}}{\min_{i' \neq i} c_{i'j}}\right) > t_r.$$

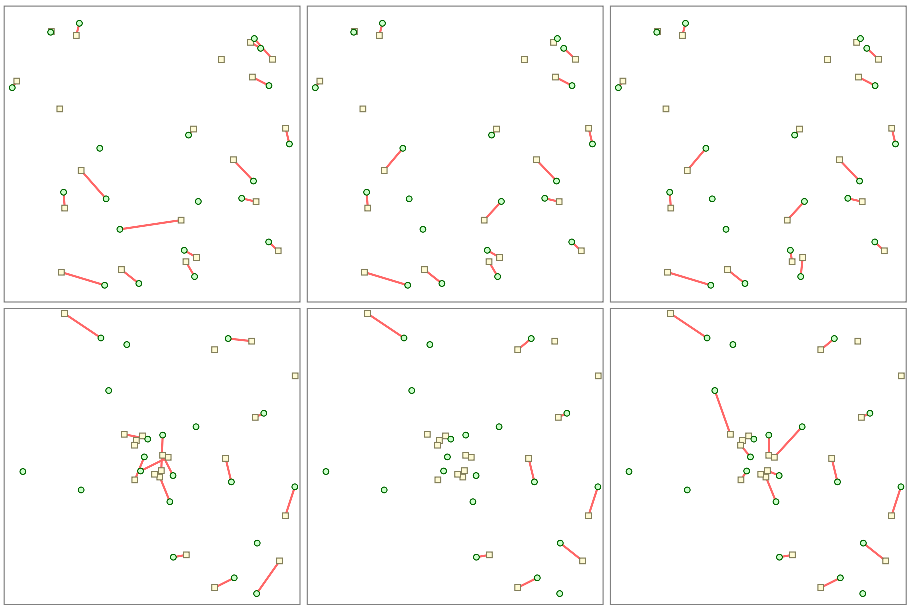


Figure 76: We display the results of the WRAP and ML algorithms when the generated assignment has been obtained using the generative model with parameters $N = M = 20$, $p_R = 0.8$ and $\sigma^2 = 0.05$ (first row), and when it has been obtained using the same generative model except that 10 of the points detected in the first image have been drawn uniformly in a disc of radius 0.2 and the 10 others have been drawn uniformly outside of this disc (second row). The left figure is the generated (correct) assignment, the center figure is the result of the WRAP algorithm with parameters p_R, σ^2 and $\lambda = 1$, and the right figure is the result of the maximum-likelihood algorithm with parameter σ^2. The WRAP algorithm behaves in a manner similar to ML when the assignment is easy to detect (the points are far apart as in the first row), but does not make pairing decisions when there are ambiguities (as in the second row), hence making fewer false detections.

(note that changing the min of the two values into a max would not impact our results very much).

The NN_r algorithm has two parameters, the usual threshold c_{thre} on the cost of allowed pairings, as well as a risk threshold t_r on the measure $t(i, j)$.

More precisely, we use the standard Nearest Neighbor algorithm that sorts the pairings $\mathbf{x}_i \to \mathbf{y}_j$ by increasing cost c_{ij}, considers them in order, and adds each pairing $\mathbf{x}_i \to \mathbf{y}_j$ in turn to the assignment if neither $\mathbf{x}_i$ nor $\mathbf{y}_j$ appear in a previously added pairing, if $\|\mathbf{x}_i \to \mathbf{y}_j\| \leqslant c_{\text{thre}}$ and if $t(i, j) > t_r$.

This algorithm indeed makes it possible to trade precision for recall, yet it is still far from the optimal WRAP curve (see Figure 78). It is however a very simple algorithm, and hopefully some more complex criterion would lead to performances on par with WRAP, with a much lower computational cost.

Figure 79 shows a situation where the NN_r algorithm recognizes a possible assignment ambiguity and decides to not make a detection in order to improve the expected precision.

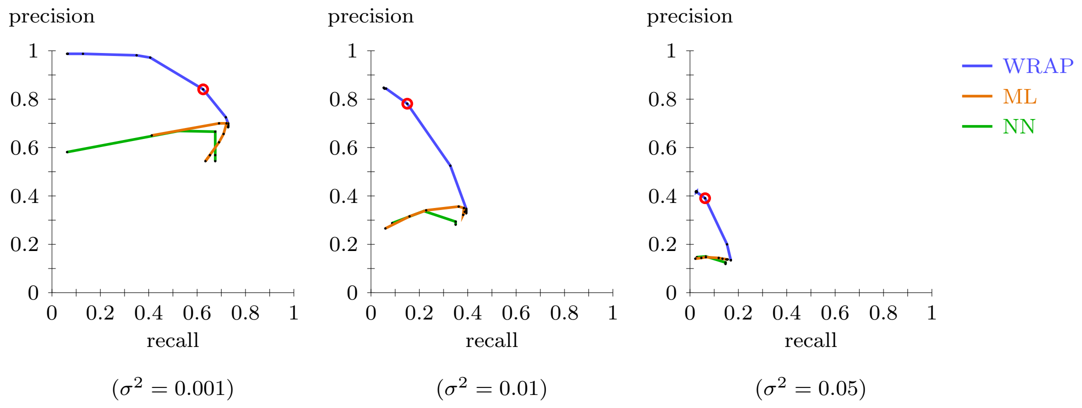


Figure 77: (**Performance of classical algorithms in presence of ambiguities**) Mean precision and recall of the ML (brown) and NN (green) algorithms as their parameters σ^2_{ML} and c_{thre} vary, compared to the WRAP (blue) optimum on data generated by the generative model with parameters $N = M = 20$, $p_R = 0.8$ and various values of σ^2, where 10 of the points detected in the first image have been drawn uniformly in a disc of radius 0.2, and the 10 others have been drawn uniformly outside this disc. The values are averages of 500 runs, and the circled point corresponds to the WRAP performances for $\lambda = 1$. Again, ML and NN are not able to make a compromise between the recall and the precision, and they lie far below the optimum WRAP curve, hinting at the fact that it is theoretically possible to greatly improve the precision by lowering the recall.

5.3.3 WRAP as an algorithm

The optimality of WRAP and the fact that it is significantly superior to the classical algorithms makes it very interesting as an algorithm by itself, rather than simply an optimality criterion.

In this case however, the values of p_R, σ^2 and λ all become parameters, as there is no reason to know in advance what they should be. We thus need to study the feasibility of selecting the WRAP parameters in such a way that the resulting algorithm gives results superior to the classical approaches.

The parameter λ controls the weight of the precision with respect to the recall. Figure 80 displays the performances of the WRAP algorithm on a sample problem for different values of λ. The value $\lambda = 1$ seems to give reasonable results, and we will use this value from now on as a reference.

In figures 81 and 82 we compare the performances of WRAP ($\lambda = 1$) and ML as algorithms when feeding them the exact parameters p_R and σ^2 that have been used to generate the data, respectively when the input data comes from the standard generative model, and the one modified by drawing points uniformly in a disc to increase the ambiguities.

We note that in this case, using the WRAP algorithm does not improve the results much, but there is a systematic bias for a greater recall for ML and a greater precision for WRAP, which is clearly perceptible in Figure 82

The WRAP criterion is thus probably more a theoretical tool to study algorithms than it is a practical algorithm to solve the point correspondence problem, although it might be useful when tackling data containing many ambiguities, when it is important that only few false detections be made.

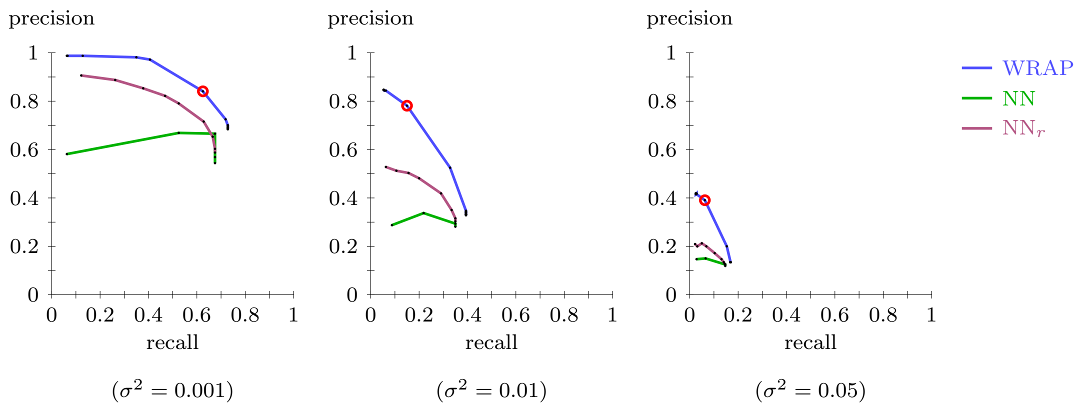


Figure 78: **(Performance of the modified Nearest-Neighbor algorithm in presence of ambiguities)** Mean precision and recall of the NN (green) as its parameter c_{thre} varies, and NN$_r$ (purple) algorithms as its parameter t_r vary and its parameter $c_{\text{thre}} = 0.4$, compared to the WRAP (blue) optimum on data generated by the generative model with parameters $N = M = 20$, $p_R = 0.8$ and various values of σ^2, where 10 of the points in the first image have been drawn uniformly in a disc of radius 0.2, and the 10 others have been drawn uniformly outside this disc. The values are averages of 500 runs, and the circled point corresponds to the WRAP performances for $\lambda = 1$. Using the NN$_r$ algorithm enables to make a compromise between the recall and the precision in a limited manner, but still does not come very close to the optimal WRAP performances.

5.4 CONCLUSION

When the generative model is known, the WRAP optimality criterion enables us to show that ML and MAP are already almost optimal in that when we choose their parameter to reach the point of maximal recall, the point reached is also that of maximal precision (see figures 74 and 77).

WRAP as an optimality curve enables us to assess the performances of the algorithms and recognize that one of the weakness of the classical algorithms is that they are ot able to trade precision for recall (they essentially have flat curves in the recall/precision space).

In turn, this has prompted us to study the behavior of the WRAP criterion in a variety of situations, and its ability to handle ambiguities in order to maximize the weighted recall and precision criterion.

We have then derived a very simple algorithm based on the Nearest Neighbor that is able to achieve a good recall/precision compromise using a simple ambiguity detection measure.

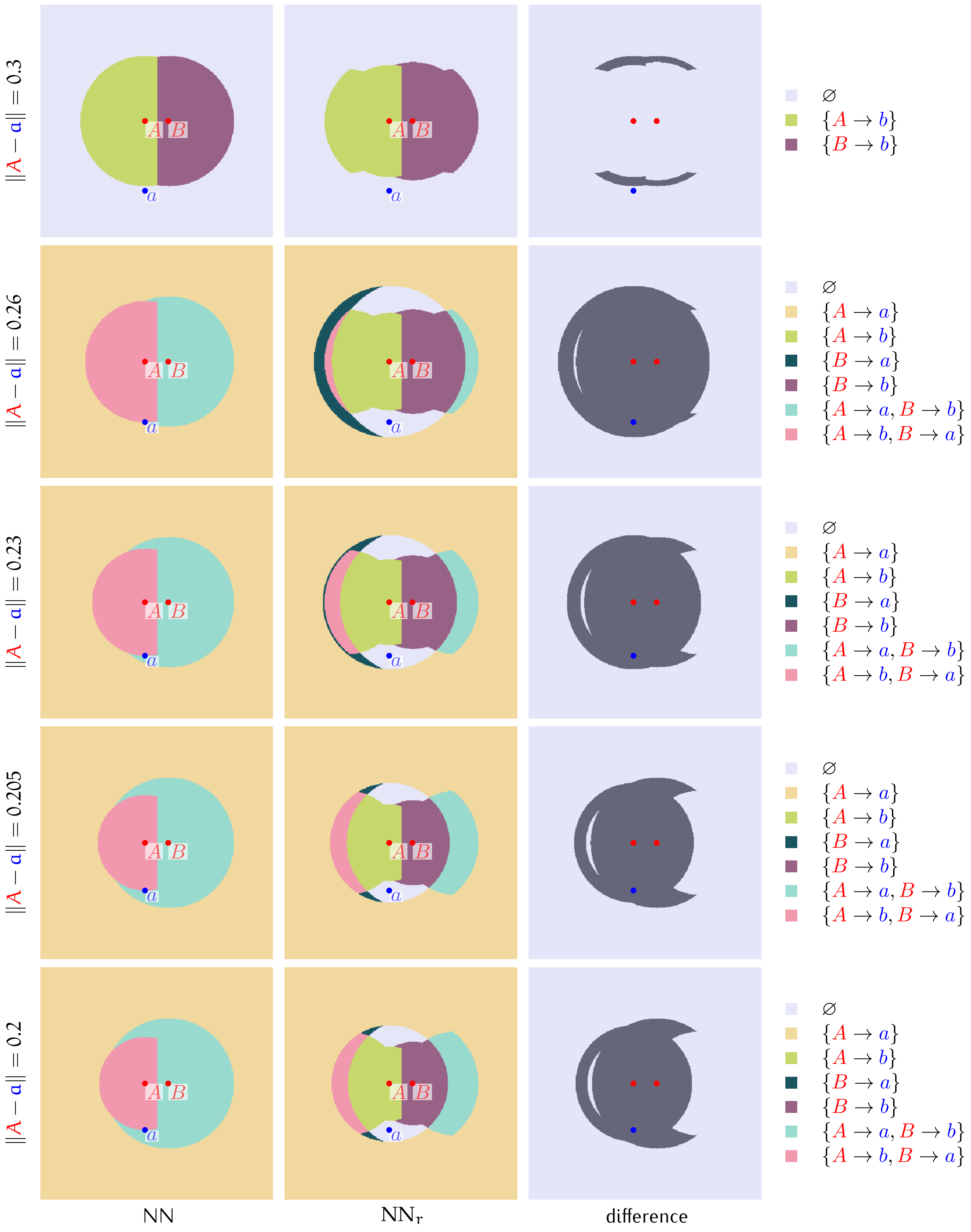


Figure 79: (**Assignment ambiguities**) Assignment maps for NN where $c_{\text{thre}} = 0.28$ and NN_r where $c_{\text{thre}} = 0.28$ and $t_r = 0.8$, as a moves closer to A. We see that the simple modification of NN to NN_r make its behavior much more complex, and might enable a compromise between precision and recall. For instance, in some situation where there is an assignment ambiguity, NN_r might choose the empty assignment, while NN will return an assignment.

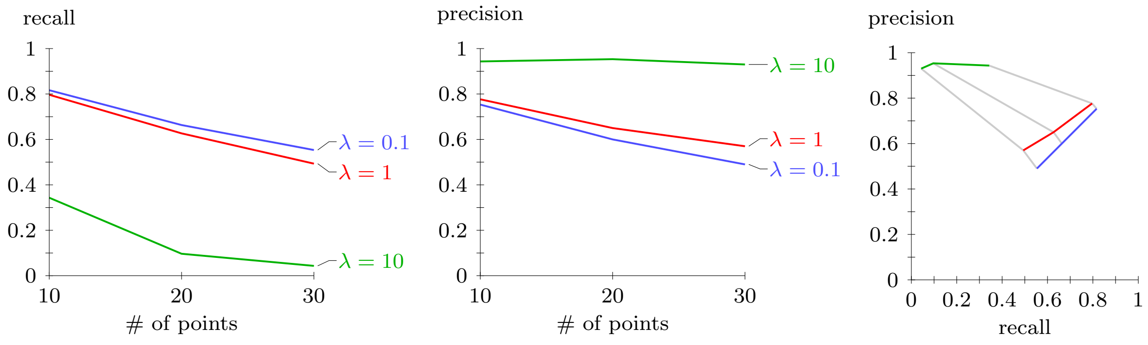


Figure 80: Recall and precision of the WRAP algorithms on data generated using the normal generative model with parameters $p_R = 0.8$ and $\sigma^2 = 0.005$, and a number of points varying from 10 to 30. The points are averages of 500 repetitions. The WRAP algorithm has exact the parameters p_R and σ^2 used to generate the data, and a variety of values for parameter λ. The results have approximately converged for $\lambda = 0.1$ and decreasing the value would not change the curve. The results have converged in precision for $\lambda = 10$, and increasing λ would only decrease the recall. From these observations, we choose $\lambda = 1$ as a reasonable value for the WRAP parameter.

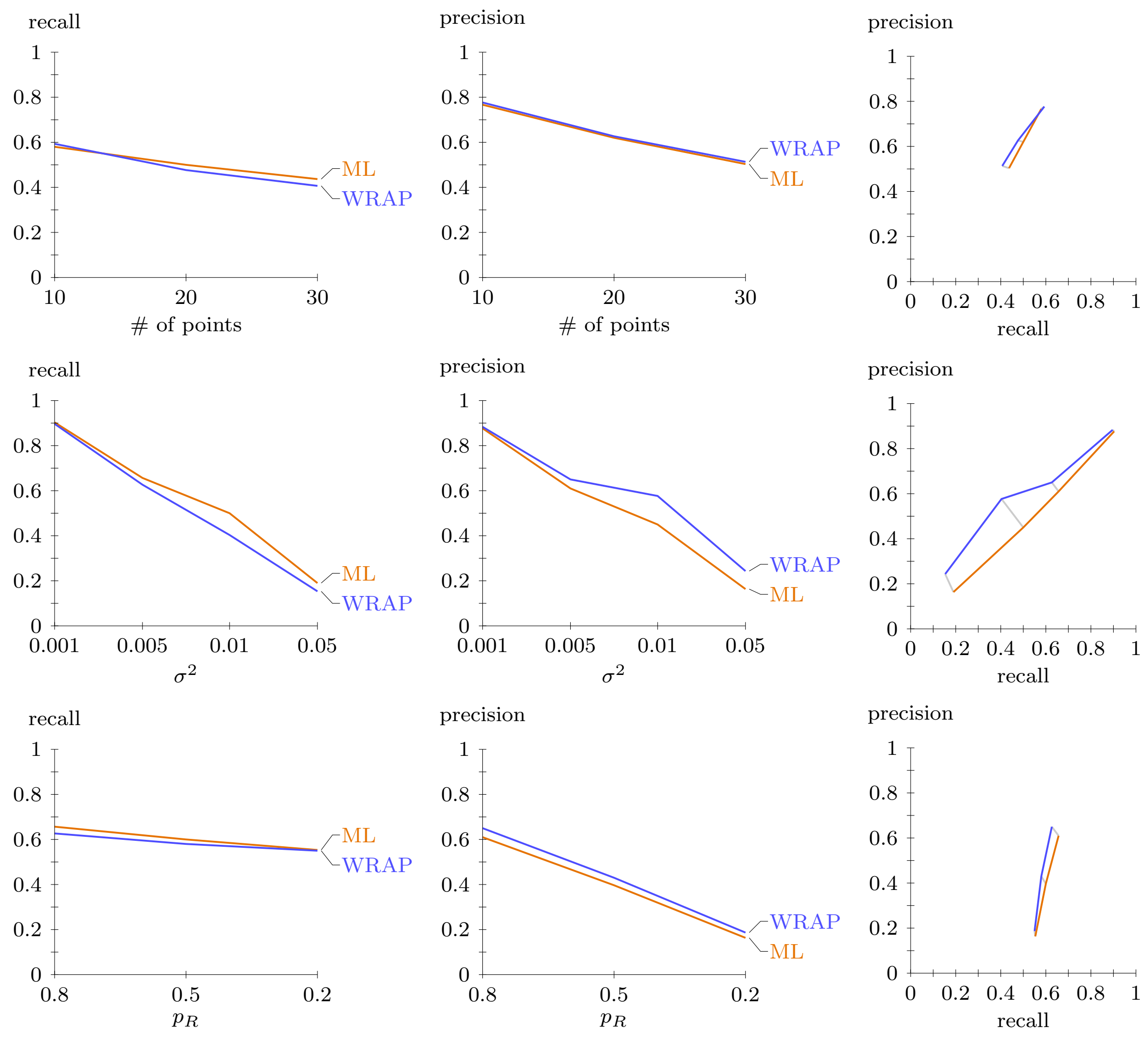


Figure 81: (**WRAP as an algorithm**) Recall and precision of the ML and WRAP algorithms on data generated using the normal generative model with default parameters $N = M = 20$, $p_R = 0.8$ and $\sigma^2 = 0.005$, but those are varied in each experiment. The first experiment varies the number of points N from 10 to 30 to increase the density of the points, the second experiment varies the value of σ^2 from 0.001 to 0.05 to increase the amplitude of the points displacements and the third experiment varies the value of p_R from 0.8 to 0.2 to increase the ratio of noise points. The ML and the WRAP algorithms have exact parameters (the exact σ^2 used to generate the data for ML and WRAP, the exact p_R for WRAP), and WRAP is used with parameter $\lambda = 1$. On this data set, WRAP as an algorithm with the parameter $\lambda = 1$ behaves essentially similarly to ML. WRAP thus seems to be more useful as a theoretical optimality criterion to study algorithms behaviors than as a practical algorithm.

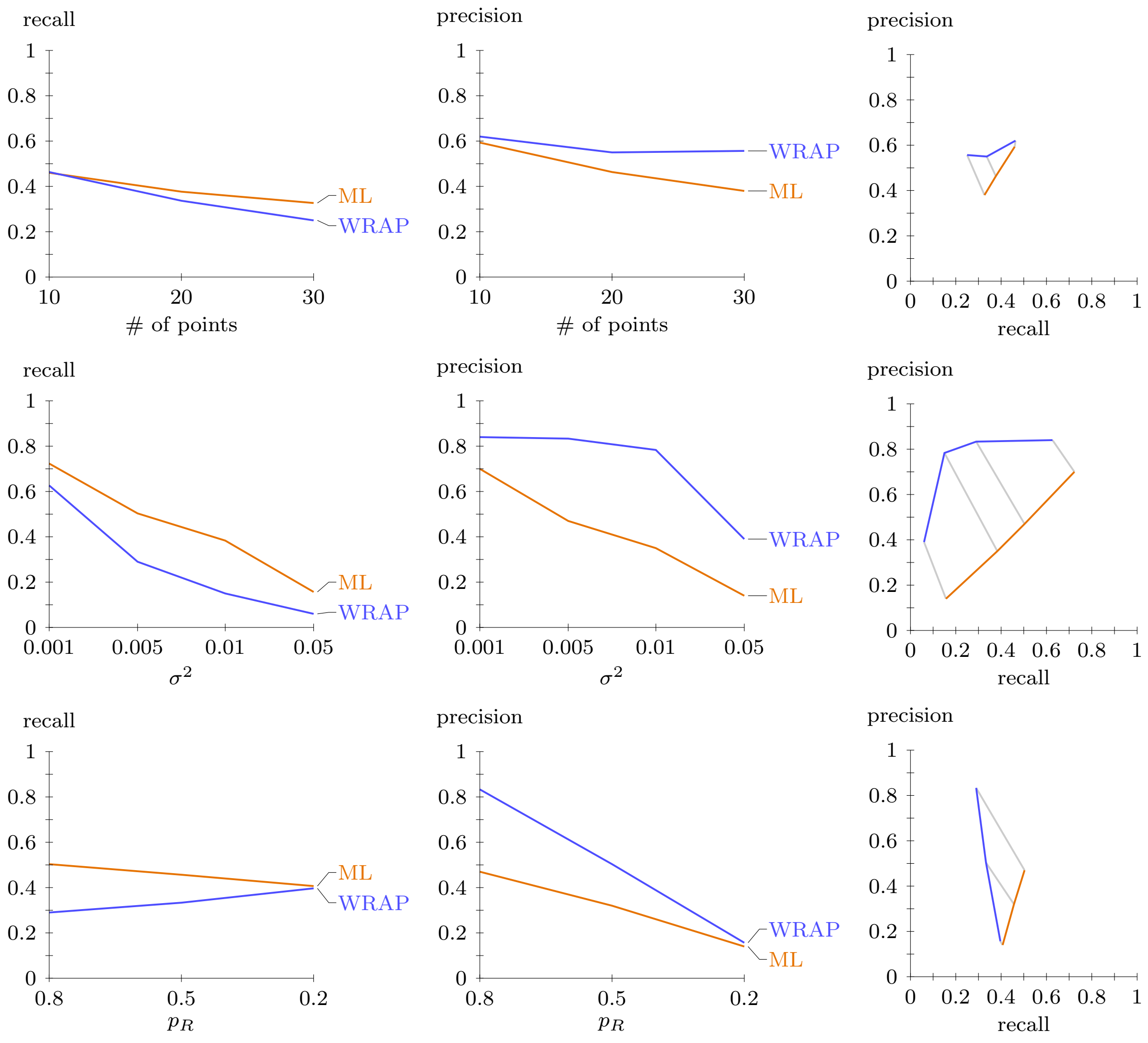
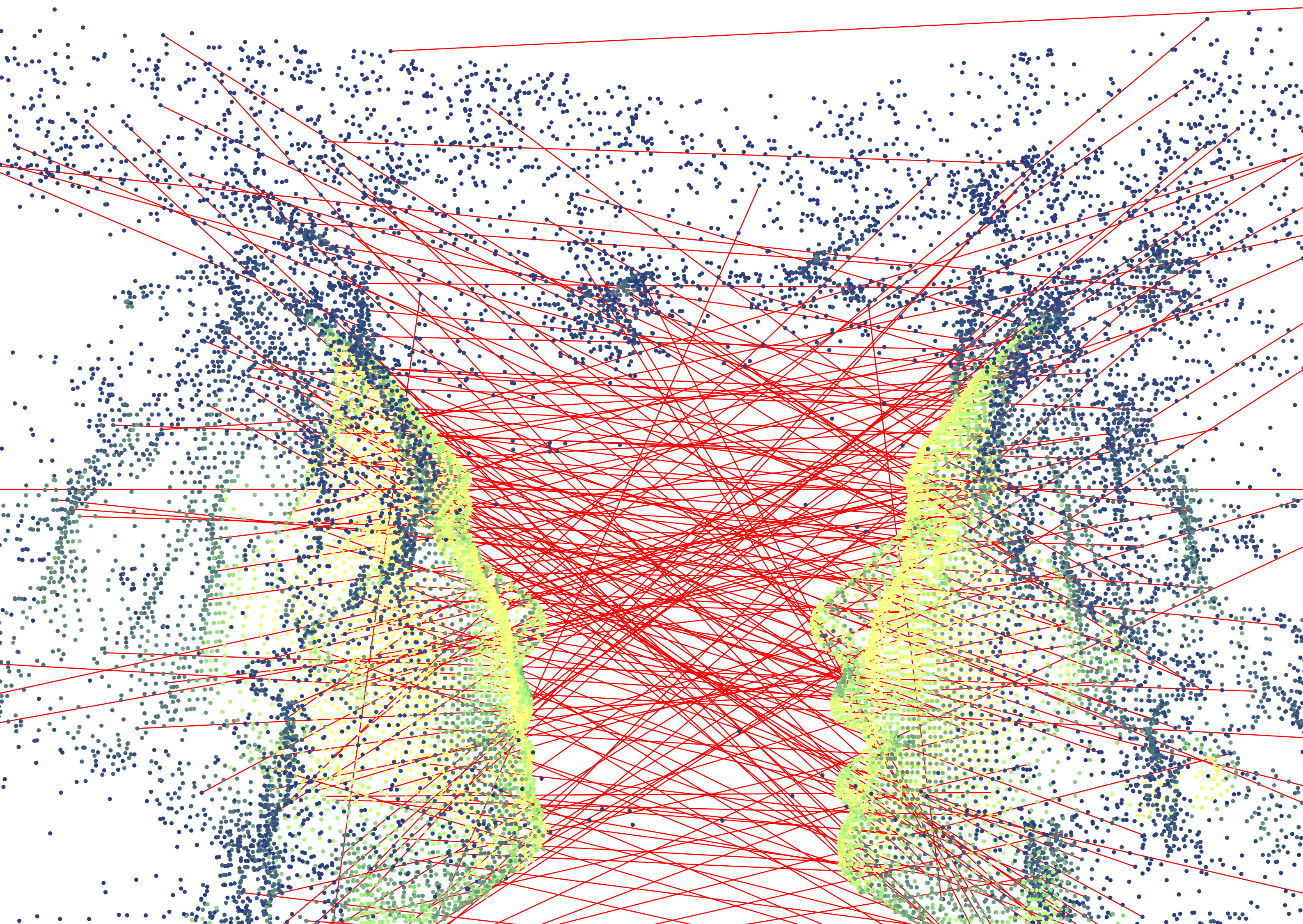


Figure 82: Recall and precision of the ML and WRAP algorithms on data generated using the normal generative model with default parameters $N = M = 20$, $p_R = 0.8$ and $\sigma^2 = 0.005$, but those are varied in each experiment. In the first image, half of the points are drawn uniformly inside a disc of radius 0.2, and the other half is drawn uniformly outside this disc. The algorithm parameters and their variations are the same as those in Figure 81 In the presence of ambiguities, WRAP as an algorithm, with parameter $\lambda = 1$ seems to make a compromise in order to improve the precision, while lowering the recall.

Composition using a random assignment
between 3D point clouds of Thom Yorke's face
extracted from the open-data video of *The House of Cards*, Radiohead



6 Parameterless algorithms for the point correspondence problem

CONTENTS

The point correspondence algorithms presented in the previous chapter that handle spurious points and occlusions require some arguably difficult to set parameters: a threshold c_{thre} on the costs of the pairings for the Nearest Neighbor, an expected assignment cardinality k for the k-cardinality assignment problem, or the parameters of the assumed generative model for the data, like p_R and σ^2 for the MAP or WRAP algorithms. Trying a wide range of candidate parameters to extract an assignment, and manually picking the best solution is often a time-consuming process, and many users would rather use a push-button solution even if the results were slightly sub-optimal.

For the trajectory tracking problem, the NFA framework presented in Chapter 3 allowed us to define an algorithm, ASTRE, with a simple to set parameter returning high quality trajectories with few false detections ; the NFA criterion could also be used on its own to filter the results of any other detection algorithm, and might help us circumvent the parameter selection problem by automatically discarding poor-quality solutions.

Likewise, in the first section, we derive three a-contrario criteria to quantify the quality of an assignment and their associated extraction algorithms. We explore the performance of the algorithms and the use of the criteria as filters for the parameter selection problem in the second section. In the third section, we describe an application of the point correspondence problem to trajectory tracking, and use the resulting algorithm to track aggregates in bacteria. We also discuss how the NFA criteria can be adapted when the problem data is quantized.

6.1 MEANINGFUL PAIRINGS AND MEANINGFUL ASSIGNMENTS

6.1.1 Meaningful pairings

We remind the reader of the main difference between the a-contrario approach and the classical detection algorithms such as those presented in the previous chapter: the latter use a probabilistic model describing how the data has been generated, while the former defines a naive model describing the background noise – that is, the structures we *do not* want to detect. An assignment is then deemed meaningful and is detected when it "looks too good" to have likely been generated by this naive model. The power of this approach is that it enables us to define the structure we wish to detect "en creux" by defining a very simple model of what *is not* a structure rather than having to define an often complex generative model sporting many parameters.

More precisely, recall from Section 3.2.1 that the definition of an NFA criterion requires two ingredients: the naive model representing the background noise, and a fitness function indicating what property of the structures of interest we should measure to distinguish them from noise.

As in the previous chapter, we assume that the domain of the images is the square $\Omega = [0,1] \times [0,1]$ (although our method extends smoothly to images of any shape and size). The natural naive model is that the detections in the images are uniformly drawn in Ω, that is, we detect $S_X = \{X_1, ..., X_N\}$ and $S_Y = \{Y_1, ..., Y_M\}$, respectively the points in the first and second images.

The first idea to detect assignments in the images is to mimic trajectory detection and sequentially extract individual correspondences between pairs of points in each image that are close enough from each other. Thus, the structures of interest are the $N \times M$ pairings (i,j) – or, equivalently, (X_i, Y_j) – between the points in each image.

The last ingredient of the a-contrario model is the measurement function. In the trajectory detection model, we assumed that trajectories were smooth, in the present case we will assume that the points have been only slightly displaced between both images, and hence a natural measure of the fitness of a pairing between two points is their distance

$$d(\mathbf{x}, \mathbf{y}) = \|\mathbf{y} - \mathbf{x}\|.$$

If (i,j) is a pairing, Proposition 1 tells us that we can build a NFA by an appropriate weighting of the probability $\mathbb{P}(d(X_i, Y_j) \leqslant \delta)$; in the present case, it obviously makes no sense to choose different weights for different pairings as the problem is completely symmetrical – and we therefore weigh the pairings uniformly. We now compute the probability bound and define the NFA criterion.

Proposition 10 (Per pairing probability bound)**.** *Let (i,j) be a pairing, then the following bound holds:*

$$\mathbb{P}(d(X_i, Y_j) \leqslant \delta) \leqslant \pi\delta^2 \tag{6.1}$$

Proof — The area of $\bar{B}(X_i, \delta) \cap \Omega$ is bounded from above by $\pi\delta^2$. □

Proposition 11 (Per pairing NFA). *The function* NFA^p *defined by*

$$\mathrm{NFA}^p(\delta) = NM \cdot \pi\delta^2 \tag{6.2}$$

is a Number of False Alarms for the measurement d.

Definition 15 (Meaningful pairing). *We say that the realization* $(\mathbf{x}_i, \mathbf{y}_j)$ *of a pairing* (X_i, Y_j) *is* ε-meaningful *if* $\mathrm{NFA}^p(d(\mathbf{x}_i, \mathbf{y}_j)) \leqslant \varepsilon$*, and that it is* meaningful *if it is* 1*-meaningful.*

The extraction of a complete assignment is then done greedily – as in the ASTRE trajectory extraction case – by choosing a NFA threshold ε, and iteratively detecting the most meaningful pairing in the data (among pairings whose end points do not belong to a previously extracted pairing) until all ε-meaningful pairings have been extracted.

The above algorithm is very simple and akin to a nearest-neighbor extraction – it only requires to sort the pairings, and runs in $\mathcal{O}(NM\log(NM))$.

6.1.2 Meaningful assignments

The greedy pairings extraction suffers from the same weakness as the nearest-neighbor algorithm or the ASTRE trajectory extraction, namely, that it considers pairings individually in the order of their NFA rather than searching for a globally optimal assignment. Most of the correspondence algorithms (and WRAP is no exception) try to extract such a global optimum by considering many pairings at once. Indeed, one may presume that knowing not only that the distances between two points is small, but also that this is *simultaneously* the case for many points, provides us with more information and enables us to detect more pairings, or pairings between points that are further apart, hence increasing the recall.

Additionally, it may be the case that assignments minimizing a global criterion are intrinsically better than greedily extracted ones, because they resolve some ambiguities, and will hence improve both the recall and the precision. This is what we propose to explore now, by defining a NFA criterion on global assignments rather than individual pairings.

We keep the same naive model and notations as in the previous section, but the structures that we now wish to detect are the assignments between the points of S_X and S_Y (we recall the definition 7 of an assignment):

Definition 16 (Assignment). *An* assignment a *of size* $|a| = k$ *between* S_X *and* S_Y *is a set of* k *pairs* $a = \{i_1 \to j_1, ..., i_k \to j_k\}$*, where for any index* p, $1 \leqslant i_p \leqslant N$ *and* $1 \leqslant j_p \leqslant M$*, and for any index* q *such that* $p \neq q$*, we have* $i_p \neq i_q$ *and* $j_p \neq j_q$*. We denote by* $\mathbb{A}$ *the set of all assignments, and by* $\mathbb{A}_k$ *the set of assignments of size* k.

There is a natural equivalence between an assignment $a \in \mathbb{A}$ and the set of pairs $X_a = \{X_{i_1} \to Y_{j_1}, ..., X_{i_k} \to Y_{j_k}\}$ – and we shall thus abusively call such sets assignments. A realization a of the random variable X_a will be called a *realization of the assignment* a. The realizations of X_a in the naive model are hence sets of randomly drawn points and they will generally *not* look like what we would want to call an assignment.

The measurement function must now be defined globally on an assignment. There are at least two natural measures for the fitness of an assignment: the sum of the distances, and the maximal distance between its paired points.

Definition 17 (Sum displacement and maximal displacement)**.** *Let* $\mathbf{z} = \{\mathbf{x}_1 \to \mathbf{y}_1, ..., \mathbf{x}_k \to \mathbf{y}_k\}$ *be a set of* k *pairs of points in the image domain, we define the* sum displacement *as*

$$c_{\text{sum}}(\mathbf{z}) = \sum_i \|\mathbf{x}_i - \mathbf{y}_i\|,$$

and the maximal displacement *as*

$$c_{\max}(\mathbf{z}) = \max_i \|\mathbf{x}_i - \mathbf{y}_i\|$$

If a is an assignment, Proposition 1 tells us that we can build a NFA by an appropriate weighting of the probabilities $\mathbb{P}(c_{\max}(X_a) \leqslant \delta)$ or $\mathbb{P}(c_{\text{sum}}(X_a) \leqslant \delta)$, that we now compute.

Proposition 12 (Max displacement probability bound)**.** *Let* a *be an assignment of size* k*, and* δ *a positive real number, we have*

$$\mathbb{P}(c_{\max}(X_a) \leqslant \delta) \leqslant (\pi \cdot \delta^2)^k \tag{6.3}$$

Proof — Assume that $a = \{i_1 \to j_1, ..., i_k \to j_k\}$, and call $\bar{B}(x, r)$ the closed disc with center x and radius r.

$$\mathbb{P}(c_{\max}(X_a) \leqslant \delta) = \prod_{p=1}^{k} \mathbb{P}\left(Y_{j_p} \in \bar{B}(X_{i_p}, \delta)\right) \leqslant (\pi \cdot \delta^2)^k$$

because the area of $\bar{B}(x, \delta) \cap \Omega$ is bounded from above by $\pi \cdot \delta^2$ for all x. □

Proposition 13 (Sum displacement probability bound)**.** *Let* a *be an assignment of size* k*, and* δ *a positive real number, we have*

$$\mathbb{P}(c_{\text{sum}}(X_a) \leqslant \delta) \leqslant (2\pi \cdot \delta^2)^k / (2k)! \tag{6.4}$$

Proof — Keeping the notation of the proof of proposition 12, we will show more generally that for $\alpha \geqslant 1$, and $\{X_1, ..., X_k\}, \{Y_1, ..., Y_k\}$ uniform iid. in $[0,1] \times [0,1]$,

$$\mathbb{P}\left(\|Y_1 - X_1\|^\alpha + ... + \|Y_k - X_k\|^\alpha \leqslant \delta\right) \leqslant C_k \cdot (\pi \delta^{2/\alpha})^k$$

where $C_k = (2/\alpha)^{k-1} \prod_{i=1}^{k-1} \beta\left(2/\alpha, (2/\alpha)i + 1\right)$ and $\beta(x, y) = \int_0^1 t^{x-1}(1-t)^{y-1}$. Defining $Z_p = \|Y_p - X_p\|^\alpha$ and $S_{k-1} = Z_1 + ... + Z_{k-1}$, we note that

$$\mathbb{P}\left(\|Y_p - X_p\|^\alpha \leqslant \delta\right) = \mathbb{P}\left(\|Y_p - X_p\| \leqslant \delta^{1/\alpha}\right) \leqslant \pi \delta^{2/\alpha}$$

We will denote $f(\delta) = \mathbb{P}(Z_p \leqslant \delta)$ and $g(\delta) = \pi\delta^{2/\alpha}$. Hence, we have $f(\delta) \leqslant g(\delta)$. The function g defines a measure μ having density $\prod_i g'(Z_i)$. We first prove that $\mathbb{P}(Z_1 + ... + Z_k \leqslant \delta) \leqslant \mu(Z_1 + ... + Z_k \leqslant \delta) = \int \mathbb{1}_{z_1 + ... + z_k \leqslant \delta} g'(z_1)...g'(z_k) dz_1...dz_k$. Let $\bar{g} = \min(g, 1)$, let $U_1, ..., U_k$ be k independent uniform variables on $[0,1]$, let $X_1, ..., X_k = f^{-1}(U_1), ..., f^{-1}(U_k)$ and $Y_1, ..., Y_k = \bar{g}^{-1}(U_1), ..., \bar{g}^{-1}(U_k)$, thus $X_p \geqslant Y_p$ for all p, and

$$\begin{aligned}
\mathbb{P}(X_1 + ... + X_k \leqslant \delta) &\leqslant \mathbb{P}(Y_1 + ... + Y_k \leqslant \delta) \\
&= \int \mathbb{1}_{y_1 + ... + y_k \leqslant \delta} \bar{g}'(y_1)...\bar{g}'(y_k) dy_1...dy_k \\
&\leqslant \int \mathbb{1}_{y_1 + ... + y_k \leqslant \delta} g'(y_1)...g'(y_k) dy_1...dy_k \\
&= \mu(Y_1 + ... + Y_k \leqslant \delta)
\end{aligned}$$

We have $\mu(Y_1 \leqslant \delta) \leqslant \pi\delta^{2/\alpha}$, and assuming $\mu(S_{k-1} \leqslant \delta) \leqslant C_{k-1}(\pi\delta^{2/\alpha})^{k-1}$, we have

$$\begin{aligned}
\forall\delta > 0, \quad \mu(S_{k-1} + Z_k \leqslant \delta) &\leqslant \int_0^\delta g'(t)\mu(S_{k-1} \leqslant \delta - t)dt \\
&\leqslant \int_0^\delta \frac{2}{\alpha}\pi t^{\frac{2}{\alpha}-1} C_{k-1}\left(\pi(\delta-t)^{\frac{2}{\alpha}}\right)^{k-1} dt \\
&\leqslant \frac{2}{\alpha}C_{k-1}(\pi\delta^{\frac{2}{\alpha}})^k \int_0^\delta (\frac{t}{\delta})^{\frac{2}{\alpha}-1}(1-\frac{t}{\delta})^{\frac{2}{\alpha}(k-1)}\frac{dt}{\delta} \\
&\leqslant \frac{2}{\alpha}C_{k-1}(\pi\delta^{\frac{2}{\alpha}})^k \int_0^1 u^{\frac{2}{\alpha}-1}(1-u)^{\frac{2}{\alpha}(k-1)}du \\
&\leqslant \frac{2}{\alpha}\cdot\beta\left(\frac{2}{\alpha}, \frac{2}{\alpha}(k-1)+1\right)\cdot C_{k-1}\cdot(\pi\delta^{\frac{2}{\alpha}})^k
\end{aligned}$$

and we deduce the sought result by induction:

$$\mathbb{P}\left(\|Y_1 - X_1\|^\alpha + \ldots + \|Y_k - X_k\|^\alpha \leqslant \delta\right) \leqslant C_k \cdot (\pi\delta^{2/\alpha})^k.$$

Using the fact that for integers p and q greater than 1, $\beta(p,q) = (p-1)!(q-1)!/(p+q-1)!$, we easily show that when $\alpha = 1$ we obtain $C_k = 2^k/(2k)!$, and we derive the announced probability bound (6.4)

$$\mathbb{P}\left(\sum_{p=1}^k \|Y_p - X_p\| \leqslant \delta\right) \leqslant \frac{(2\pi\cdot\delta^2)^k}{(2k)!} \tag{6.5}$$

Similarly when $\alpha = 2$, we obtain $C_k = 1/k!$ and we can hence also derive that if we observed the squared distances rather than the distances, the probability would become

$$\mathbb{P}\left(\sum_{p=1}^k \|Y_p - X_p\|^2 \leqslant \delta\right) \leqslant \frac{(\pi\cdot\delta)^k}{k!} \tag{6.6}$$

This will be used later to compare the NFA on distances with the NFA on squared distances, since the fact that our generative model uses a Gaussian displacement suggests that we should measure the global error by summing up the squared distances. □

To weigh the structures, we chose to group the assignments together according to their size, and divide the set of assignments into $\min(N,M)$ groups $\mathbb{A} = \mathbb{A}_1 \cup \ldots \cup \mathbb{A}_{N\wedge M}$. An assignment a belonging to group $\mathbb{A}_k$ has weight $w_a = \min(N,M)\cdot|\mathbb{A}_k|$, where the cardinality of the set $\mathbb{A}_k$ of all assignments of size k is $|\mathbb{A}_k| = \binom{N}{k}\cdot\binom{M}{k}\cdot k!$.

Proposition 14 (Maximal displacement NFA). *The family of functions* $(\mathrm{NFA}^m_a)_{a\in\mathbb{A}}$ *defined by*

$$\forall k, \forall a \in \mathbb{A}_k, \quad \mathrm{NFA}^m_a(\delta) = \min(N,M)\cdot\binom{N}{k}\binom{M}{k}k!\cdot(\pi\cdot\delta^2)^k \tag{6.7}$$

is a Number of False Alarms for the measurement $c_{\max}$.

Proposition 15 (Sum displacement NFA). *The family of functions* $(\mathrm{NFA}^s_a)_{a\in\mathbb{A}}$ *defined by*

$$\forall k, \forall a \in \mathbb{A}_k, \quad \mathrm{NFA}^s_a(\delta) = \min(N,M)\cdot\binom{N}{k}\binom{M}{k}k!\cdot(2\pi\cdot\delta^2)^k/(2k)! \tag{6.8}$$

is a Number of False Alarms for the measurement c_{sum}.

Definition 18 (Meaningful assignment)**.** *We say that the realization* $\mathbf{z}$ *of an assignment* X_a *is* ε-meaningful *for the sum (resp. max) displacement criterion if*

$$\mathrm{NFA}^s_a(c_{\mathrm{sum}}(\mathbf{z})) \leqslant \varepsilon \quad (\textit{resp. } \mathrm{NFA}^m_a(c_{\max}(\mathbf{z})) \leqslant \varepsilon),$$

and that it is meaningful *if it is* 1*-meaningful (see Figure 83).*

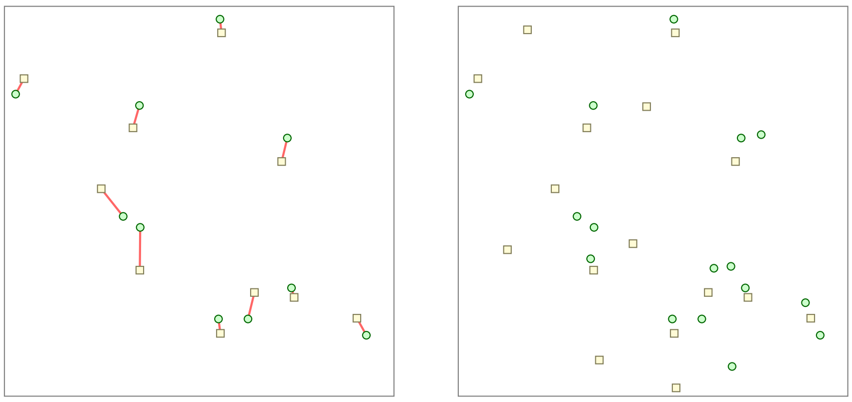


Figure 83: The left figure depicts a meaningful assignment for the sum criterion ($\log \mathrm{NFA}^s = -7.3$). The points of the first image (yellow squares) are linked to their corresponding points in the second image (green circles). The minimal number of points to add to each image such that the assignment is no longer meaningful is computed to be 6 (then, $\log_{10} \mathrm{NFA}^s = 0.7$). The second figure depicts the same points, with 6 randomly added points to visualize a corresponding realization. The assignment shown in the first figure would not be meaningful in the second figure, but this does not mean that no assignment would be detected in this figure.

6.1.3 Most meaningful assignment extraction, largest meaningful assignment extraction

The reader should make sure to mark the fundamental difference between the two approaches described above. The greedy meaningful pairings detection works like the trajectories detection by iteratively extracting as many meaningful pairings as possible, while the NFA threshold ensures that the number of false alarms is controlled. Raising the threshold would hence increase the recall and lower the precision. On the other hand, it often makes no sense to iteratively extract meaningful assignments, as when two assignments are compatible and meaningful, their union is generally more meaningful than each of them – this is not necessarily the case, but on experiments, we found that it almost always is the case.

We can however choose to either extract the *most meaningful* assignment, or the *largest* meaningful assignment. In the former case, raising the NFA threshold will usually have no effect, since the most meaningful assignment in the data might well already be below the threshold, while in the latter case, raising the NFA threshold will allow us to detect larger assignments, thus increasing the recall and lowering the precision. We now describe algorithms for these two extraction processes.

Let $s_X = \{\mathbf{x}_i\}_{1 \leqslant i \leqslant N}$ and $s_Y = \{\mathbf{y}_j\}_{1 \leqslant j \leqslant M}$ be the points detected respectively in the first and second images. We would like to detect $(\varepsilon-)$meaningful assignments in this data for the maximal (resp. sum) displacement criterion, that is, detect assignments $a = \{i_1 \to j_1, \ldots, i_k \to j_k\}$

such that $\mathbf{z}_a = \{\mathbf{x}_{i_1} \to \mathbf{y}_{j_1}, ..., \mathbf{x}_{i_k} \to \mathbf{y}_{j_k}\}$ verifies $\mathrm{NFA}^m_a(c_{\max}(\mathbf{z}_a)) \leqslant \varepsilon$ (resp. $\mathrm{NFA}^s_a(c_{\mathrm{sum}}(\mathbf{z}_a) \leqslant \varepsilon$).

The computation of a most meaningful assignment or a largest meaningful assignment for the maximal displacement NFA amounts to compute, for all $1 \leqslant k \leqslant \min(N, M)$, the solution (or more precisely, a solution) a^m_k to the k-cardinality bottleneck assignment problem

$$a^m_k \in \operatorname*{arg\,min}_{a \in \mathbb{A}_k} c_{\max}(\mathbf{z}_A) = \operatorname*{arg\,min}_{a \in \mathbb{A}_k} \max_{i \to j \in a} \|\mathbf{y}_j - \mathbf{x}_i\|.$$

The most meaningful assignment $\hat{a}^m$ realizes

$$\hat{a}^m \in \operatorname*{arg\,min}_{a \in \{a^m_k\}_{1 \leqslant k \leqslant N \wedge M}} \mathrm{NFA}^m_a(c_{\max}(\mathbf{z}_a)),$$

and the largest meaningful assignment is $\hat{a}^{m,\ell} = a^m_{\hat{k}}$, where

$$\hat{k} = \max\left\{k \mid \mathrm{NFA}^m_{a^m_k}(c_{\max}(\mathbf{z}_{a^m_k})) \leqslant \varepsilon\right\}.$$

Similarly, the computation of a most meaningful assignment or a largest meaningful assignment for the sum displacement NFA amounts to compute, for all $1 \leqslant k \leqslant \min(N, M)$, the solution a^s_k to the k-cardinality linear assignment problem

$$a^s_k \in \operatorname*{arg\,min}_{a \in \mathbb{A}_k} c_{\mathrm{sum}}(\mathbf{z}_A) = \operatorname*{arg\,min}_{a \in \mathbb{A}_k} \sum_{i \to j \in a} \|\mathbf{y}_j - \mathbf{x}_i\|$$

and detect the most meaningful assignment $\hat{a}^s$, realizing

$$\hat{a}^s \in \operatorname*{arg\,min}_{a \in \{a_k\}_{1 \leqslant k \leqslant N \wedge M}} \mathrm{NFA}^s_a(c_{\mathrm{sum}}(\mathbf{z}_a))$$

or the largest meaningful assignment $\hat{a}^{s,\ell} = a^s_{\hat{k}}$, where

$$\hat{k} = \max\left\{k \mid \mathrm{NFA}^s_{a^s_k}(c_{\mathrm{sum}}(\mathbf{z}_{a^s_k})) \leqslant \varepsilon\right\}.$$

Solving the k-cardinality bottleneck or linear assignment problems can be done efficiently using the classical algorithms described in appendix A, leading to algorithms of computational complexity $\mathcal{O}(N^5)$ for the maximal displacement NFA, and $\mathcal{O}(N^4)$ for the sum displacement NFA (assuming $N = M$).

Remark 3 (Most meaningful assignment for the maximal displacement NFA)**.** *Generally, there will be more than one most meaningful assignment (or largest meaningful assignment) for the maximal displacement* NFA*: since the value of the* NFA *depends only on the size of the assignment and on the value of the largest pairing, we may switch some other pairings without changing both the size and maximal cost of the assignment.*

In the experiments of Section 6.2 below, we usually directly choose the one returned by the k-cardinality bottleneck assignment algorithm, but we could choose to extract the assignment having the minimal sum-cost among all the most meaningful assignments for the max-cost, for instance.

6.1.4 Asymptotic behaviors

A strength of the a-contrario framework is that it is possible to study the behavior of the algorithms by studying the NFA criteria equations (6.2), (6.7) and (6.8), and deduce which algorithm is better suited for each particular type of data.

Maximal distance between points for NFA^p *(pairings) and* NFA^m *(global assignments)*

We want to understand in which case one should use the NFA^p greedy pairings extraction, or the criterion NFA^m on the global assignment (for the maximum distance cost).

We observe two images containing N points, and we assume that all of them are paired between the images. For $\varepsilon > 0$, we denote δ_c^p the critical distance such that $\mathrm{NFA}^p(\delta_c^p) = \varepsilon$, and similarly δ_c^m the critical distance such that $\mathrm{NFA}^m(\delta_c^m) = \varepsilon$, that is

$$\log \pi {\delta_c^p}^2 = \log \varepsilon - 2\log N, \quad \text{and} \tag{6.9}$$

$$\log \pi {\delta_c^m}^2 = \frac{1}{N}\big(\log \varepsilon - \log N - \log N!\big). \tag{6.10}$$

From Stirling's formula, one easily derives that

$$\log \frac{\delta_c^p}{\delta_c^m} = \frac{1}{2}\log\frac{\varepsilon}{eN} + \frac{3}{4N}\log N + \underset{N\to+\infty}{\mathcal{O}}(1/N),$$

and therefore for large values of N, δ_c^p is greater than δ_c^m if and only if $\varepsilon > eN$, which seems to confirm our intuition that extracting assignments globally using NFA^m will enable us to detect points further apart, since for reasonable values of ε, the critical distance of NFA^p is greater than that of NFA^m only when the number of points N is very small (see Figure 84).

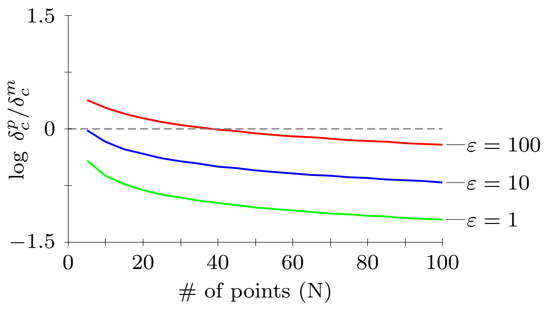


Figure 84: Plot of $\log \delta_c^p/\delta_c^m$ for various values of the maximal NFA ε and number of points N. We should use NFA^p when the curve is above the dotted line, and NFA^m when it is below. We observe that δ_c^m gets quickly larger than δ_c^p, reinforcing the intuition that working with assignments globally rather than with individual pairings enables us to detect pairings that are farther apart.

Maximal distance between points for NFA^m *and* NFA^s

We now study in what cases we should rather use NFA^s than NFA^m. We assume that the two images contain N detected points, $k = \eta N$ of which are paired between the images, the other being noise points. We denote δ_c the critical distance such that $\mathrm{NFA}_a^m(\delta_c) = 1$, that is

$$\log \pi \delta_c^2 = -\frac{1}{\eta N}\left(\log N + 2\log\binom{N}{\eta N} + \log(\eta N)!\right) \tag{6.11}$$

Using Stirling's formula again, one derives that

$$\log\binom{N}{\eta N} = -Nh(\eta) - \frac{1}{2}\log N + \underset{N\to+\infty}{\mathcal{O}}(1)$$

where $h(\eta) = \eta \log \eta + (1-\eta)\log(1-\eta)$. Hence, as N increases,

$$\pi\delta_c^2 = \underset{N\to+\infty}{\mathcal{O}}(1/N).$$

To compare this with the detection threshold for NFA^s, we assume that all the pairs in the assignment have the same cost, that is, the cost of the assignment is $k\delta_c$. Using similar computations, we also derive that $\pi\delta_c^2 = \mathcal{O}(1/N)$ as N increases if the assignment is meaningful. We observe however that NFA^m is in practice much better in this particular case (where all the points in the assignment have been displaced at a distance of exactly δ_c) than NFA^s (see Figure 85).

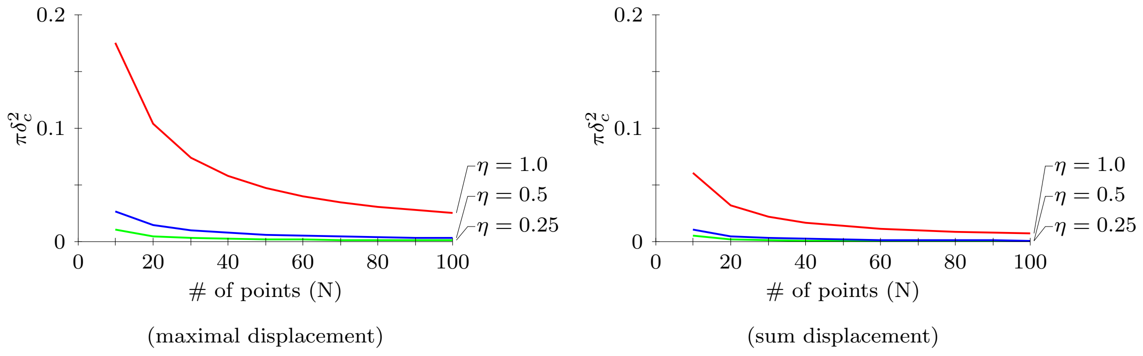


Figure 85: Plot of the maximal value of $\pi\delta_c^2$ such that an assignment of size $k = \eta N$ is meaningful for various values of η, when N varies from 10 to 100. When N grows large, $\pi\delta_c^2$ decreases like $1/N$. The left figure represents the critical value $\pi\delta_c^2$ for the max NFA, the right figure represents $\pi\delta_c^2$ for the sum NFA, when we assume that every pairing in the assignment has the maximal cost δ_c, and thus the sum assignment has total cost $k\delta_c$. We see that in this particular case, NFA^m allows for a greater distance between the points.

Relation between NFA^m *and* NFA^s

When all the pairings are maximally separated, we should thus rather use NFA^m, but in practice, this is rarely the case. We now study what happens in the more general case where the pairings all have different costs.

We observe two images, each containing N points, and k of them belonging to the underlying assignment a. We assume that the maximal and the sum displacements of a are respectively δ_m and δ_s (and therefore $\delta_s \leqslant k\delta_m$), and we study the ratio $\mathrm{NFA}_a^m(\delta_m)/\mathrm{NFA}_a^s(\delta_s)$ in the case where $\delta_s = \gamma \cdot k\delta_m$, with $\gamma \in [0,1]$.

Proposition 16. *When* k *is large,* $\mathrm{NFA}_a^s(\delta_s) \leqslant \mathrm{NFA}_a^m(\delta_m)$ *if and only if* $\gamma < \sqrt{2}/e \approx 0.52$ *(see Figure 86).*

Proof — Since $\mathrm{NFA}_a^m(\delta_m)/\mathrm{NFA}_a^s(\delta_s) = (2k)!/(2\gamma^2k^2)^k$ (not depending on N or δ_m), we easily derive from Stirling's formula again that

$$\log\left(\frac{\mathrm{NFA}_a^m(\delta_m)}{\mathrm{NFA}_a^s(\delta_s)}\right) = k\left(\log(2/\gamma^2) - 2\log e\right) + \underset{k\to+\infty}{\mathcal{O}}(\log k),$$

which implies the sought result. $\square$

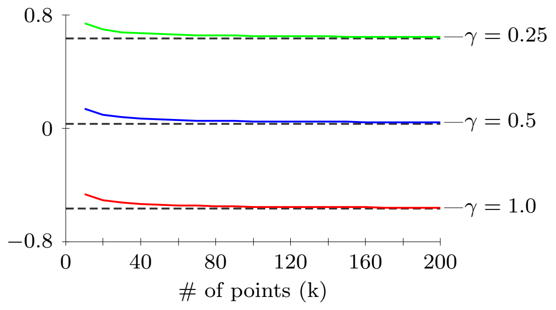


Figure 86: Plot of the ratio $\frac{1}{k}\log\left(\frac{\mathrm{NFA}^m_a(\delta_m)}{\mathrm{NFA}^s_a(\delta_s)}\right)$ when k varies from 10 to 200, and $\delta_s = \gamma \cdot k\delta_m$ for various values of the average cost γ of a pairing relative to the maximal cost of a pairing. The ratio does not depend on the particular value of δ_m, and tends to its limit value $\log(2/\gamma^2) - 2\log e$ (dashed lines) when k increases. When $\gamma \leqslant 0.52$, the curve is above 0 and thus the value of NFA^s is smaller (that is, NFA^s enables more detections) than NFA^m.

We have shown that theoretically, when $\gamma < \sqrt{2}/e$, one should rather use NFA^s than NFA^m, but what are the expected values of γ in practice? Let us assume that we have k points in the first image being displaced according to a certain probability distribution in the second image, that is, $\delta_m = \max_{1 \leqslant i \leqslant k} \|N_i\|$ and $\delta_s = \sum_{i=1}^{k} \|N_i\|$ where the $\{N_i\}_{1 \leqslant i \leqslant k}$ are the random variables corresponding to the displacement of each point.

We have experimentally computed the expectation of $\gamma = \delta_s / k\delta_m$ for a range of values of k and a variety of distributions, as well as the expectation of the ratio $\frac{1}{k}\log\left(\frac{\mathrm{NFA}^m(\delta_m)}{\mathrm{NFA}^s(\delta_s)}\right)$ (see Figure 87), leading us to a choice of a particular algorithm for each displacement distribution: we observe that NFA^s is better suited for Gaussian or Laplace distributions, whereas NFA^m is superior for uniform distributions. This is verified by running both algorithms on real data generated by these displacement models and computing the experimental average recall and precision, although the actual difference in performance is rather small (see Table 10).

	r_m	r_s	p_m	p_s	d_m	d_s
Gaussian	0.49	0.57	0.64	0.64	80%	68%
Laplace	0.52	0.62	0.74	0.73	91%	84%
uniform	0.49	0.49	0.57	0.55	69%	50%

Table 10: Recall and precision obtained with the NFA^m and NFA^s algorithms for various models of the point displacement. The parameters for the generative model are $p_R = 0.8$ and $N = M = 10$, and the parameter for the distributions is chosen such that the displacement on both coordinates has mean 0 and variance $\sigma^2 = 0.01$. The results are averages of 10 000 repetitions, and only problems were a meaningful assignment has been detected are taken into account (the detection ratios d_m and d_s are shown as a reference). The difference in recall and precision between the models is small, but coherent with our predictions. We observe that NFA^m has higher detection ratios than NFA^s on all models.

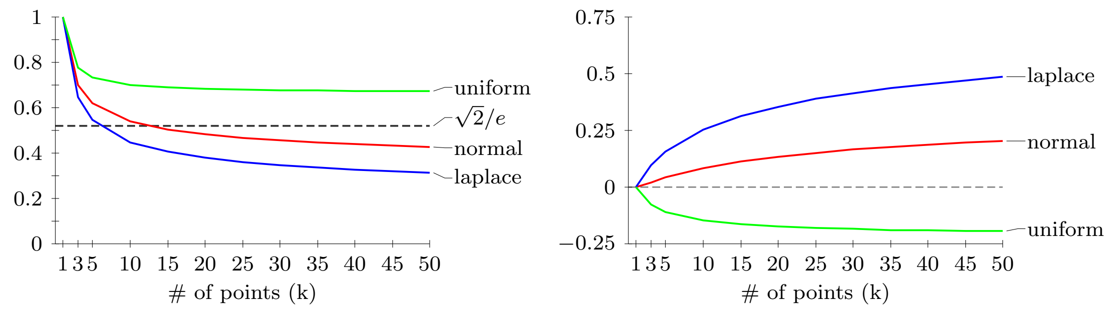


Figure 87: Numerical approximation of $\mathbb{E}[\gamma] = \mathbb{E}[\delta_s/k\delta_m]$ (left) and $\mathbb{E}\left[\frac{1}{k}\log\left(\frac{\mathrm{NFA}^m_k(\delta_m)}{\mathrm{NFA}^s_k(\delta_s)}\right)\right]$ (right) as k varies from 1 to 50 where $\delta_s = \sum_i \|N_i\|$ and $\delta_m = \max_i \|N_i\|$, for various distributions of the variables $\{N_i\}_{1\leqslant i\leqslant k}$ (the N_i are either drawn uniformly in a disc centered on the origin, or from a centered Gaussian or Laplace distribution on each coordinate, the exact parameters of those distributions not being relevant as we study the normalized ratio δ_s/δ_m). The values displayed are averages of 20 000 draws of the N_i. For a large k, if $\gamma \leqslant \sqrt{2}/e$, we know that $\mathrm{NFA}^s \leqslant \mathrm{NFA}^m$ (left figure). However, in practice, even for small k, we observe that $\mathrm{NFA}^s \leqslant \mathrm{NFA}^m$ for the Gaussian and Laplace distributions, while NFA^m seems to be better suited in the case of a uniform distribution (right figure).

6.2 EXPERIMENTS

6.2.1 Most meaningful assignment and largest meaningful assignment

As we discussed in Section 6.1.3, when using the NFA criteria on complete assignments (NFA^m and NFA^s) we only make one detection – and we may either extract the most meaningful assignment or the largest meaningful assignment. When does it make sense to use one or the other?

Obviously, there will be a trade-off between the recall and the precision: detecting the most meaningful assignment will ensure a higher precision, while detecting the largest meaningful assignment will increase the recall. To quantify these behaviors and confirm our intuition, we show the mean recall and precision for each algorithm on data generated using the generative model of section 5.1.3 with $p_R = 0.5$ and $N = M = 10$, for various values of σ^2 (see Figure 88 and table 11).

We observe that for average or large displacement variances – that is, complex problems – one should rather extract largest meaningful assignments, as the precision is equivalent and the recall slightly higher. For small displacement variances, on should choose the appropriate extraction method depending on whether a high recall or a high precision is more important.

6.2.2 Choice of the NFA parameter

We often say that the NFA methods are parameterless, yet we have to choose the maximal allowed value ε of the NFA to know when to stop extracting assignments in order to avoid false detections. For many NFA algorithms, the value $\varepsilon = 1$ is, if not the best, at least a natural choice – and they thus become essentially parameterless. To quantify the role of ε in the case of assignment extraction, we observe the behavior of the recall and precision as we vary it for each NFA criterion (see Figure 89).

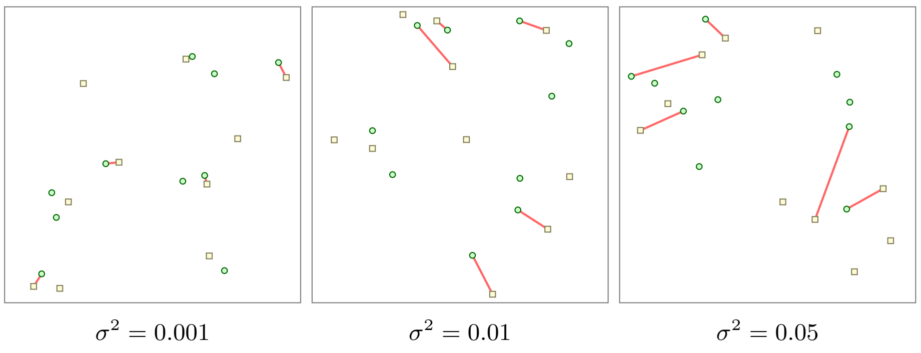


Figure 88: Illustration of the assignments obtained using the generative model of section 5.1.3 with $p_R = 0.5$ and $N = M = 10$, for various values of the displacement variance σ^2, which measures the complexity of the problem. We here show assignments having exactly 5 points for illustration purposes, but the exact percentage of points in the assignment is random (with mean p_R).

NFA^m	r_m	r_ℓ	p_m	p_ℓ	d
$\sigma^2 = 0.001$	0.82	0.89	0.84	0.75	97%
$\sigma^2 = 0.01$	0.39	0.45	0.5	0.5	50%
$\sigma^2 = 0.05$	0.13	0.14	0.22	0.22	26%

NFA^s	r_m	r_ℓ	p_m	p_ℓ	d
$\sigma^2 = 0.001$	0.9	0.9	0.82	0.62	95%
$\sigma^2 = 0.01$	0.4	0.47	0.5	0.48	39%
$\sigma^2 = 0.05$	0.1	0.14	0.21	0.22	17%

Table 11: The recall and precision obtained using the NFA^m and NFA^s algorithms when extracting the most meaningful assignment (r_m) or the largest meaningful assignment (r_ℓ). These numbers are the averages computed on instances where a meaningful assignment has been detected (as a reference, we show the percentage d of the 1000 problems where the algorithms have detected a meaningful assignment). Extracting the largest meaningful assignment generally results in a higher recall and a lower precision. When the displacement variance is average or high, the largest meaningful assignment exhibits a larger precision for a comparable recall. When the displacement variance is low, the table suggests that extracting the most meaningful assignment for NFA^s is preferable.

As a general rule in NFA methods where several structures are detected in an image, increasing the value of ε increases the number of detections and thus the recall, and decreasing the value of ε reduces the number of detections and thus also the number of false detections. This is what we observe when extracting individual pairings with NFA^p (note that when ε is really small, the number of detections is very small and therefore the precision is not reliable, this is why the curves of Figure 89 drop for small values of ε). Indeed, in this case, ε is really the parameter of the underlying algorithm – essentially a nearest-neighbor with a threshold on the maximal distance for the pairings.

However, when we detect complete assignments using NFA^m, increasing the NFA threshold will usually lower both the precision and the recall. In this case indeed, increasing the NFA threshold no longer corresponds to simply increasing the number of pairings in the assignment,

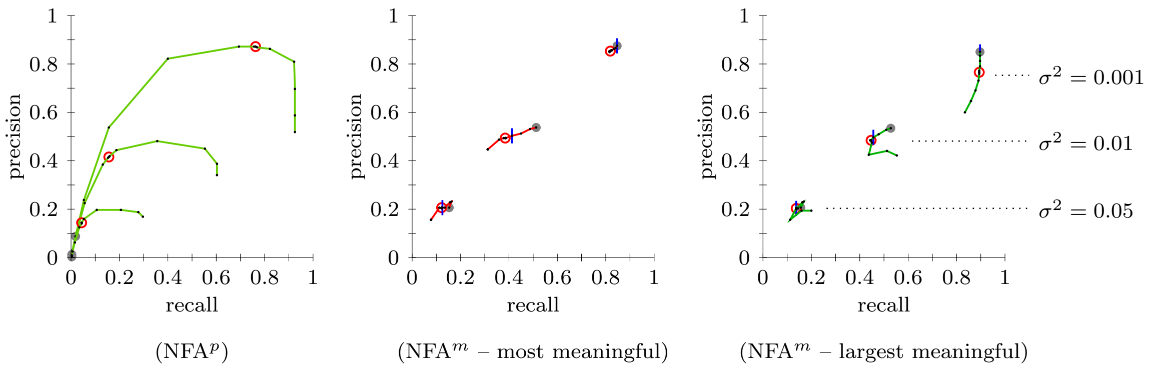


Figure 89: (Effect of the parameter ε on NFA^p and NFA^m.) We display the mean recall and precision of NFA^p and NFA^m (both when when extracting the most meaningful and the largest meaningful assignment) as the ε parameter varies from 10^{-5} (marked by a gray dot on the graphs) to 10^{10} ; the special value $\varepsilon = 1$ is indicated by a red circle. The shown values have been obtained from 10 000 runs on data generated using the generative model of Section 5.1.3, where $p_R = 0.5$, $N = M = 10$, and each curve has been generated using a different value of the displacement variance σ^2, measuring the complexity of the generated assignments. Keep in mind, and this is very important to understand the behavior of the algorithms, that the results of the algorithms are taken into account in the graph *only when a detection has been made by the algorithm* – otherwise the result is dropped from the graph. The number of detections is shown in Table 12.
When extracting individual pairings with NFA^p (left figure), increasing the threshold ε does increase the recall and decrease the precision as one would expect. We note however that for an average or high displacement variance, the value $\varepsilon = 1$ is quite far from the optimum.
On the other hand, when detecting complete assignments with NFA^m (middle and right figures), increasing the value of the NFA threshold tends to lower both the recall and the precision (but also increases the number of problems where a meaningful assignments is detected – not shown on this graph). This is because increasing the value of the threshold does not simply results in more pairings being added, but can result in a completely different found assignment. Note however that the amplitude of the precision and recall changes are much smaller than those of NFA^p – and thus the algorithm seems more robust to parameter variations.
When extracting the most meaningful assignment for very small or very large values of σ^2, changing the value of ε has essentially no effect. For medium values of σ^2, it appears that $\varepsilon = 1$ is far from the optimal value, but in practice, using smaller values of ε to increase the precision and the recall considerably diminishes the number of detections. In order to visualize this, we added a vertical bar that marks the point on the curve separating the values of ε such that the algorithm made a decision in at least 20% and in less than 20% of the cases – the value $\varepsilon = 1$ is close to this threshold. The behavior when extracting the largest meaningful assignment (right figure) is similar. Of course, if we had shown values for ε farther below 10^{-5}, the NFA^m algorithm would have stopped making detections, and the precision and recall would have dropped to 0.

but might completely change the pairings detected in the found assignment, because of the global nature of the algorithm. When extracting the most meaningful assignment, increasing ε will not even have an effect on the algorithm in the case where an assignment was already detected for a lower value of ε – precisely because it extracts the assignment of smallest NFA!

When using NFA^m, the effect of increasing the parameter ε is seen not so much on the average performance on the cases where a detection is made, but rather because *it increases the number of those cases where an assignment is detected*.

Let us explain this: a low NFA threshold will allow the detection of a meaningful assignment only when the problem at hand is very simple, and thus only in few problems – but then the detected assignment will often be correct, and yield both a high recall and a high precision. On the other hand, a high NFA threshold will increase the number of problems on which a detection is made, but will have no effect on problems where an assignment is already detected at a lower threshold (when extracting the most meaningful assignments). Thus, problems where assignments are now detected will tend to be more complex, the detected assignment will be less often correct, and the overall recall and precision will decrease.

In this case, the trade-off controlled by ε is not really between the recall and precision, but between the recall, the precision and *the average number of cases where an assignment is detected*; which must be high enough for the algorithm to be useful. Table 12 shows the average percentage of detections on the generated problems for various values of ε. The value $\varepsilon = 1$ seems to be a good compromise for a high recall and precision, while keeping a fairly high number of detections for reasonably complex problems.

$\varepsilon =$	10^{-2}	10^{-1}	1	10^{1}	10^{2}
$\sigma^2 = 0.001$	65%	85%	97%	100%	100%
$\sigma^2 = 0.01$	5%	18%	52%	100%	100%
$\sigma^2 = 0.05$	1%	5%	25%	100%	100%

Table 12: The percentage of problems where an ε-meaningful assignment has been detected by NFA^m for various values of the maximal NFA parameter ε. The problems are generated using the Gaussian generative model, with parameters $N = M = 10$, $p_R = 0.5$ and various values for the variance of the displacement σ^2; and the values are computed on 10 000 repetitions.

6.2.3 Performance of the NFA algorithms

The results obtained using NFA^s and NFA^m are essentially similar, and although we might think that an NFA algorithm working on the squared distances between the points (using Equation 6.6) might be better suited for the Gaussian generative model, it actually also exhibits a similar behavior, as shown in Table 13.

Therefore, in order to simplify the graphs of this section, we will only compare the performances of NFA^p and NFA^s (using the most meaningful assignment extraction) to those of the Maximum Likelihood algorithm described in Section 5.1.3 (the Maximum A Posteriori algorithm behaves essentially like ML). The choice of NFA^s rather than NFA^m is motivated by the fact that the criterion works with the sum of the distances, just as the ML algorithm does. Similarly, in order to improve readability, we will display the results of the algorithms for only one parameter ε. We will use the canonical value $\varepsilon = 1$ for NFA^s, and the value $\varepsilon = 10$ for NFA^p, which both seem to be good compromises between recall and precision on the synthetic

	recall	precision	detection ratio
NFA^m	0.49	0.64	80%
NFA^s	0.57	0.64	68%
NFA^{s^2}	0.56	0.64	75%

Table 13: The precision and recall obtained using the NFA^m, NFA^s and the NFA^{s^2} algorithm working on squared distances (based on Equation 6.6). We also display the detection ratio for each algorithm. The parameters for the generative model are $N = M = 10$, $p_R = 0.8$ and $\sigma^2 = 0.01$. The results are averages of 10 000 repetitions. The NFA^m algorithm is slightly less suited for Gaussian displacements, but the NFA^s and NFA^{s^2} behave essentially the same, the major difference being the proportion of problems where a meaningful assignment is detected.

data we used. The behavior of the NFA^p algorithm as ε varies is coherent with the findings of the previous section, and an example is shown on Figure 90 as a reference.

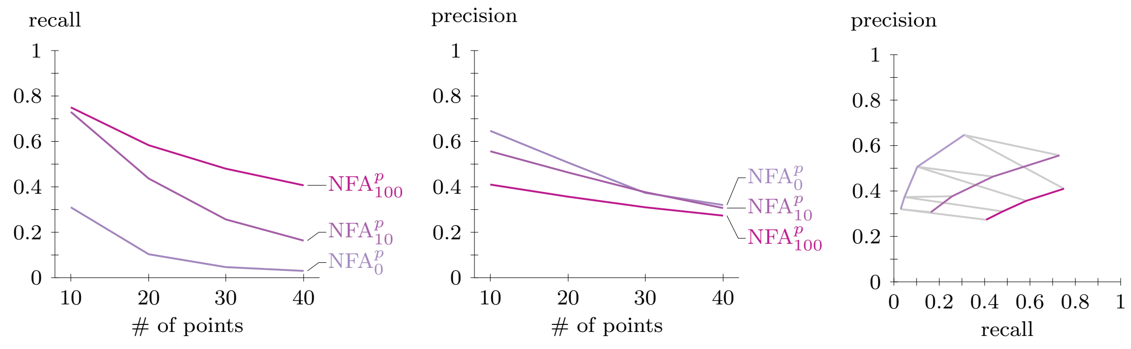


Figure 90: Recall and precision of the algorithm NFA^p for various values of the threshold parameter ε. The data have been generated using the Gaussian generative model, with parameters $N = M$ varying from 10 to 40, $p_R = 0.5$ and $\sigma^2 = 0.005$. As expected, an increase in ε increases the recall and decreases the precision. The value $\varepsilon = 10$ seems to be a good compromise between recall and precision, and we will use this value in further comparisons.

The performances are compared on data generated using the Gaussian generative model of Section 5.1.3, where we vary the total number of points, the amplitude of the displacement or the proportion of noise points (see Figure 91).

The results show that assignments in the problems quickly becomes indistinguishable from pure noise as far as the NFA criterion is concerned, and hence the NFA algorithms become much more conservative than ML. This ensures a higher precision, but the number of detections and the recall quickly drop when the complexity of the problems increases.

6.2.4 Filtering and parameterless algorithms

Simple filtering

If we are certain that there is an assignment in the data and we know how to adapt the parameters of the ML algorithm to the data, the results from the previous section suggests that

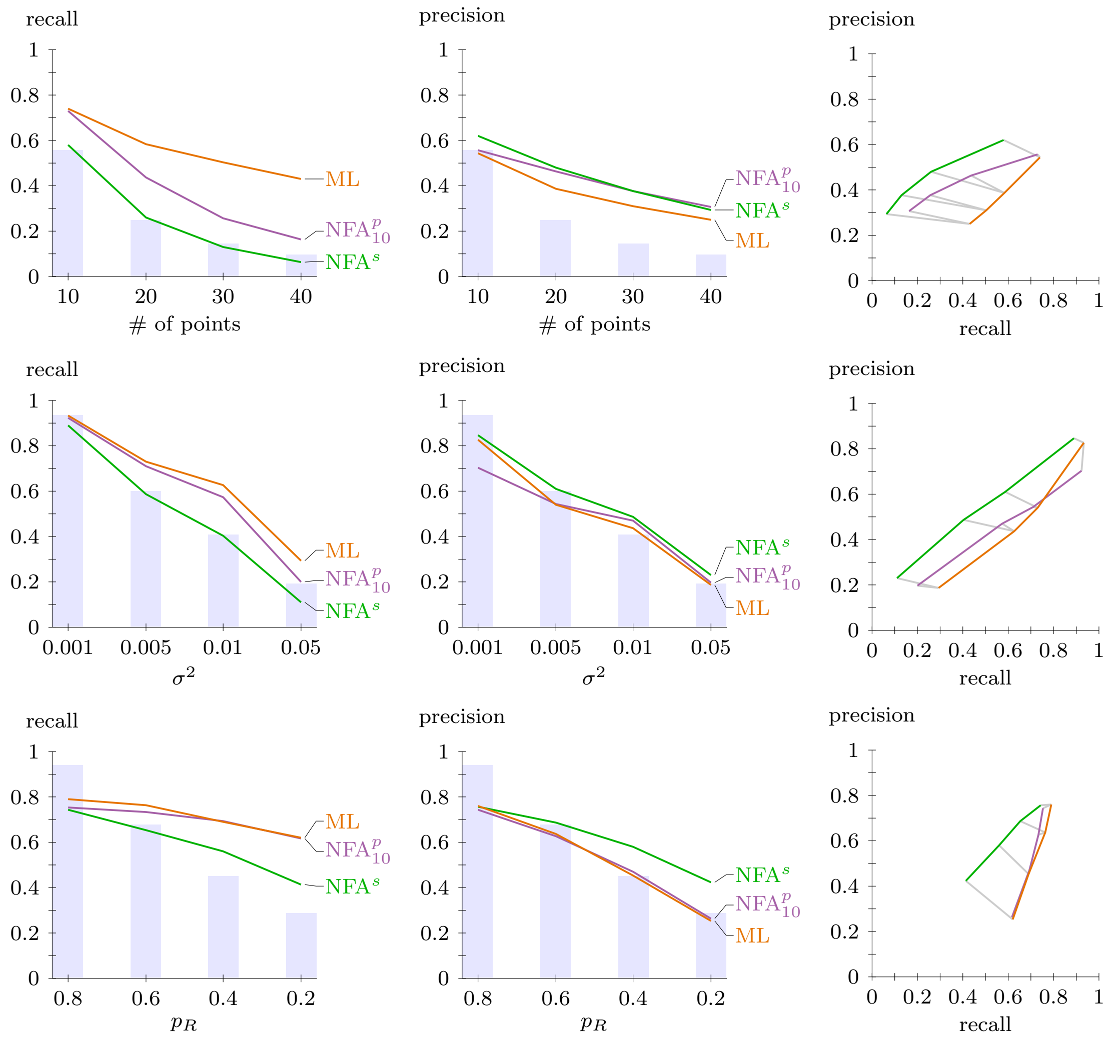


Figure 91: Recall and precision of the ML, NFA^s and NFA^p algorithms on data generated using the Gaussian generative model with default parameters $N = M = 10$, $p_R = 0.5$ and $\sigma^2 = 0.005$, but those are varied in each experiment. The first experiment varies the number of points N from 10 to 40 to increase the density of the points, the second experiment varies the value of σ^2 from 0.001 to 0.05 to increase the amplitude of the points displacements and the third experiment varies the value of p_R from 0.8 to 0.2 to increase the ratio of noise points. The ML algorithm has parameter σ^2 (the exact σ^2 used to generate the data), NFA^s has parameter $\varepsilon = 1$ and NFA^p has parameter $\varepsilon = 10$. The blue bars shows the proportion of data where NFA^s has detected an assignment. When the problems get more complex, the NFA algorithms get quickly more conservative and make very few detections, and have thus a lower recall, and a slightly higher precision. Thus, this data seems to indicate that one should rather use the ML criterion to detect assignments. Keep in mind however, that in this case the ML parameter has been chosen optimally, and it might be difficult in practice to pick the best parameter when the data has not been generated using a theoretical model.

one should rather use the ML algorithm, or any other algorithm that expects a structure to be found in the data, and extracts from all the possible structures the one that optimizes an appearance criterion. On the other hand, if the fact that no assignment be detected in noise is mandatory, or if the optimal ML parameters are difficult to find, NFA algorithms become viable alternatives.

To illustrate this, we choose the parameter of the Maximum Likelihood algorithm in such a way that it makes almost as few detections as the NFA^s algorithm when they are presented to pure noise data (with parameter $\varepsilon = 1$) and we compare the average recall and precision for both algorithms (see table 14). In this table, the ML algorithm is deemed to have made a detection if it has detected at least one pairing. This might seem overly conservative, but other means of comparison, such as choosing the ML parameter to have it detect as many pairings between two random images on average as the NFA^s algorithm does not change our results very much and we keep the exact same conclusions. Note also that if we were to choose the parameter of the NFA^p algorithm in a similar way (we would obtain $\varepsilon = 0.16$), it would behaves essentially like ML – it is thus less suited than the global assignment detection approach at avoiding detections in noise.

	r_{NFA^s}	r_{ML}	p_{NFA^s}	p_{ML}	$d^a_{\mathrm{NFA}^s}$	d^a_{ML}
$\sigma^2 = 0.001$	0.89	0.32	0.83	0.90	95%	77%
$\sigma^2 = 0.005$	0.59	0.16	0.63	0.68	58%	37%
$\sigma^2 = 0.01$	0.40	0.12	0.49	0.50	38%	26%

Table 14: When presented to problems containing 10 random points in each images, the NFA^s algorithm with parameter $\varepsilon = 1$ detects an assignment in about 14% of the cases. We chose the largest parameter σ^2_{ML} for ML such that it has the same detection ratio in pure noise (where an assignment is detected if ML detected at least one pairing), yielding $\sigma^2_{\mathrm{ML}} \approx 3.2 \cdot 10^{-5}$. We display the recall and precision, as well as the detection ratio, obtained by both algorithms on the same data generated using the Gaussian generative model with parameters $N = M = 10$, $p_R = 0.5$, and various values for the variance of the displacement amplitude σ^2. The values are averages of 10 000 repetitions. We observe that for an equivalent level of detection in noise, the NFA algorithm has a slightly lower precision, but a much higher recall and detection ratio than ML.

Requiring the ML algorithm to make very few detections in pure noise drastically decrease its recall compared to NFA^s, and the NFA^s criterion thus appears useful to discriminate noise from signal, while keeping a relatively high level of recall.

Another way to illustrate this difference in behavior is presented in Figure 92, where we generate images that either contain pure noise or an assignment, and display the proportion of instances in which NFA^s and ML correctly make no detection when presented to a pure noise problem as their parameters vary. Obviously, NFA is better at detecting the presence of the assignments, since this is in essence why we constructed this criterion in the first place.

A benefit of the NFA criteria is that they give a way to filter the output of any other assignment algorithm. Using the NFA criterion of our choice (NFA^m, NFA^s or NFA^p), we can reject non-meaningful assignments to improve the precision of classical algorithms when they are presented with noisy data.

There are several ways to filter the results of a detection algorithm: directly applying the NFA criterion to the detected assignment, and discarding it if it is not meaningful, or choosing the sub-assignment of the detected assignment that exhibits the minimal NFA (possibly discarding it if it is not meaningful), or even extracting the largest meaningful sub-assignment of

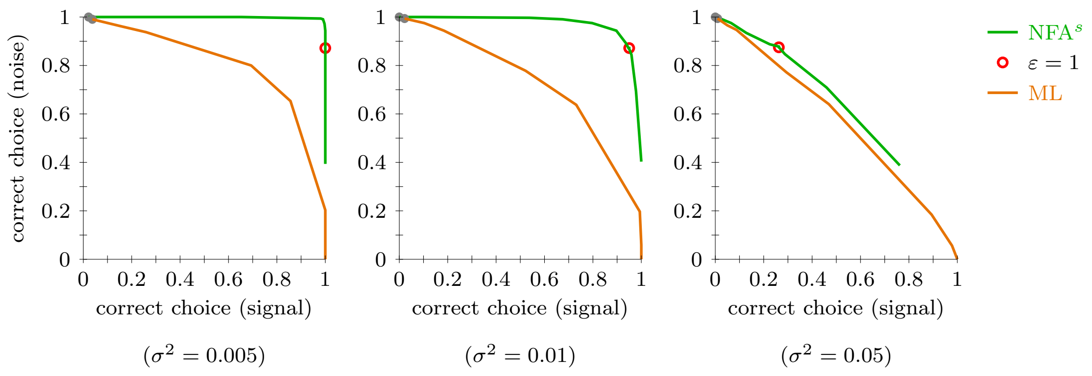


Figure 92: We generate 1 000 problems containing pure noise, and 1 000 problems containing assignments generated with the Gaussian generative model, with parameters $N = M = 10$, $p_R = 1.0$ and various values for σ^2. We display for both algorithms NFA^s and ML, and for a variety of parameters for those algorithms, the proportion of cases where each algorithm has made a detection when an assignment as present (no matter whether the detection is the correct assignment or not) and the proportion of cases where each algorithm has made no detection when presented with a pure noise problem. A detection is made when ML returns an assignment containing at least one pairing, and when NFA^s returns a meaningful assignment. The gray disc shows the point corresponding to the minimal parameter for each algorithm ($\varepsilon = 10^{-8}$ for NFA^s and $\sigma^2_{ML} = 10^{-6}$ for ML), and the red circle corresponds to $\varepsilon = 1$ for the NFA^s algorithm. We observe that NFA is consistently better suited at detecting the presence or absence of an assignment.

the detected assignment. The method we found to give the best results is to simply discard the assignments that are not meaningful for the NFA^s or NFA^m criterion (see Table 15).

	r_{NFA^s}	r_{ML}	$r_{ML_{NFA^s}}$	p_{NFA^s}	p_{ML}	$p_{ML_{NFA^s}}$	$d^a_{NFA^s}$	d^a_{ML}	$d^a_{ML_{NFA^s}}$
$\sigma^2 = 0.001$	0.89	0.88	0.9	0.83	0.61	0.67	95%	100%	74%
$\sigma^2 = 0.005$	0.59	0.74	0.78	0.63	0.51	0.60	58%	100%	37%
$\sigma^2 = 0.01$	0.40	0.61	0.65	0.49	0.42	0.50	38%	100%	17%

Table 15: We display the recall, precision and detection ratio of NFA^s, ML with parameter $\sigma^2_{ML} = 0.01$, and the filtered results of ML where only meaningful assignments have been kept. Filtering ML with NFA^s increases both the recall and the precision, but lowers the number of detections significantly, which is necessary to ensure that the algorithms detect only few assignments in pure noise (NFA^s and ML$_{NFA^s}$ detect an assignment in pure noise in about 14% of the cases, while ML almost always detect an assignment). We observe that ML$_{NFA}$ has a significantly higher recall, but lower precision and detection ratio than NFA, for equivalent detection ratios in pure noise. Hence, the choice between using the NFA algorithm directly, or using an algorithm having a high recall and filtering it to improve its precision boils down to a recall and precision compromise.

Parameterless algorithms

The reader should make sure to mark that the extraction of the most meaningful assignment for the maximal displacement criterion NFA^m can be seen as the selection of the parameter k in the k-cardinality assignment problem, by choosing the k that maximizes $\text{NFA}_k^m(c_{\max}(\mathbf{z}_{a_k}))$ (see section 6.1.3). We can use the same idea to derive new parameterless algorithms for any assignment algorithm by discretizing a reasonable range of values for the parameters, solving the corresponding assignment problem for each value, and keeping the most meaningful assignment (if it is meaningful, that is, of NFA less than 1). We explore this idea with the ML algorithm and the NFA^s criterion to see how parameter selection behaves in practice (see Figure 93). The results suggest that parameter selection using NFA combined with a filtering to keep only meaningful assignments results in a high-precision algorithm, but with a quickly plummeting detection ratio. However, if it is mandatory that only few detections are made in pure noise, this might become a promising parameterless algorithm.

6.3 APPLICATION TO PARAMETERLESS TRACKING, AND USE WITH QUANTIZED DATA

6.3.1 Parameterless trajectory tracking

Many trajectory tracking algorithms use the point correspondence problem to link points from frame to frame. The interest of using the parameterless NFA correspondence algorithms in such a setting is that it would automatically yield a parameterless tracking algorithm, that might be particularly useful when tracking points in presence of noise, since this is their main strength.

The state-of-the-art tracking algorithm ROADS [Veenman, Reinders, and Backer, 2003b] to which we compared ASTRE in Chapter 3 is an extension of the simpler algorithm GOA [Veenman, Reinders, and Backer, 2001]. They both use the point correspondence solver at their core for frame to frame tracking – although ROADS modifies it to take into account several frames at once (typically, a window of 2 or 3 frames). In order to focus on the simple point correspondence problem and understand whether using a parameterless algorithm would affect its performance for this application, we built a simple algorithm inspired by GOA. However, adapting the ROADS algorithm in a similar way should not be much more complicated – in essence, the point correspondence solver used by any algorithm can be readily replaced by any of the NFA algorithms.

Given a sequence of K frames $I_1, ..., I_K$ containing the N_k points detected in the corresponding images from a video sequence, $I_k = \{x_1^k, ..., x_{N_k}^k\}$, where a point can either correspond to a real object in the sequence or to a false detection, we want to extract a set of trajectories, where each trajectory is a sequence of pairs $T = \{(x_1, t_1), ..., (x_n, t_n)\}$ with x_i a point of frame I_{t_i} and $t_i < t_{i+1}$.

The simple tracking algorithm that we propose has three parameters: $\varepsilon > 0$ corresponding to the maximal value of the NFA in the assignment problems, $p_{\min} \geqslant 1$ the minimal number of points that must be present on a trajectory before we start tracking it and $a_{\max} \geqslant 0$ the maximal number of successive frames where we lost the trajectory before we stop tracking it. Note that the parameter ε will usually be set to the canonical value $\varepsilon = 1$, and setting $p_{\min} = 1$ and $a_{\max} = +\infty$ thus renders the tracking algorithm parameterless.

The tracking proceeds in the following way: assume that we have already tracked p trajectories up to frame I_k, and that we want to extend them to frame I_{k+1}. We use the location of the

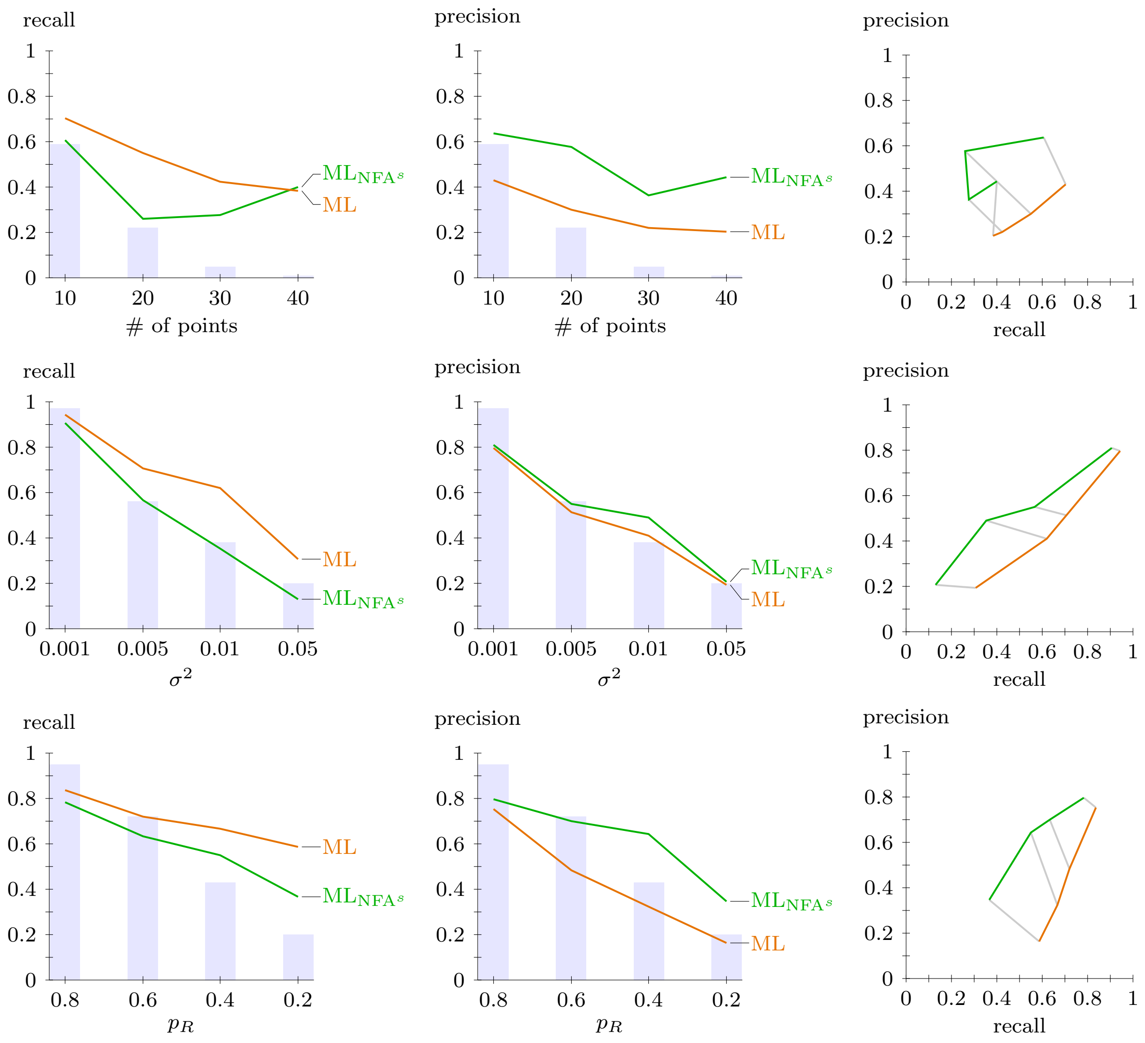


Figure 93: Recall and precision of the ML and $\mathrm{ML}_{\mathrm{NFA}^s}$ algorithms on data generated using the Gaussian generative model with default parameters $N = M = 10$, $p_R = 0.5$ and $\sigma^2 = 0.005$, but those are varied in each experiment. The first experiment varies the number of points N from 10 to 40 to increase the density of the points, the second experiment varies the value of σ^2 from 0.001 to 0.05 to increase the amplitude of the points displacements and the third experiment varies the value of p_R from 0.8 to 0.2 to increase the ratio of noise points. The ML algorithm has been purposedly given the wrong parameter $\sigma^2_{\mathrm{ML}} = 0.05$ to simulate parameter approximation. The $\mathrm{ML}_{\mathrm{NFA}^s}$ chooses the best parameter σ^2_{ML} (among $0.001, 0.005, 0.01, 0.05$) by keeping the one that results in the most meaningful assignment, and the results are then filtered to keep only 1-meaningful assignments. The bars shows the proportion of data where $\mathrm{ML}_{\mathrm{NFA}^s}$ has detected an assignment. The NFA criterion thus enables us to circumvent parameter choice and obtain a parameterless algorithm sporting a fairly high precision.

points in the previous frames to predict their most likely position in the frame I_{k+1} (using a simple linear interpolation). We then solve the correspondence problem between the interpolated trajectories points and the observed detections in frame I_{k+1} using the NFA algorithm and use the found pairings to extend the trajectories. We then try to find candidate trajectories starting in frame I_k by solving the point correspondence problem between the points of frame I_k and the points of frame I_{k+1} that do not belong to any trajectory (we assume that trajectories have null speed when they start). Finally, between each iteration, we end the trajectories that should not be followed anymore because they don't match the requirements from parameters p_{min} and a_{max} (see algorithm 9).

```
Algorithm: extract_trajectories
input  : $I_1, ..., I_K$ the sets of detected points in each image
output: $T$ the set of trajectories

$T \leftarrow \varnothing$
for $1 \leqslant k \leqslant K-1$ do
    $P \leftarrow \varnothing$
    foreach $t \in T$ that is not ended do
        $P \leftarrow P \cup \{$ interpolation of $t$ in frame $k+1$ $\}$
    end

    foreach $(x \to y) \in$ most_meaningful_assignment($P, I_{k+1}$) do
        extend the corresponding trajectory $t$ with $(y, k+1)$
    end

    foreach $t \in T$ do
        if $t$ has less than $p_{min}$ points and could not be extended then
            remove $t$ from $T$
        else if $t$ could not be extended since more than $a_{max}$ frames then
            end the trajectory $t$
        end
    end

    $C_k \leftarrow \{x \in I_k$ that does not belong to a trajectory in $T\}$
    $C_{k+1} \leftarrow \{y \in I_{k+1}$ that does not belong to a trajectory in $T\}$
    foreach $(x \to y) \in$ most_meaningful_assignment($C_k, C_{k+1}$) do
        $T \leftarrow T \cup \{ [(x,k),(y,k+1)] \}$
    end
end
remove trajectories in $T$ having less than 3 points
return T
```

Algorithm 9: Extract trajectories from a sequence of detected points.

Note that `most_meaningful_assignment` can be a routine extracting the most meaningful assignment for any NFA criterion. In practice, the sum criterion seems to work best. Before returning the set of trajectories T, we remove all spurious trajectories having less than 3 points (for instance, all the candidate trajectories between the last two frames).

To assess the performance of the tracking algorithm, we generated random trajectories $t = \{\mathbf{x}_1, ..., \mathbf{x}_K\}$ by drawing $\mathbf{x}_1$ uniformly at random, and iteratively displacing the points using the equation $\mathbf{x}_k = 2\mathbf{x}_{k-1} - \mathbf{x}_{k-2} + N(0, \sigma^2)$ where $N(0, \sigma^2)$ is a Gaussian variable of variance σ^2 and $\mathbf{x}_0 = \mathbf{x}_1$ by convention. We then added noise points uniformly in each image. We display the result of the NFA tracking algorithm and that of the same algorithm where NFA has been replaced with the Maximum Likelihood algorithm (see Table 16).

	recall	precision
$\mathrm{NFA}^s, \varepsilon = 1$	0.93	0.98
ML, $\sigma^2 = 10^{-2}$	0.77	0.5
ML, $\sigma^2 = 10^{-3}$	0.96	0.87
ML, $\sigma^2 = 10^{-4}$	0.97	0.98
ML, $\sigma^2 = 10^{-5}$	0.22	0.98

Table 16: We display the mean recall and precision (in terms of links between two successive points in a trajectory) of the tracking algorithm where the point correspondence problem is either solved using NFA^s (most meaningful assignment extraction), or with ML with various values of σ^2. We generated 10 trajectories and added 10 noise points in each frame, the motion of each trajectory being a constant-speed displacement plus a random Gaussian noise of variance 0.01. The trajectories span 20 frames. The values are averages of 100 repetitions. We observe that the results of the parameterless NFA tracking algorithm are on par with the best results that obtained with a manually chosen parameter for ML.

The results suggest that the parameterless NFA tracking algorithm is suitable for tracking points in noise, and is on par with the best results that can be obtained using the ML algorithm with the best possible parameter.

6.3.2 Application to aggregate tracking

The simple tracking algorithm has been successfully applied to real data in order to track aggregates in bacteria, where we are given the position of the detected aggregates, as well as the bacteria shapes and their lineages (the mother to daughters relationship), and we need to track the aggregates through the sequence and associate them to the unique bacteria they belong to.

We could use the unmodified tracking algorithm and track aggregates with only location information, but we chose a slightly more robust way. We associate to each aggregate all the possible cells it can belong to (some aggregates are not precisely located, and touch several cells). Because the motion of cells is not smooth (they can bump into each other), rather than predicting the motion of an aggregate from its previous locations, we predict it from the average motion of the cells it might belong to.

We also restrict the problem further by only allowing a point from a trajectory that might possibly belong to a set of cells A to a point in frame I_k that might possibly belong to a set of cells B if at least one cell from B is a descendant of a cell from A.

In practice, this motion model and the restrictions suffice to almost perfectly track the aggregates and associate them to a unique lineage (see Figure 94).

6.3.3 Quantization

As was noted in Section 3.2.3, in practical cases, the input data to our algorithms has been quantized (eg. the positions of the points in the images are integers) and the probability computations like $\mathbb{P}(\|Y - X\| \leqslant \delta) \leqslant \pi \cdot \delta^2$ do not hold anymore. Indeed, the probability of two points having the same position in both images is no longer null, and using the above

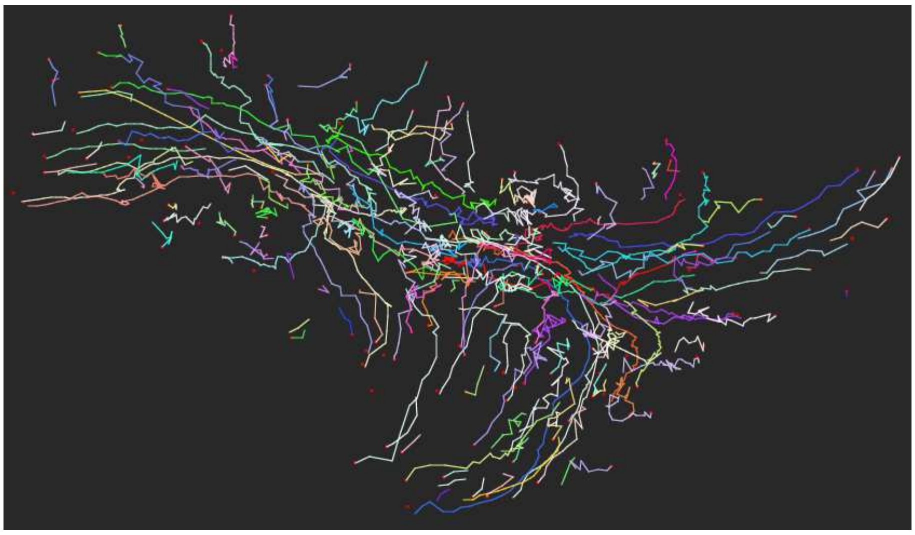


Figure 94: Tracking of cell aggregates.

probability computation in such a case would result in a null NFA for the assignment pairing those two points.

If we assume as in Section 3.2.3 that the data has been quantized on the finite integer grid Ω^d of $\mathbb{Z}^2$ (containing $|\Omega^d|$ pixels), that is, the random variables $X_1, ..., X_M$ and $Y_1, ..., Y_M$ in the naive model are now uniformly distributed in Ω^d, we can define a discrete version of the area $\pi \cdot \delta^2$ used in the computations of the NFA, namely

$$\pi_d(x, y) = \frac{|S_r|}{|\Omega^d|},$$

where $x, y \in \Omega^d$, $r = \|y - x\|$ and $|S_r|$ is the number of pixels enclosed in the discrete disc $S_r = \mathbb{Z}^2 \cap \bar{B}(0, r)$ (see Figure 34 in Section 3.2.3). It follows that

$$\mathbb{P}(\pi_d(X, Y) \leqslant \delta) \leqslant \delta,$$

and in particular, when $x = y$, $\pi_d(x, y) = 1/|\Omega^d|$ which no longer leads to a null NFA.

Discrete pairings NFA *and maximal discrete displacement* NFA

This new measure and probability translates immediately into a corresponding discrete NFA for both the pairings NFA and the maximal displacement NFA.

Proposition 17 (Discrete pairings NFA). *Assume $X_1, ..., X_N$ and $Y_1, ..., Y_M$ are uniformly distributed on the finite integer grid Ω^d of $\mathbb{Z}^2$, the function* NFA^{p,d} *defined by*

$$\text{NFA}^{p,d}(a) = NM \cdot a \tag{6.12}$$

is a Number of False Alarms for the measurement π_d.

Definition 19 (Discrete maximal displacement). *If $a = \{\mathbf{x}_1 \to \mathbf{y}_1, ..., \mathbf{x}_k \to \mathbf{y}_k\}$ is an assignment, the* discrete maximal displacement *of a is defined by*

$$\pi_{d,\max}(a) = \max_{1 \leqslant i \leqslant k} \pi_d(\mathbf{x}_i, \mathbf{y}_i)$$

Proposition 18 (Discrete maximal displacement NFA)**.** *Assume* $X_1, ..., X_N$ *and* $Y_1, ..., Y_M$ *are uniformly distributed on the finite integer grid* Ω^d *of* $\mathbb{Z}^2$*, the family of functions* $(\mathrm{NFA}_A^{m,d})_{A \in \mathbb{A}}$ *defined by*

$$\forall k, \forall A \in \mathbb{A}_k, \quad \mathrm{NFA}_A^{m,d}(a) = \min(N, M) \binom{N}{k} \binom{M}{k} k! \cdot a^k \tag{6.13}$$

is a Number of False Alarms for the measurement $\pi_{d,\max}$.

Let us now examine the performances of these discrete versions of the NFA before we study the discrete sum displacement NFA. Table 17 compares the behaviors of the continuous NFA to their discrete counterparts that run on the same data after it has been quantized on the integer grid, suggesting that the performances of the continuous and discrete algorithms are similar.

	r_{NFA^p}	$r_{\mathrm{NFA}^{p,d}}$	p_{NFA^p}	$p_{\mathrm{NFA}^{p,d}}$	d_{NFA^p}	$d_{\mathrm{NFA}^{p,d}}$
$\sigma^2 = 0.001$	0.93	0.93	0.69	0.69	100%	100%
$\sigma^2 = 0.005$	0.74	0.73	0.56	0.56	100%	100%
$\sigma^2 = 0.01$	0.55	0.54	0.44	0.44	100%	100%

	r_{NFA^m}	$r_{\mathrm{NFA}^{m,d}}$	p_{NFA^m}	$p_{\mathrm{NFA}^{m,d}}$	d_{NFA^m}	$d_{\mathrm{NFA}^{m,d}}$
$\sigma^2 = 0.001$	0.80	0.81	0.82	0.82	96%	97%
$\sigma^2 = 0.005$	0.39	0.39	0.46	0.44	71%	70%
$\sigma^2 = 0.01$	0.19	0.18	0.24	0.23	50%	47%

Table 17: Comparison of the performance of the continuous and discrete versions of the NFA. The data is generated using the Gaussian generative model with parameters $N = M = 10$, $p_R = 0.5$ and various values for σ^2. The values are averages of 1 000 repetitions. NFA^p has parameter $\varepsilon = 10$, while NFA^m has parameter $\varepsilon = 1$. The discrete versions of the algorithms have the same parameters and are given a discretized version of the data on a 100×100 grid. The performances of the continuous and discrete NFA are similar.

Sum discrete displacement NFA

When the random variables lie on the integer grid, an analytic formula estimating a tight upper bound for the probability $\mathbb{P}(\sum_{i=1}^k \|Y_i - X_i\| \leqslant \delta)$ is more involved to derive, and we hence propose to use alternative approaches. We can either derive a larger bound on this probability, alter the original data by randomizing the point location, or numerically approximate the exact probability. We now examine each possibility.

Bound on the sum discrete displacement NFA

We know that $\mathbb{P}(\|Y - X\| \leqslant \delta) \leqslant \pi_d(\delta) = |S_\delta|/|\Omega^d|$, we can derive an upper bound on this probability, for instance the area of the continuous ball of radius $\delta + \sqrt{2}$, that is, $\mathbb{P}(\|Y - X\| \leqslant \delta) \leqslant \pi(\delta + \sqrt{2})^2/|\Omega|$. We can then use this bound to compute a bound on the sought probability as in the proposition 13

$$\mathbb{P}\left(\sum_{i=1}^k \|Y_i - X_i\| \leqslant \delta\right) \leqslant \frac{\left(2\pi(\delta + k\sqrt{2})^2\right)^k}{(2k)!}$$

it is hence sufficient to add $\sqrt{2}$ to all the pairing costs and use the regular NFA^s algorithm.

Proposition 19 (Discrete sum displacement NFA ($\mathrm{NFA}_b^{s,d}$)). *Assume* $X_1, ..., X_k$ *and* $Y_1, ..., Y_k$ *are uniformly distributed on the finite integer grid* Ω *of* $\mathbb{Z}^2$*, the family of functions* $({\mathrm{NFA}_b^{s,d}}_A)_{A \in \mathbb{A}}$ *defined by*

$$\forall k, \forall A \in \mathbb{A}_k, \quad {\mathrm{NFA}_b^{s,d}}_A(a) = \min(N, M) \binom{N}{k} \binom{M}{k} k! \cdot \frac{\left(2\pi(\delta + k\sqrt{2})^2\right)^k}{(2k)!} \tag{6.14}$$

is a Number of False Alarms for the measurement d_{sum}.

This bound is very large and thus not useful in practice: using the Gaussian generative model with parameters $N = M = 10$, $p_R = 0.8$ and $\sigma^2 = 0.005$, we generated 1 000 data and compared the results of the NFA^s algorithm on the data and of the $\mathrm{NFA}_b^{s,d}$ algorithm on the data quantized on a 100×100 grid. The recall, precision and detection ratio drop to half their values when using $\mathrm{NFA}_b^{s,d}$.

Data dequantization

Another approach to circumvent the quantization problem is to dequantify the data by randomly adding a small real vector and using the continuous NFA, resulting in a random algorithm (two runs of which might not result in the same assignment). In practice, we add a uniform displacement in the $[0, 1] \times [0, 1]$ pixel area to each point, and we observe similar performances as those of the NFA algorithm before quantization (see table 18). This approach might thus be a simple way of solving the assignment problem on quantized data, although its drawback is that the algorithm is non-deterministic.

	r_{NFA^s}	$r_{\mathrm{NFA}_r^{s,d}}$	p_{NFA^s}	$p_{\mathrm{NFA}_r^{s,d}}$	d_{NFA^s}	$d_{\mathrm{NFA}_r^{s,d}}$
$\sigma^2 = 0.001$	0.85	0.79	0.78	0.78	95%	95%
$\sigma^2 = 0.005$	0.34	0.34	0.37	0.37	58%	59%
$\sigma^2 = 0.01$	0.15	0.15	0.17	0.17	38%	38%

Table 18: Comparison of the performance of the continuous and randomized versions of the NFA. The data is generated using the Gaussian generative model with parameters $N = M = 10$, $p_R = 0.5$ and various values for σ^2. The values are averages of 1 000 repetitions. Both algorithms have parameter $\varepsilon = 1$, and the randomized NFA is given a discretized version of the data on a 100×100 grid, which has then be randomized by adding a small random quantity to the position of each point. The performances of the continuous and randomized NFA are similar for medium or large displacement variances. For very small displacements such as $\sigma^2 = 0.001$, the recall logically drops a little, since the random perturbations are larger than the expected point displacement.

Numerical approximations

Rather than using a bound or dequantifying the data, we can numerically compute the exact probability of two discrete points being at a certain distance. We assume that the integer grid is the square $\Omega^d = [1, w] \times [1, w]$, and let $\{X_i\}_i$ and $\{Y_i\}_i$ be independent uniform random variables on Ω^d, let $D_i = \|Y_i - X_i\|$ and $S_k = D_1 + ... + D_k$. The probability that an assignment of size k has a sum of distance less than δ in the naive model is

$$\mathbb{P}(S_k \leqslant \delta) = \frac{\left|\{(\mathbf{x}_i, \mathbf{y}_i)_{1 \leqslant i \leqslant k} \in (\Omega^{d^2})^k \; ; \; \sum_{i=1}^{k} \|\mathbf{y}_i - \mathbf{x}_i\| \leqslant \delta\}\right|}{|\Omega^d|^{2k}}$$

An exact computation of this probability is intractable and in practice we need to approximate it. Let $\eta > 0$ be a quantification step, we compute the probability that one of the random variables $D_i = \|Y_i - X_i\|$ belongs to a ring of width η

$$\begin{aligned} \forall 0 \leqslant p \leqslant \left\lfloor \frac{\sqrt{2w^2}}{\eta} \right\rfloor, & \\ u^\eta(p) &= \mathbb{P}\left(p\eta \leqslant D_i < (p+1)\eta\right) \\ &= \frac{\left|\{(\mathbf{x}, \mathbf{y}) \in \Omega^{d^2} \; ; \; p\eta \leqslant \|\mathbf{y} - \mathbf{x}\| < (p+1)\eta\}\right|}{|\Omega^d|^2}, \end{aligned}$$

and denoting L the set of values taken by the variables D_i, we use a recursive computation to bound the probability on the sum of k displacements by

$$\begin{aligned} \mathbb{P}_k^\eta(j) &= \mathbb{P}(S_k \leqslant j\eta) = \mathbb{P}(S_{k-1} + D_k \leqslant j\eta) \\ &= \sum_{t \in L} \mathbb{P}(D_k = t)\mathbb{P}\left(S_{k-1} \leqslant j\eta - t\right) \\ &\leqslant \sum_{p \geqslant 0} \mathbb{P}\left(p\eta \leqslant D_k < (p+1)\eta\right) \mathbb{P}\left(S_{k-1} \leqslant (j-p)\eta\right) \\ &= \sum_{p \geqslant 0} u^\eta(p)\mathbb{P}_{k-1}^\eta(j-p) \\ &= (u^\eta * \mathbb{P}_{k-1}^\eta)(j), \end{aligned}$$

and we finally approximate

$$\mathbb{P}(S_k \leqslant \delta) \leqslant \mathbb{P}_k^\eta(\lceil \delta/\eta \rceil).$$

If η is chosen small enough, the probability is exact (but the time and space required for the computation and storage increase significantly), in practice, for 100×100 images, $\eta = 0.1$ seems to suffice (see Figure 95).

We compare the performance of the discrete NFA with that of the continuous NFA using the same protocol as previously (see Table 19). We observe that the performances of the discrete NFA are slightly better than those of the continuous NFA, even though the data has been quantized. This might be explained by the fact that the upper bound on the probability for the continuous NFA^s (Equation 6.4) is rather loose.

The proposed approximation is still computationally intensive, since the initial u^η computation requires $\mathcal{O}(w^4)$ instructions and $|u| = \lfloor \sqrt{2w^2}/\eta \rfloor$ space, and each successive $\mathbb{P}_j^\eta$ requires $\mathcal{O}(j|u|^2)$ time and $\mathcal{O}(j|u|)$ space, hence, the computation requires $\mathcal{O}(K^2\sqrt{2w^2}/\eta)$ space and $\mathcal{O}(w^4 + 2K^2w^2/\eta^2)$ time, where K is the number of $\mathbb{P}_j$ that we want to compute. However, this precomputation can be done once for a large K and used subsequently for several detections (for images having the same size (w, w) and containing M, N points, where $\min(N, M) \leqslant K$).

We could theoretically improve the time complexity by transforming the data in the Fourier domain to speed up the convolution, but with the several Fourier transform implementations

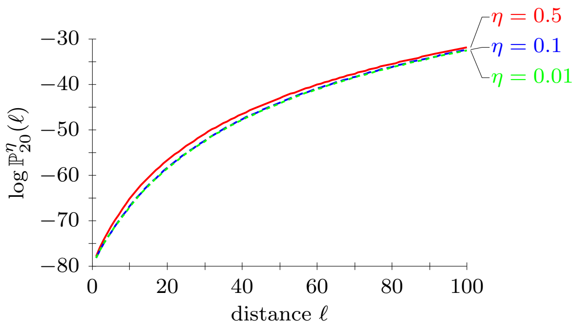


Figure 95: We display $\log \mathbb{P}^{\eta}_{20}(\ell)$ for various values of the approximation parameter η, and ℓ ranging from 1 to 100. We observe that the probability approximation has almost converged for $\eta = 0.1$.

	r_{NFA^s}	$r_{NFA^{s,d}_{\eta}}$	p_{NFA^s}	$p_{NFA^{s,d}_{\eta}}$	d_{NFA^s}	$d_{NFA^{s,d}_{\eta}}$
$\sigma^2 = 0.001$	0.84	0.86	0.78	0.76	95%	95%
$\sigma^2 = 0.005$	0.33	0.4	0.37	0.39	58%	63%
$\sigma^2 = 0.01$	0.15	0.2	0.18	0.2	38%	43%

Table 19: Comparison of the performance of the continuous and discrete versions of the NFA. The data is generated using the Gaussian generative model with parameters $N = M = 10$, $p_R = 0.5$ and various values for σ^2. The values are averages of 1 000 repetitions. Both algorithms have parameter $\varepsilon = 1$, and the discrete NFA is given a discretized version of the data on a 100×100 grid. The performances of the continuous and discrete NFA are similar, and even surprisingly slightly better for the discrete version of the NFA, although the data has been quantized. This might be explained by the fact that the upper bound on the probability for the continuous NFA^s (Equation 6.4) is rather loose.

that we used, this quickly resulted in numerical errors and ultimately to aberrant values for the probabilities.

Also note that we could reduce the complexity of the u^{η} computation to $\mathcal{O}(w^2)$ without degrading the precision too much in practice by approximating

$$\begin{aligned} u^{\eta}(p) &= \mathbb{P}(p\eta \leqslant D_i < (p+1)\eta) \\ &\leqslant \sum_{r \in L, p\eta \leqslant r < (p+1)\eta} |L_r|/|\Omega| \end{aligned}$$

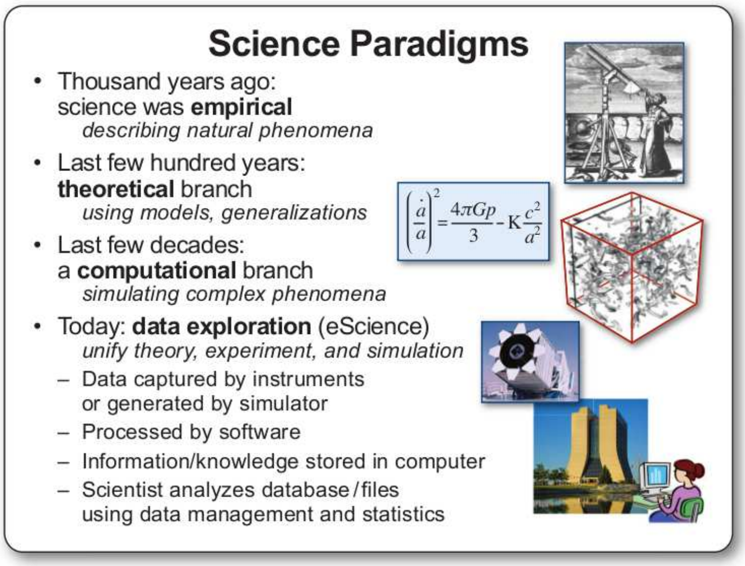
where L_r is the number of points on the integer grid that lie at a distance r from the origin, which is similar to the approximation made in the continuous case, and amounts to neglect all border effects.

7
Conclusion

The last decades have been marked by the astronomical growth of the storage and computation capacities at our disposal. In a recent publication [Hilbert and López, 2011], researchers from the University of South California have estimated the world capacity of storage, communication and computational power, and the number they provide defy our ability to picture them: the authors estimate that in 2007, mankind was able to store 2.9×10^{20} optimally compressed bytes of data, to transfer almost 2×10^{21} bytes and to compute at a rate of about 6.4×10^{18} instructions per second.

In this era of speed and accumulation, scientists work in a radically different way: they record, store, sort, annotate, analyze and publish – in an automated manner if possible – ever larger quantities of data. This scientific exploration paradigm (see Figure 96) has been described in *The Fourth Paradigm for Science* (Hey, Tansley, and Tolle, 2009):

> The world of science has changed, and there is no question about this. The new model is for the data to be captured by instruments or generated by simulations before being processed by software and for the resulting information or knowledge to be stored in computers. Scientists only get to look at their data fairly late in this pipeline. The techniques and technologies for such data-intensive science are so different that it is worth distinguishing data-intensive science from computational science as a *new, fourth paradigm for scientific exploration*.



Science Paradigms

- Thousand years ago:
  science was **empirical**
  describing natural phenomena
- Last few hundred years:
  theoretical branch
  using models, generalizations
- Last few decades:
  a **computational** branch
  simulating complex phenomena
- Today: **data exploration** (eScience)
  unify theory, experiment, and simulation
  - Data captured by instruments
    or generated by simulator
  - Processed by software
  - Information/knowledge stored in computer
  - Scientist analyzes database / files
    using data management and statistics

$$\left(\frac{\dot{a}}{a}\right)^2 = \frac{4\pi G\rho}{3} - K\frac{c^2}{a^2}$$

Figure 96: **The Fourth Paradigm of Science.** Nowadays storage, transfer and computation capacities enable a new scientific paradigm, in which the data play a central role. *Source: Illustration by Jim Gray, excerpt from Hey, Tansley, and Tolle, 2009.*

Let us give some examples of the quantities of data manipulated by nowadays scientists: among 1 700 researchers surveyed in a study by the Science magazine [Science, 2011], 20% regularly use or analyze datasets exceeding 100 gigabytes, and 7% handle datasets of more than 1 terabyte (10^{12} bytes). The *Large Hadron Collider* yields around 1.8 gigabyte of data per second in peak use, totaling about 15 petabytes (15×10^{15} bytes) annually. In 2008, the servers of *Google* processed more than 20 petabytes of data every day [Dean and Ghemawat, 2008].

*

This evolution translates into a strong demand for algorithms that are both *completely automated* – because manual processing is fastidious, time-consuming, and it is also error-prone; and *fast enough* to be applied to the gigantic datasets that the scientific research produces today. In order to address these latter efficiency exigencies, researchers and engineers now often turn themselves to parallelization, enabling them to efficiently harness the graphics processing units, the multi-core processors or the distributed computation grids at their disposal. We have not chosen to focus our attention on this particular concern, although the algorithms that we propose are often naturally parallelizable. We have rather aimed at understanding what the essential requirements are that an algorithm must answer in order to be perceived as completely automated. Such an algorithm must obviously be capable of scaling to large datasets (whose size force them to have a natural inner variability) *without modification, or excessive parametrization*. The algorithms must therefore be *robust to data variation*, and use the physical constants inherent to the data as well as the intrinsic geometry of the images, rather than abstract parameters implicitly defined by one or several algorithmic processing steps. For instance, we use the intrinsic geometrical features of the images (their level lines) in Céleste to renormalize them without having to choose the contrast change parameters manually, the latter being too sensitive to the variability of the acquisition noise. When possible, we think that the algorithms should even be virtually *parameterless*, as the robust ASTRE trajectory tracking algorithm derived from the a-contrario framework in Chapter 3.

We also think that a completely automated algorithm must be *simple enough to understand*, in order to make it possible to predict its behavior theoretically, and to adapt and correct it easily when changes in the acquisition conditions (or the natural variability of the datasets produce errors).

*

The robust Céleste cell tracking algorithm is now used routinely in the biology research laboratory at the TaMaRa's lab from INSERM in Paris, and has increased the quality of the segmentation and tracking results while lowering the data processing time. However, for longer movies, the large number of potential cells to be considered often renders the algorithm unusable if a human operator does not help simplifying the data by manually segmenting some ambiguous cells. Additionally, the presence of a model of the bacteria motion is still somewhat disturbing: we would prefer to reason only using purely photometrical clues. Indeed, in some biological experiments, the nature of the treatment applied to the bacteria might alter their growth or motion properties significantly, and the algorithm might then make wrong segmentation and tracking decisions.

*

As was demonstrated in Chapter 3, the ASTRE a-contrario trajectory detection algorithm yields very good performances in presence of noise, and matches state-of-the-art performances on real-world data, even though it is (virtually) parameterless. This research could be extended in a variety of directions: the fitness measure could be the sum of the accelerations, rather than the max. It could also be adapted to a specific motion in some applications, when the prior knowledge on the motion is more precise than simply having a low acceleration. Similarly, it would also be interesting to be able to take feature descriptors into account in the naive model when they are available. Our preliminary research has shown that this might however prove difficult, and in particular, trying to track points having a scalar intensity in $[0, 1]$ by simply processing the feature as an additional dimension of the spatial point representation yields poor results in practical cases; the proper way to blend the motion and the features into one scalar criterion is yet to be found.

The ASTRE algorithm can be naturally parallelized, but might however remain slow when the length of the sequence and the number of points is very large. It might be interesting to find approximations that would reconstruct a whole trajectory from smaller subtrajectories obtained by restraining the problem to smaller subsequences of the original sequence and see if they yield almost correct results. Additionaly, the limits of the globality in time should also be adressed. Intuitively, if we have a two-hour long video of someone, we might expect that what he does after one hour and a half does not influence how we perceived his motion after 10 minutes. In practice, the ASTRE algorithm will often break a trajectory in two if it detects a sudden change in the acceleration because of a change in behavior. However, a time scale parameter might perhaps help the algorithm make some decisions in some applications.

*

The orthogonal tracking approach to tracking trajectories sequentially and globally in time is to follow all the trajectories at once ("globally in space") from frame to frame; this is the multi-frame point correspondence problem. We have proposed in Chapter 6 a parameterless a-contrario algorithm solving the two-frame point correspondence problem that yields performances on par with classical algorithms having an optimally chosen parameter, and can readily be plugged in other algorithms as a replacement for their point correspondence problem solver, to obtain new parameterless tracking algorithms for example. As in the case of ASTRE, it might be interesting to see whether features could be added to the points.

We have also defined an optimal criterion that bounds the performances of classical point correspondence algorithms (the WRAP criterion in Chapter 5), and we were thus able to study them more precisely. In particular, our analysis has underlined the fact that most of the classical algorithms are not able to detect ambiguities to properly balance their recall and precision. It seems however that there are very few point correspondence problems that are neither very easy or very complex to solve, and simple local algorithms and complex global algorithms yield similar performances. It would be interesting to use this theoretical framework for the analysis of the point correspondence problem to tackle this problem and see whether there is a significant intermediate class of problems where global algorithms perform better than local ones.

*

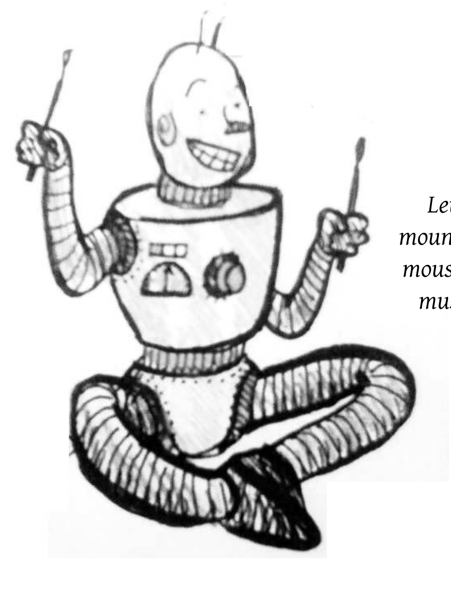
Going further, would it be possible to compute a criterion that is both global in time and in space? Recent research introduced in Jiang, Fels, and Little, 2007 and Berclaz, Fleuret, and Fua, 2009 seems to indicate that it is possible to do so by replacing the formulation of the multi-frame correspondence problem from a boolean "deterministic" setting where two points are either linked by a pairing or not, to a continuous "fuzzy" version that can be efficiently solved using linear programming – and which almost always results in a quasi-boolean assignment between the points. It would be interesting to see whether these promising algorithms could be adapted to handle arbitrary motion models, points entering or exiting the image at any location, and still having only a limited set of parameters.

Howling circles
stomping

turning trumpets
sliding

music is made
from bolts and nuts
light in tubes
plastic wire

Let's shake it like it's Tokyo,
mountains of wisdom and jelly
moustache on a fish and crazy
music made by space robots.



Let's shake it like it's Tokyo,
Maël Primet, july 2010

A
Classical algorithms for the assignment problem

CONTENTS

This section describes the algorithms solving the linear and the bottleneck assignment problems, required in the classical point correspondence solutions, in the WRAP algorithm, and for the NFA assignment algorithms (see chapters 5 and 6).

The algorithms are described in the general case of the linear and the bottleneck assignment problems where all pairings can be added to an assignment, but it is easy to adapt them to the case where some pairings are forbidden (for instance, pairings of cost greater than a constant c_{thre}).

A.1 THE LINEAR ASSIGNMENT PROBLEM

We use the same notations as before: the points in the first image are the $\{\mathbf{x}_i\}_{1\leqslant i\leqslant N}$, those of the second image are the $\{\mathbf{y}_j\}_{1\leqslant j\leqslant M}$, and the pairing costs are the c_{ij}.

We want to compute the solution to any k-cardinality linear assignment problem, that is, given $0 \leqslant k \leqslant N \wedge M$, find

$$a_k \in \operatorname*{arg\,min}_{a\in\mathbb{A}_k} \sum_{i\to j\in a} c_{ij}$$

where $\mathbb{A}_k$ is the set of assignments of size k.

We will actually build the a_k sequentially using a successive shortest path algorithm. More precisely, let $a_0 = \varnothing$ be the empty assignment, and suppose that optimal assignments $a_0, ..., a_{k-1}$ of size $0, ..., k-1$ have already been computed. We show how to compute an optimal assignment a_k among all assignments of size k.

Definition 20 (Assignment graph)**.** *The assignment graph G is a weighted bipartite graph. The first node set S contains the points in the first image, the second node set T contains those in the second image. The edge between a point $i \in S$ and a point j in T is oriented from T to S with cost $-c_{ij}$ if $(i \to j)$ belongs to a_{k-1}, and from S to T with cost c_{ij} otherwise (see Figure 97).*

Definition 21 (Free nodes)**.** *The free nodes are the vertices of the assignment graph that do not belong to a_{k-1}, the free nodes from S are denoted F_S, and those from T are denoted F_T.*

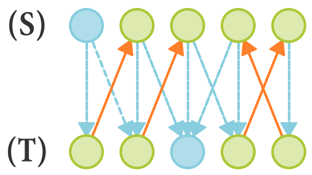


Figure 97: The bipartite graph corresponding to an assignment problem where the points in the first image are represented by the nodes in S, the points in the second image by the points in T, the four pairings in the assignment are the full edges directed from T to S, and the pairings not in the assignment are the dashed edges directed from S to T (not all such pairings have been represented for clarity reasons). The free nodes from F_S and F_T are the nodes (represented in blue) that do not appear in any pairing in the assignment. In this case, there is only one node in F_S and one node in F_T.

Definition 22 (Augmenting path)**.** *An augmenting path π is a path in the assignment graph from a free node in F_S to a free node in F_T. We will write $(i \to j) \in \pi$ where $i \in S$ and $j \in T$ if there is an edge joining i and j in the path π (either from i to j, or from j to i).*

Definition 23 (Path cost)**.** *Let π be an augmenting path. We define the path cost of π as the sum of its edge costs.*

$$c(\pi) = \sum_{(i \to j) \in \pi \setminus a_{k-1}} c_{ij} - \sum_{(i \to j) \in \pi \cap a_{k-1}} c_{ij}$$

Definition 24 (Augmentation along a path)**.** *If π is an augmenting path, we define the augmentation of a_{k-1} along π as the assignment*

$$a = a_{k-1} \oplus \pi = \{ (i \to j) \mid (i \to j) \in (a_{k-1} \cup \pi) \setminus (a_{k-1} \cap \pi) \}$$

The augmented assignment a has size k and cost $c_{\text{sum}}(a) + c(\pi)$ (see Figure 98).

Let us show that we can obtain an optimal assignment among those of size k using an augmentation of a_{k-1} along a shortest path from F_S to F_T the free nodes in each image.

Lemma 1. *Let $h \in \mathbb{A}_k$ be an optimal assignment among assignments of size k. There exists a path π_h in the assignment graph that starts from a free node in F_S appearing in h (we will say that the node is in $h \setminus a_{k-1}$), ends in a free node in F_T appearing in h, and alternates between edges belonging to h and edges belonging to a_{k-1}.*

Proof. Indeed, since h has strictly more edges than a_{k-1}, there exists necessarily a node $x \in S$ in $h \setminus a_{k-1}$. We denote y the node that is paired with it in h, and build the path $\pi = x \to y$. If y does not belong to a_{k-1}, we can stop and choose $\pi_h = \pi$. Otherwise, y is paired in a_{k-1} to

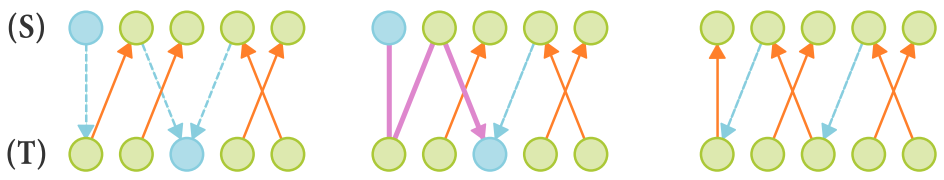


Figure 98: Augmentation of an assignment along a path. Full lines: assignment, dotted lines: unused pairings (not all unused pairings have been represented for clarity), bold lines in the second image: augmenting path. By flipping the edges along the augmenting path, we obtain an assignment containing one additional edge (seen as the set of edges orientated from T to S).

a node $z \neq x$. We can add the edge $y \to z$ to the path π. If the node z does not appear in h, we can remove all the nodes from the path π, which contains as many pairings in h as in a_{k-1}, and iterate the argument on the restrictions $\bar{h}$ of h and $\bar{a}_{k-1}$ of a_{k-1} that do not contain nodes in π, since we still have $|\bar{a}_{k-1}| < |\bar{h}| < |h|$. This operation decreases the number of pairings in the assignments, and can hence only be applied a finite number of times. If the node z belongs to h, we continue by adding its paired node in h to π, and we iterate the process. We must eventually find an augmenting path π_h starting from a free node and ending in a free node. □

Lemma 2. *Let $h \in \mathbb{A}_k$ be an optimal assignment among assignments of size k, and π_h a path as described in lemma 1. The assignment $a = a_{k-1} \oplus \pi_h$ has size k, and is of minimal cost among assignments of size k.*

Proof. By construction, the assignment a coincides with the assignment h on the nodes appearing in π_h. We now consider the restrictions $\bar{h}$ of h and $\bar{a}_k$ of a_k to the nodes not in π_h. The restriction $\bar{a}_k$ coincides with the restriction $\bar{a}_{k-1}$. Those restrictions have the same size p, and by optimality of h and a_{k-1}, they are optimal among the assignments of size p that do not contain nodes from π_h, and thus have the same cost. We finally deduce that the total cost of a is equal to the total cost of h, and thus a is optimal among assignments of size k. □

Proposition 20. *If we augment a_{k-1} along a minimal-cost path from F_S to F_T, we obtain an assignment of size k having minimal cost among such assignments.*

Proof. Let $h \in \mathbb{A}_k$ be an optimal assignment among assignments of size k, and π_h a path as described in lemma 1. Let π be a minimal-cost path from a free node in the first image to a free node in the second image, and let $a_k = a_{k-1} \oplus \pi$. The assignment a_k has size k, and from the optimality of π and lemma 2, its cost is $c_{\text{sum}}(a_k) = c_{\text{sum}}(a_{k-1}) + c(\pi) \leqslant c_{\text{sum}}(a_{k-1}) + c(\pi_h) = c_{\text{sum}}(h)$. Therefore, since h is an optimal assignment of size k, $c_{\text{sum}}(a_k) = c_{\text{sum}}(h)$ and a_k is optimal. □

With a naive implementation of this shortest-path augmentation approach (see Algorithm 10), the complexity is $\mathcal{O}(\min(N, M)NM(N+M)) = \mathcal{O}(N^4)$ (assuming $N = M$), since we repeat $\min(N, M)$ times a shortest-path computation of cost $\mathcal{O}(NM(N+M))$ using the Bellman-Ford shortest-path algorithm that is suited when the edges of G can have negative values.

Note that the algorithm can easily be adapted to the case where some pairings $i \to j$ are forbidden in the assignment, to return the k-cardinality optimal assignments among those where such pairings do not appear.

```
Algorithm: kcard-LAP
input  : $\{c_{ij}\}_{1\leqslant j\leqslant M}^{1\leqslant i\leqslant N}$ the pairing costs
output: The solutions $a_0, ..., a_{N\wedge M}$ of the k-cardinality LAP
$a_0 \leftarrow \varnothing$
for $k = 1$ to $N \wedge M$ do
    $G \leftarrow$ assignment graph for $a_{k-1}$
    $F_S \leftarrow$ free nodes in G (relative to $a_{k-1}$) in the first image
    $F_T \leftarrow$ free nodes in G (relative to $a_{k-1}$) in the second image
    $\pi \leftarrow$ minimal-cost path from $F_S$ to $F_T$
    $a_k \leftarrow a_{k-1} \oplus \pi$
end
return $a_0, ..., a_{N\wedge M}$
```

Algorithm 10: Algorithm computing the k-cardinality LAP solutions. The solution of the general LAP is the solution of cardinality $\min(N, M)$.

A faster augmentation algorithm

This section presents a slight enhancement in the algorithm and can be safely skipped. By modifying the algorithm to ensure that the edges keep positive weights, we can use the $\mathcal{O}(NM)$ Dijkstra algorithm to search for minimal-cost paths, leading to a computational complexity of $\mathcal{O}(\min(N, M)NM) = \mathcal{O}(N^3)$.

This requires adding to the assignment graph a source node s which has a null-cost outgoing edge to each node of the first image, and a destination node t which has a null-cost incoming edge from each node in the second image. Clearly, the shortest paths between a free node from the first image and a free node from the second image are in bijection with the shortest paths from s to t in the augmented graph. If we arbitrarily define potentials $\sigma_i \in \mathbb{R}$ for each node i in the graph, and that we replace the costs c_{ij} by $\bar{c}_{ij} = c_{ij} + \sigma_i - \sigma_j$, then we do not change the shortest paths from s to t in the assignment graph (since we only offset their costs by the constant $\sigma_s - \sigma_t$).

The initial edges costs are positive, and we can thus use Dijkstra to find for each node i the minimal cost δ_i of a path from s to i. Let $\pi = s \to i_1 ... \to i_\ell \to t$ be such a minimal-cost path from s to t. We augment the assignment along this path, and we obtain a new assignment graph where the directions of the edges in π have been inverted and the cost c of those edges have been replaced by $-c$.

Now let us choose potentials in a way that will render all costs positives, which won't change the set of solutions to the problem (the minimal-cost paths are the same), but makes it possible to use the Dijkstra algorithm. We can show easily that the potentials $\sigma_i = \delta_i$ verify this property, where δ_i is the length of the shortest path from s to i. But in order to have a slightly faster algorithm in practice (although having the same worst-case complexity), we will choose the potentials $\sigma_i = \min(0, \delta_i - \delta_{\min})$, where $\delta_{\min}$ is the cost of a shortest path from s to t.

Indeed, let (i, j) be an edge of the graph of cost c. We have $\delta_j \leqslant \delta_i + c$ by definition of shortest paths, and we can thus study the different cases:

- if $\delta_i < \delta_{\min}$ and $\delta_j < \delta_{\min}$, we obtain $\bar{c} = c + \delta_i - \delta_j \geqslant 0$,
- if $\delta_i < \delta_{\min}$ and $\delta_j \geqslant \delta_{\min}$, we obtain $\bar{c} = c + \delta_i - \delta_{\min}$, or $c \geqslant \delta_j - \delta_i$, thus $\bar{c} \geqslant \delta_j - \delta_{\min} \geqslant 0$,

- if $\delta_i \geqslant \delta_{\min}$ and $\delta_j < \delta_{\min}$, we obtain $\bar{c} = c + \delta_{\min} - \delta_j \geqslant 0$ and finally
- if $\delta_i > \delta_{\min}$ and $\delta_j > \delta_{\min}$, we obtain $\bar{c} = c \geqslant 0$.

Using this potentials definition, we only change the costs of the edges adjacent to the nodes verifying $\delta_i < \delta_{\min}$, and since a node is processed by the Dijkstra algorithm in the order of the shortest path from s to the node, we can stop the Dijkstra algorithm as soon as we have reached the node t, and we will only need to update the potentials of nodes that have been processed by the algorithm.

A.2 THE BOTTLENECK ASSIGNMENT PROBLEM

Keeping the same notations as in the previous section, we now want to compute the solutions to the k-cardinality bottleneck assignment problem, that is, for all $1 \leqslant k \leqslant \min(N, M)$, an assignment a_k of size k minimizing

$$a_k \in \operatorname*{arg\,min}_{a \in \mathbb{A}_k} \max_{i \to j \in a} c_{ij}$$

Definition 25 (Partial graphs)**.** *Let $p_1, ..., p_{NM}$ be the sequence of edges $[1, N] \times [1, M]$ sorted by increasing cost $c_p = c_{ij}$ if $p = (i \to j)$. The* partial graph *of order q is the set of pairings $G_q = \{p_1, ..., p_q\}$.*

Clearly, if $q_k = \inf \{q \mid G_q \text{ contains an assignment of size } k\}$, then $a_k \subseteq G_{q_k}$, and

$$c_{q_k} = \min_{a \in \mathbb{A}_k} c_{\max}(a)$$

We deduce that in order to find a most meaningful assignment of size k, it suffices to build the graphs $G_1 \subset G_2 \subset ... \subset G_{NM}$, and to look for the smallest $p \in [1, NM]$ such that G_p contains an assignment of size k.

This can be done efficiently in an iterative way. We start with the empty assignment a_0 and the empty partial graph $G = G_0$. Either the current assignment is of maximal size in the current partial graph (this is the case for a_0 and G_0) and we continue with the next graph (eg. $G = G_1$), or we can modify the assignment to increase its size by one edge in a manner similar to the path augmentation from the previous section. Iterating this process yields the sequence of assignments $a_1, ..., a_{N \wedge M}$ (see Algorithm 11).

Increasing the size of an assignment $a \subseteq G$, or checking whether the assignment is of maximal size in G can be done in a manner similar to the path augmentation of the previous section, as shown in lemma 3.

Lemma 3 (Assignment maximality)**.** *Let a be an assignment contained in the partial graph G. We regard G as an directed bipartite graph between the nodes S representing the points in the first image and the nodes T representing the points in the second image, where edges are oriented from T to S if they are in a, and from S to T otherwise. The sets of free nodes in S and T (relative to a) are denoted F_S and F_T.*

If there exists a path π from F_S to F_T in G, then $a' = a \oplus \pi$ is an assignment containing exactly one more edge than a, that is contained in the graph G. If there exists no such path, a must be of maximal cardinality in G.

Proof. Suppose a is not a maximal cardinality assignment, let b be a maximal cardinality assignment in G, and consider the directed graph G' containing the pairings $i \to j$ appearing only in a or only in b but not in both, directed from S to T if the pairing belongs to b and from T to S otherwise. Clearly, G' is a subgraph of G (we only remove edges that belong to both assignments, and edges from S to T that are not in b). We consider all the connected components of this graph. We necessarily have at least one connected component having more edges appearing in b than in a (otherwise, the number of edges in each assignment would be the same, and a would be maximal). Such a connected component contains an augmenting path, and since G' is contained in G, it is also an augmenting path in G. □

```
Algorithm: kcard-BAP
input  : $\{c_{ij}\}_{1 \leqslant j \leqslant M}^{1 \leqslant i \leqslant N}$ the pairing costs
output: The solutions $a_0, ..., a_{N \wedge M}$ of the k-cardinality BAP

$G \leftarrow \varnothing$
$a_0 \leftarrow \varnothing$
for $k = 1$ to $N \wedge M$ do
    while $a_k$ is undefined do
        $F_S \leftarrow$ free nodes in G (relative to $a_{k-1}$) in the first image
        $F_T \leftarrow$ free nodes in G (relative to $a_{k-1}$) in the second image
        if there is an augmenting path $\pi$ from $F_S$ to $F_T$ in G then
            $a_k \leftarrow a_{k-1} \oplus \pi$
        else
            // $a_{k-1}$ is a maximal assignment in G, add an edge
            $G \leftarrow G \cup \{i \to j\}$ where $(i, j) \in \arg\min_{(i,j) \notin G} c_{ij}$
        end
    end
end
return $a_0, ..., a_{N \wedge M}$
```

Algorithm 11: Algorithm computing the k-cardinality BAP solutions. The solution of the general BAP is the solution of cardinality $\min(N, M)$.

The algorithmic complexity of the extraction of a most meaningful assignment is thus $\mathcal{O}(N^2M^2)$. Indeed, each augmenting path search has a $\mathcal{O}(NM)$ cost (we find any path from F_S to F_T using Dijkstra's algorithm), and we repeat it at most NM times.

B

Installation and usage of the ASTRE software

CONTENTS

B.1 REQUIREMENTS

You need an environment providing

- a C compiler (tested with `gcc` 4.3), as well as
- Python and PyQt4 for the optional visualization tools (see below for the installation)

B.2 INSTALLATION OF THE DEPENDENCIES

B.2.1 cbase

Download and install cbase (version 1.3.5 or greater) from the website `http://www.hyperrealm.com/cbase/`

```
$ tar -xzf cbase-1.3.5.tar.gz
$ cd cbase-1.3.5
$ ./configure
$ make
$ sudo make install
$ cd ..
```

B.2.2 argtable2

Download and install argtable2 (version 1.3 or greater) from the website http://argtable.sourceforge.net/

```
$ tar -xzf argtable2-13.tar.gz
$ cd argtable2-13
$ ./configure
$ make
$ sudo make install
$ cd ..
```

Add the library installation path to the `LD_LIBRARY_PATH` environment variable in your `.bash_profile` or `.profile` file:

```
export LD_LIBRARY_PATH=/usr/local/lib:${LD_LIBRARY_PATH}
```

You can then remove the source directories `cbase-1.3.5` and `argtable2-13` that are no longer needed.

B.2.3 Install ASTRE

Download ASTRE from the website http://www.math-info.univ-paris5.fr/~moisan/astre/ and compile the binaries:

```
$ tar -xzf astre-1.0.tar.gz
$ cd astre-1.0
$ make
```

Add the Python library path to your `PATH` environment variable in your `.bash_profile` or `.profile` file:

```
export PATH=/path/to/astre-1.0/bin:${PATH}
```

B.2.4 Install the visualization tools [optional]

Install Python and PyQt4:

```
$ sudo apt-get install python python-qt4-dev
```

Add the Python library path to your `PYTHONPATH` environment variable in your `.bash_profile` or `.profile` file:

```
export PYTHONPATH=/path/to/astre-1.0/lib/python/:${PYTHONPATH}
```

B.3 USAGE OF ASTRE AND VISUALIZATION OF THE RESULTS

B.3.1 Visualize points description files

Trajectories visualization

If you have installed the Python libraries, you can use the `tview.py` points description files viewer. ASTRE comes bundled with sample data in the `data` directory. To visualize the snow sequence, type

```
$ tview.py data/snow.30Hz.desc
```

and press `f`(orward) and `b`(ackward) to move through the sequence.

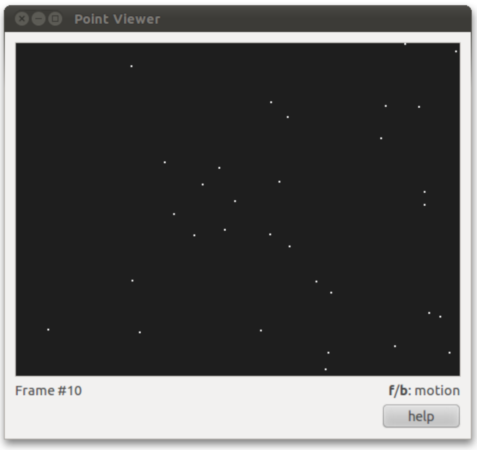


Figure 99: Visualization of the snow sequence

B.3.2 Run ASTRE on the snow sequence

`astre-noholes`

We run ASTRE (without holes) on the trajectory set and visualize the found trajectories using the `-t` option of `tview.py` with parameter $\varepsilon = 0$ (maximal value of $\log_{10}(\text{NFA})$)

```
$ astre-noholes -e 0.0 data/snow.30Hz.desc tjs
$ tview.py -t tjs
```

We can run ASTRE (without holes) with a larger parameter to discover more trajectories:

```
$ astre-noholes -e 5.0 data/snow.30Hz.desc tjs
$ tview.py -t tjs
```

You could do the same with `astre-holes` to obtain the results shown in our paper, but this takes much more time and memory on such a large sequence. You can however use the `-h` parameter of `astre-holes` to limit the size of the holes in a trajectory, and speed up the

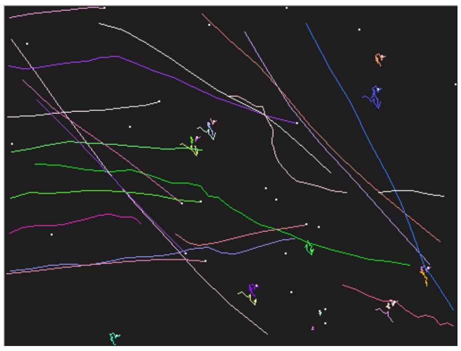


Figure 100: Detections found using the ASTRE algorithm without holes for the parameter $\varepsilon = 0$. Only a few trajectories have been detected.

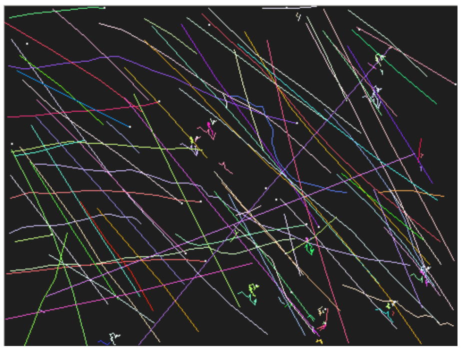


Figure 101: Detections found using the ASTRE algorithm without holes for the parameter $\varepsilon = 5$. This time, the number of detected trajectories is much larger.

computation at the expense of finding suboptimal solutions (the computation took about 2 minutes on a standard laptop):

astre-holes, and constraint on hole size

```
$ astre-holes -e 5.0 -h 1 data/snow.30Hz.desc tjs
$ tview.py -t tjs
```

As a reference, this is the computation times on a standard laptop for astre-holes with parameter $\varepsilon = 0$ and various values of the h parameter on the snow sequence (which has 40 frames):

h	1	2	3	4	10	20	30	40
approx. time (m)	2	6	9	14	31	43	46	45

trajectory identifiers

astre-noholes and astre-holes copy the input points description file to the output and add a column containing the trajectory identifier (or -1 if the point does not belong to a trajectory).

$\log_{10}$(NFA) headers

Note also that astre-noholes and astre-holes add a header for each detected trajectory of the form `traj:<id>:lNFA = <lNFA>` showing the $\log_{10}(\text{NFA})$ of the corresponding trajectory id.

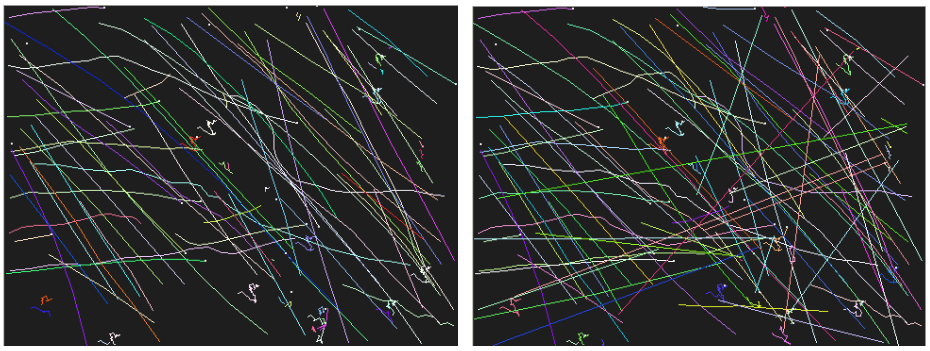

(a) maximal hole size $h = 1$ (b) maximal hole size unbounded

Figure 102: Detections found using the ASTRE algorithm without holes for the parameter $\varepsilon = 5$ when the sizes of the holes (**a**) are limited to one frame to speed-up computations, or (**b**) are unbounded. You can see that almost all the trajectories have been already correctly captured when considering only trajectories with holes of length at most one frame.

B.4 EVALUATION OF ASTRE RESULTS

B.4.1 Algorithm evaluation on real data

A qualitative visual inspection of the found trajectories already gives a rough idea of the algorithms performances. Yet one would sometimes rather have a way to quantify those performances.

recall and precision

The recall and the precision of the detections can be computed using the `tstats` program. Rather than taking into account only whole trajectories (without any missing or spurious point), we chose to compute the recall and precision in terms of the number of correct links found, that is, two consecutive points on a trajectory, possibly separated by a hole. The recall and precision are then

$$\begin{aligned} \text{precision} &= \frac{\text{\# of correct links found}}{\text{\# of links found}}, \\ \text{and recall} &= \frac{\text{\# of correct links found}}{\text{\# of actual links}}, \end{aligned}$$

where a link is real if it belongs to the ground-truth, is found if it belongs to the detected trajectories and is correct if it belongs to both.

The sample snow sequence has been originally acquired at 210Hz to make it possible to reconstruct the ground-truth by hand on this simple to track version, and has then been subsampled to 30Hz to obtain a more challenging to track sequence. The snow sequence thus also contains the ground-truth, making it possible to compute the recall and precision of our results:

```
$ astre-noholes -e 0.0 data/snow.30Hz.desc tjs
$ tstats tjs
[MD] {'recall': 0.497377, 'precision': 0.993711,
      'num_detected_trajs': 39, }
```

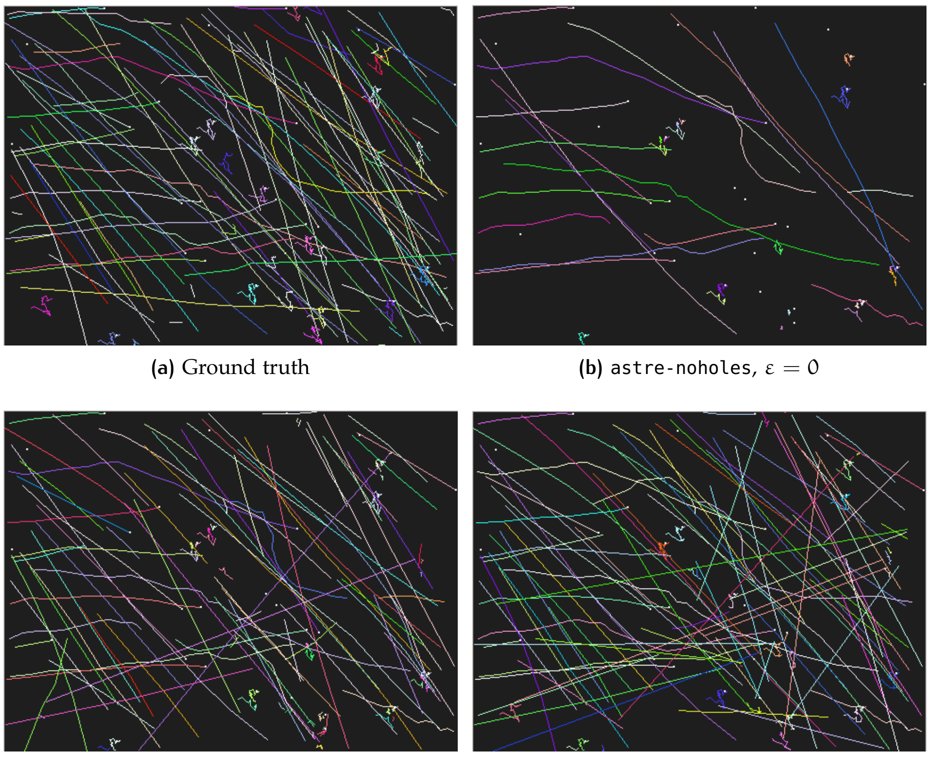

(a) Ground truth (b) `astre-noholes`, $\varepsilon = 0$ (c) `astre-noholes`, $\varepsilon = 5$ (d) `astre-holes`, $\varepsilon = 5$

Figure 103: Visual inspection of the algorithms results on the neige sequence

The `tstats` program expects to be given a points description file with the ground truth as the fourth column and the found trajectories as the last column, but you can also specify the index of the column if needed, and even specify two different input files, one containing the ground truth, the other containing the detected trajectories.

The output array of `tstats` is in Python format, and the results of the experiments can be quickly processed by filtering the lines starting with `[MD]` (stands for metadata) and evaluating the following data structure in Python.

```
>>> statistics = eval("{'recall': 0.497377,
   'precision': 0.993711, 'num_detected_trajs': 39, }")
```

We compare the performances of `astre-noholes` when we increase the maximal value of the $\log_{10}(\text{NFA})$:

```
$ astre-noholes -e 5.0 data/snow.30Hz.desc tjs
$ tstats tjs
[MD] {'recall': 0.709339, 'precision': 0.948107,
      'num_detected_trajs': 124, }
```

Obviously, the number of detected trajectories (and thus the recall) has increased, while the precision has slightly decreased (we made some false detections).

We can now compare this last result with those of `astre-holes` with parameter $\varepsilon = 5$, both when we constrain the maximal size of a hole to 1 and when it is unbounded:

```
$ astre-holes -e 5.0 -h 1 data/snow.30Hz.desc tjs
$ tstats tjs
[MD] {'recall': 0.751312, 'precision': 0.939633,
      'num_detected_trajs': 99, }
```

Since you might not want to run the computation in the unbounded case on your computer, you might want to download the results of ASTRE on the snow sequence from the ASTRE website (`http://www.math-info.univ-paris5.fr/~moisan/astre/`) and compute its performances. In this case, since `lNFA_5.tjs` does not contain both the ground truth and the detected trajectories, but only the latter, we specify both the ground truth file and the detected trajectories file:

```
$ tstats snow.30Hz.desc lNFA_5.tjs
[MD] {'recall': 0.764953, 'precision': 0.895577,
      'num_detected_trajs': 111, }
```

Clearly, we have gained almost nothing in recall, but slightly lost in precision by allowing unbounded hole lengths. However, this might be due to properties of this particular data and might not be the case in a more general setting.

As a reference, here is a table with the recall, precision, and number of detected trajectories for `astre-holes` on the snow sequence with parameter $\varepsilon = 0$ and various values for the maximal hole length parameter `h` (the sequence spans 40 frames):

h	1	2	3–6	7–40
recall	0.56	0.56	0.57	0.58
prec.	0.94	0.93	0.92	0.91
#trajs	39	38	37	39

and when $\varepsilon = 5$:

h	1	2	3	4–6	7–40
recall	0.76	0.77	0.76	0.77	0.76
prec.	0.92	0.91	0.90	0.90	0.90
#trajs	119	112	110	111	108

B.4.2 Systematic algorithm evaluation using synthetic trajectories

synthetic trajectories generation

We generate 5 trajectories spanning 20 frames, add 10 random noise points in each frame and then remove 20% of the trajectory points, and run ASTRE (with holes) with the default parameter $\varepsilon = 0$

```
$ tpsmg -N 10 20 5 pts
$ tcripple -r 20 pts pts
$ astre-holes pts tjs
```

Note that when generating (integer-valued) point description files with `tpsmg`, we forbid two trajectories from sharing a point (otherwise the overlapping trajectory is regenerated), and we do not add noise points at the same location as one trajectory point. By default, all trajectories are constrained to stay in the frame (otherwise they are regenerated), but this behavior can be changed with the `-F` option (see the ASTRE reference in chapter C for a description of the astre options).

Also note that when crippling points description files with `tcripple`, only points belonging to a trajectory are removed (the noise points are kept).

C

User reference for the ASTRE software

CONTENTS

C.1 ASTRE POINTS AND TRAJECTORIES DATA FILE FORMAT

The format of the files is the following

```
type = PointsFile v.1.0
uid = 1
width = 480
height = 360
DATA
0 292 204 4
0 247 304 51
...
```

All the lines before `DATA` are headers of the form `key = value`, and all the lines after `DATA` are array of floats representing one point, the minimum data for a point is 3 column:

```
<frame> <x> <y>
```

but often they have a trajectory identifier

trajectory identifier

```
<frame> <x> <y> <t>
```

where `t` is the id of the trajectory to which the point belong, or `-1` if the point does not belong to a trajectory.

The data entries can be preceded by an optional column name:

```
f:0 x:292 y:204 t:4
...
```

in which case, all the tags of one column must be the same.

If you need it, you can add other data columns to represent the intensity of the points, etc. It is also possible to add length-varying descriptors (for instance, a polygonal shape descriptor) by using special headers: describe each shape with a unique identifier as header entries, and reference the corresponding identifier in your data column:

```
type = PointsFile v.1.0
...
shape:0 = (4.0, 18.0) (17.0, 10.0) ... (14.0, 18.0)
shape:1 = (2.0, 5.0) (14.0, 11.0) ... (5.0, 15.0)
...
DATA
<frame> <x> <y> <shape_id> <trajectory_id>
...
```

points description file headers

The required headers are

- `type` to identify the file format version (its value should be `PointsFile v.1.0`),
- `width` and `height` representing the frame size,
- `uid` with an integer identifier representing the data. This can be an arbitrary integer, and from our personal experience, it sometimes proves useful to avoid accidentally mixing data and this is why it is mandatory, although you can safely set it to `0` if you do not plan to use it.

C.2 USER REFERENCE

C.2.1 ASTRE (`astre-noholes` and `astre-holes`)

`astre-noholes`
`astre-holes`

The `astre-noholes` and `astre-holes` programs greedily detect trajectories in a points description file using the a-contrario framework. The `astre-noholes` can be used when the trajectories do not contain holes, and its detection criterion is optimized for this case. In general, `astre-noholes` is faster and more memory-efficient than `astre-holes`.

The basic usage to detect trajectories of $\log_{10}(\text{NFA})$ less than 0 is:

```
$ astre-noholes <in> <out>
$ astre-holes <in> <out>
```

The options are:

`-epsilon <e>` (or `-e <e>`)	set the maximal $\log_{10}(\text{NFA})$ for the trajectories detection (default: 0.0)

`-max-hole-length <h>` (or `-h <h>`)	set the maximal size of a hole when using `astre-noholes` (default: any length). This can be used to lower the computational and memory costs, at the expense of not considering all the possible trajectories.

ASTRE adds a column containing trajectory identifiers, or `-1` if a point does not belong to a detected trajectory. It also adds headers of the form `traj:<id>:lNFA = <lNFA>` that describe the $\log_{10}(\text{NFA})$ of each trajectory.

Using the programs with the option `-tag-NFA` only tags each trajectory with its NFA and exits. This can be useful if you have detected trajectories using another program, and want to filter out trajectories above a certain NFA.

For long computations, you might want ASTRE to periodically save the partial detections, to make detection backups or to monitor detected trajectories. This is possible using the `-save-partial <filename>` option. Restarting a computation is done using the `-restart` option and using the partial detections file as input:

```
$ astre-holes --save-partial partial_tjs pts tjs
...[interrupt computations]
$ astre-holes --save-partial partial_tjs --restart partial_tjs tjs
```

Finally, the `auto-crop` option is not thoroughly tested but might help detecting trajectories when image sequences have some points in the center of the images and a lot of empty space around, by automatically cropping the empty space, which changes the implicit scale of the images, and hence the NFA.

C.2.2 Recall and precision computation (`tstats`)

`tstats`

The `tstats` program outputs the recall and precision defined in terms of real, correct and found links, as well as the number of detected trajectories.

The basic usage is:

```
$ tstats [<ground truth file>] <detected trajectories file>
```

If the ground truth file is not specified, this assumes that the detected trajectories file contains both the ground truth and the trajectories detected by the algorithm. The ground truth and the detected trajectories file must have the same `uid` header. By default, the ground truth file should have the trajectories identifiers in the fourth column and the detected trajectories file should have them in the last column.

A link is two successive (possibly separated by a hole) detected points in a trajectory. A link is real if it belongs to the ground truth, it is found if it belongs to the detected trajectories, and it is correct if it belongs to both, and we define

$$\begin{aligned}\text{precision} &= \frac{\text{\# of correct links found}}{\text{\# of links found}},\\ \text{and recall} &= \frac{\text{\# of correct links found}}{\text{\# of actual links}},\end{aligned}$$

The recall corresponds to the proportion of real points that have been detected, while the precision corresponds to the proportion of detected points that are correct, that is, it inversely relates to the number of false detections.

The options are:

`-real-traj <r>` (or `-r <r>`)	the 0-based index of the column containing the real trajectories in the ground truth points description file (default: 3)
`-found-traj <f>` (or `-f <f>`)	the 0-based index of the column containing the real trajectories in the detected trajectories points description file (default: -1, the last column)

C.2.3 PSMG Point Set Motion Generator (`tpsmg`)

`tpsmg`

The `tpsmg` program emulates the Point Set Motion Generator, that randomly draws some points in the image frame and displace them by updating their acceleration from frame to frame with a Gaussian variable.

The basic usage to generate `n` trajectories spanning `K` frames in the file `out` is:

```
$ tpsmg <K> <n> <out>
```

The options are:

`-N <N>`	add `N` noise points to each image (the noise points cannot share the location of a trajectory point)
`-r`	the number of noise points added to each image is not constant, but rather a number drawn uniformly between `0` and `N`
`-F`	do not force trajectories to stay inside the frames
`-w <w> -h <h>`	set the image width and height
`-a <a>`	set the variance of the speed amplitude update (default: `0.2`)
`-o <o>`	set the variance of the speed orientation update (default: `0.2`)
`-v <v>`	set the mean initial speed of the trajectories (default: `5.0`)
`-V <V>`	set the variance of the initial speed of the trajectories (default: `0.5`)

trajectory generation

The trajectories are generated in the following way:

1. the first point of each of the `n` trajectories is drawn uniformly in the first frame of the sequence,
2. each trajectory is generated in turn:
   a) we draw a normal variable of mean `v` and variance `V` that defines the initial speed amplitude, and a variable drawn in $[0, 2\pi]$ that defines the initial speed orientation,
   b) we update the trajectory position according to the speed,
   c) we add the point to the trajectory, unless it leaves the frame or it is on the location of a previously generated trajectory, in which case we regenerate the whole trajectory,
   d) we draw a normal variable of mean 0 and variance `a` representing the speed amplitude update, and a normal variable of mean 0 and variance `o` representing the speed orientation update,
   e) we update the speed amplitude and orientation
3. we add `N` random points in each frame such that they do not share position with a trajectory point (if the `-r` option is set, we add a number of noise points drawn uniformly between 0 and `N`,

4. we shuffle the position of all the points in each frame in the points description file.

If the -F option has been set, the trajectories are not forced to stay inside the frame, and as they leave the image they are ended and a new trajectory is started in the next frame by choosing a random point uniformly on the border of the image frame and iterating the trajectory generation process. Note in this case that if a trajectory has less than 3 points in the image frame, we regenerate it, and if a trajectory leaves the image frame less than 3 frames before the end of the sequence, no new trajectory is generated, since it could not comply with the above 3-frames constraint.

Note that for some parameters, it is possible that the tpsmg does not terminate, in particular if the amplitude of the trajectory acceleration is too high, it is possible that no trajectory fits in the frame, and the program keeps trying to regenerate them.

The tpsmg program also outputs metadata indicating the global maximal speed and acceleration of the generated trajectories:

```
$ tpsmg 20 5 pts
[MD] { 'max_speed': 5.83095, 'max_accel': 2.82843 }
```

C.2.4 Point set crippling (tcripple)

tcripple

The tcripple program remove a certain amount of points from the trajectories contained in a points description file to simulate missing detections.

Note that the number of points removed is generally not a fixed proportion of the input points, but rather, each point from a trajectory is removed with a certain probability r.

Note also that tcripple only affects point on a trajectory, that is, points having a trajectory tag different from -1 (and thus, points description files need to have a trajectory identifier column when using tcripple).

The basic usage to remove points in trajectories with probability r is:

```
$ tcripple -r <r> <in> <out>
```

The options are:

-r <r>	the probability that a point in a trajectory is removed
-traj-col <t> (or <tt>-t <t>)	the 0-based index of the column containing the trajectories (default: -1, the last column)

C.2.5 Viewer (tview.py)

tview.py

If you have installed the Python libraries, you can use the tview.py program to view points description files and the point trajectories.

The basic usage is

```
$ tview.py [-t] <points description file>
```

The option -t indicates that we want to load the trajectories contained in the points description file, where the trajectories are defined by the last column of the points description file. Points having the same identifier as the last column are in the same trajectory, and points having the special identifier -1 do not belong to any trajectory.

The options are:

`-trajectory-column <n>`	the column (index is 0-based) describing the trajectories (default is -1, the last column)
`-show-trajectories` (or `-t`)	load and show trajectories using the trajectory identifiers in the column `trajectory-column`

and the commands are:

`f/b`	go forward or backward in the sequence
`q`	quit the program
`+/-/=`	zoom in/out/default
`t`	toggle trajectory drawing (when trajectories have been loaded)
`m`	set the current frame as the first frame from which to draw trajectories
`h`	highlight the points that belong to a trajectory in red
`x`	randomize the trajectory colors

C.2.6 Naive ASTRE implementation in Python (`naive_astre.py`)

`naive_astre.py`

ASTRE comes with a naive implementation in the Python language for clarity, that should however not be used with large datasets, as it is slow and requires a lot of memory.

The basic usage is

```
$ naive_astre.py <in> <out>
```

and the options are

`-solver [noholes|holes]`	the solver (without holes or with holes)
`-eps <e>` (or `-e <e>`)	the maximal value of $\log_{10}(\text{NFA})$

Bibliography

Ali, Anjum and J. K. Aggarwal

2001 "Segmentation and Recognition of Continuous Human Activity", *Detection and Recognition of Events in Video, IEEE Workshop on*, p. 28. (Cited on p. 48.)

Ali, Rehan, Mark Gooding, Tünde Szilágyi, Borivoj Vojnovic, Martin Christlieb, and Michael Brady

2011 "Automatic segmentation of adherent biological cell boundaries and nuclei from brightfield microscopy images", *Machine Vision and Applications*, 10.1007/s00138-011-0337-9, pp. 1–15, ISSN: 0932-8092.

Althoff, K., J. Degerman, and T. Gustavsson

2005 "Combined Segmentation and Tracking of Neural Stem-Cells", in *Scandinavian Conference on Image Analysis*, pp. 282–291.

Ashdown, Mark, Kenji Oka, and Yoichi Sato

2005 "Combining head tracking and mouse input for a GUI on multiple monitors", in *Extended Abstracts Proceedings of the 2005 Conference on Human Factors in Computing Systems, CHI 2005, Portland, Oregon, USA, April 2-7, 2005*, ed. by Gerrit C. van der Veer and Carolyn Gale, Association for Computing Machinery, pp. 1188–1191, ISBN: 1-59593-002-7. (Cited on p. 48.)

Avidan, Shai

2001 "Support Vector Tracking", in *2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2001), with CD-ROM, 8-14 December 2001, Kauai, HI, USA*, IEEE Computer Society, pp. 184–191, ISBN: 0-7695-1272-0. (Cited on p. 48.)

Bar-Shalom, Y.

2006 "On hierarchical tracking for the real world", *IEEE Transactions on Aerospace and Electronic Systems*, 42, 3, pp. 846–850, ISSN: 0018-9251.

Bar-Shalom, Y., T. Fortmann, and M. Scheffe

1983 "Sonar tracking of multiple targets using joint probabilistic data association", *IEEE Journal of Oceanic Engineering*, 8, 3, pp. 173–184. (Cited on pp. 48, 49.)

Bar-Shalom, Y. and T. Kirubajan

2004 "Probabilistic Data Association Techniques for Target Tracking in Clutter", *Proceedings of the IEEE*, 92, 3, pp. 536–557, ISSN: 0018-9219.

Berclaz, J., F. Fleuret, and P. Fua

2009 "Multiple Object Tracking using Flow Linear Programming", in *12th IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (Winter-PETS)*. (Cited on pp. 50, 168.)

Berclaz, J., E. Turetken, F. Fleuret, and P. Fua

2011 "Multiple Object Tracking using K-Shortest Paths Optimization", *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 33, 9, pp. 1806–1819. (Cited on p. 50.)

Bertalmío, Marcelo, Guillermo Sapiro, and Gregory Randall

2000 "Morphing Active Contours", *IEEE Trans. Pattern Anal. Mach. Intell.*, 22, 7, pp. 733–737. (Cited on p. 48.)

Betke, M., DE Hirsh, A. Bagchi, NI Hristov, NC Makris, and TH Kunz

2007 "Tracking large variable numbers of objects in clutter", in *2007 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, pp. 1–8. (Cited on p. 6.)

Beucher, S. and F. Meyer

1992 *The morphological approach to segmentation: the watershed transformation*, ed. by E. Dougherty, Marcel Dekker, New York, NY, USA. (Cited on p. 19.)

Bhaskar, Harish and Sameer Singh

2007 "Live cell imaging: a computational perspective", *Journal of Real-Time Image Processing*, 1 [3], 10.1007/s11554-007-0022-4, pp. 195–212, ISSN: 1861-8200. (Cited on p. 15.)

Black, Michael J. and Allan D. Jepson

1998 "EigenTracking: Robust Matching and Tracking of Articulated Objects Using a View-Based Representation", *International Journal of Computer Vision*, 26, 1, pp. 63–84. (Cited on p. 48.)

Blake, Andrew and M. Isard

1998 *Active Contours: The Application of Techniques from Graphics,Vision,Control Theory and Statistics to Visual Tracking of Shapes in Motion*, 1st, Springer-Verlag New York, Inc., Secaucus, NJ, USA, ISBN: 3540762175. (Cited on pp. 19, 48.)

Boulanger, Jérôme, Charles Kervrann, Patrick Bouthemy, Peter Elbau, Jean-Baptiste Sibarita, and Jean Salamero

2010 "Patch-Based Nonlocal Functional for Denoising Fluorescence Microscopy Image Sequences", *IEEE Trans. Med. Imaging*, 29, 2, pp. 442–454. (Cited on p. 22.)

Broida, T.J. and R. Chellappa

1986 "Estimation of object motion parameters from noisy images", *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 1, pp. 90–99.

Butail, Sachit and Derek A. Paley

2011 "Three-dimensional reconstruction of the fast-start swimming kinematics of densely schooling fish", *Journal of The Royal Society Interface* [June], DOI: 10.1098/rsif.2011.0113.

Canny, J.

1987 "A computational approach to edge detection", *Readings in computer vision: issues, problems, principles, and paradigms*, 184, 87-116, p. 86.

Carpenter, Anne, Thouis Jones, Michael Lamprecht, Colin Clarke, In Kang, Ola Friman, David Guertin, Joo Chang, Robert Lindquist, Jason Moffat, Polina Golland, and David Sabatini

2006 "CellProfiler: image analysis software for identifying and quantifying cell phenotypes", *Genome Biology*, 7, 10, R100, ISSN: 1465-6906, DOI: 10.1186/gb-2006-7-10-r100.

Caselles, Vicent, Ron Kimmel, and Guillermo Sapiro

1995 "Geodesic Active Contours", in *International Conference on Computer Vision*, pp. 694–699.

Comaniciu, Dorin, Visvanathan Ramesh, and Peter Meer

2003 "Kernel-Based Object Tracking", *IEEE Trans. Pattern Anal. Mach. Intell.*, 25, 5, pp. 564–575. (Cited on p. 48.)

Cornelissen, T, M Elsing, I Gavrilenko, W Liebig, E Moyse, and A Salzburger

2008 "The new ATLAS track reconstruction (NEWT)", *Journal of Physics: Conference Series*, 119, 3. (Cited on p. 48.)

Cortes, Corinna and Vladimir Vapnik

1995 "Support-vector networks", *Machine Learning*, 20 [3], 10.1007/BF00994018, pp. 273–297, ISSN: 0885-6125.

Cox, Ingemar J.

1993 "A review of statistical data association techniques for motion correspondence", *International Journal of Computer Vision*, 10 [1], pp. 53–66, ISSN: 0920-5691.

Dean, Jeffrey and Sanjay Ghemawat

2008 "MapReduce: simplified data processing on large clusters", *Commun. ACM*, 51 [1], pp. 107–113, ISSN: 0001-0782, DOI: http://doi.acm.org/10.1145/1327452.1327492. (Cited on p. 166.)

Debeir, Olivier, Philippe Van Ham, Robert Kiss, and Christine Decaestecker

2005 "Tracking of migrating cells under phase-contrast video microscopy with combined mean-shift processes", *IEEE Trans. Med. Imaging*, 24, 6, pp. 697–711. (Cited on p. 18.)

Delgado-Gonzalo, R., N. Dénervaud, S. Maerkl, and M. Unser

2010 "Multi-target tracking of packed yeast cells", in *Biomedical Imaging: From Nano to Macro, 2010 IEEE International Symposium on*, IEEE, pp. 544–547. (Cited on pp. 5, 20.)

Dellaert, Frank, Steven M. Seitz, Charles E. Thorpe, and Sebastian Thrun

2003 "EM, MCMC, and Chain Flipping for Structure from Motion with Unknown Correspondence", *Machine Learning*, 50, 1-2, pp. 45–71. (Cited on pp. 122–124.)

Desolneux, Agnès, Lionel Moisan, and Jean-Michel Morel

2000 "Meaningful Alignments", *International Journal of Computer Vision*, 40, 1, pp. 7–23, ISSN: 0920-5691. (Cited on p. 51.)

2001 "Edge Detection by Helmholtz Principle", *Journal of Mathematical Imaging and Vision*, 14, 3, pp. 271–284. (Cited on p. 51.)

2003 "Maximal Meaningful Events and Applications to Image Analysis", *The Annals of Statistics*, 31, 6, pp. 1822–1851. (Cited on p. 8.)

2008 *From Gestalt Theory to Image Analysis: A Probabilistic Approach (Interdisciplinary Applied Mathematics)*, Springer, ISBN: 0387726357. (Cited on pp. 50, 51, 78, 84.)

Dijkstra, E. W.

1959 "A note on two problems in connexion with graphs", *Numerische Mathematik*, 1 [1], pp. 269–271, ISSN: 0029-599X. (Cited on p. 28.)

Edwards, Gareth J., Christopher J. Taylor, and Timothy F. Cootes

1998 "Interpreting Face Images Using Active Appearance Models", in *3rd International Conference on Face & Gesture Recognition (FG '98), April 14-16, 1998, Nara, Japan*, IEEE Computer Society, pp. 300–305. (Cited on p. 48.)

Elgammal, Ahmed M., David Harwood, and Larry S. Davis

2000 "Non-parametric Model for Background Subtraction", in *Computer Vision - ECCV 2000, 6th European Conference on Computer Vision, Dublin, Ireland, June 26 - July 1, 2000, Proceedings, Part II*, ed. by David Vernon, vol. 1843, Lecture Notes in Computer Science, Springer, pp. 751–767, ISBN: 3-540-67686-4.

Fernàndez, G., M. Kunt, and J.P. Zrÿd

1995 "A new plant cell image segmentation algorithm", *Proc. of the 8th Int. Conference on Image Analysis and Processing*, pp. 229–234. (Cited on p. 19.)

Fieguth, Paul W. and Demetri Terzopoulos

1997 "Color-Based Tracking of Heads and Other Mobile Objects at Video Frame Rates", in *1997 Conference on Computer Vision and Pattern Recognition (CVPR '97), June 17-19, 1997, San Juan, Puerto Rico*, IEEE Computer Society, pp. 21–27. (Cited on p. 48.)

Fleuret, Francois, Jerome Berclaz, Richard Lengagne, and Pascal Fua

2008 "Multicamera People Tracking with a Probabilistic Occupancy Map", *IEEE Trans. Pattern Anal. Mach. Intell.*, 30 [2], pp. 267–282, ISSN: 0162-8828, DOI: 10.1109/TPAMI.2007.1174. (Cited on pp. 3, 7, 49.)

Garfinkel, Robert S.

1971 "An Improved Algorithm for the Bottleneck Assignment Problem", *Operations Research*, 19, 7, pp. 1747–1751, DOI: 10.2307/169194. (Cited on pp. 99, 100.)

Godinez, WJ, M. Lampe, R. Eils, B. Muller, and K. Rohr

2011 "Tracking multiple particles in fluorescence microscopy images via probabilistic data association", in *Biomedical Imaging: From Nano to Macro, 2011 IEEE International Symposium on*, IEEE, pp. 1925–1928. (Cited on p. 6.)

Gor, V., M. Elowitz, T. Bacarian, and E. Mjolsness

2005 "Tracking cell signals in fluorescent images", in *Proc. CVPR'05*, vol. 03, p. 142.

Gordon, N.J., D.J. Salmond, and A.F.M. Smith

1993 "Novel approach to nonlinear/non-Gaussian Bayesian state estimation", in *Radar and Signal Processing, IEE Proceedings F*, vol. 140, 2, IET, pp. 107–113.

Gough, Nancy R. and Michael B. Yaffe

2011 "Focus Issue: Conquering the Data Mountain", *Sci. Signal.*, 4, 160, eg2, DOI: 10.1126/scisignal.2001871, eprint: http://stke.sciencemag.org/cgi/reprint/sigtrans;4/160/eg2.pdf. (Cited on p. 15.)

Grosjean, Bénédicte and Lionel Moisan

2009 "A-contrario Detectability of Spots in Textured Backgrounds", *Journal of Mathematical Imaging and Vision*, 33, 3, pp. 313–337. (Cited on pp. 51, 52.)

Gross, O.

1959 *The Bottleneck Assignment Problem*, The Rand Corporation, Santa Monica, CA, USA. (Cited on p. 99.)

Gui, L.C. and W. Merzkirch

1996 "A method of tracking ensembles of particle images", *Experiments in Fluids*, 21, 6 [Nov.], pp. 465–468, ISSN: 0723-4864. (Cited on p. 47.)

Han, Ji, Toby Breckon, David Randell, and Gabriel Landini

2010 "The application of support vector machine classification to detect cell nuclei for automated microscopy", *Machine Vision and Applications*, 10.1007/s00138-010-0275-y, pp. 1–10, ISSN: 0932-8092.

Hand, A.J., T. Sun, D.C. Barber, D.R. Hose, and S. MacNeil

2009 "Automated tracking of migrating cells in phase-contrast video microscopy sequences using image registration", *Journal of Microscopy*, 234, 1, pp. 62–79, ISSN: 1365-2818, DOI: `10.1111/j.1365-2818.2009.03144.x`. (Cited on pp. 2, 15.)

Haralick, Robert M.

1984 "Digital Step Edges from Zero Crossing of Second Directional Derivatives", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6, pp. 58–68, DOI: `10.1109/TPAMI.1984.4767475`.

Harris, C. and M. Stephens

1988 "A combined corner and edge detector", in *Alvey vision conference*, vol. 15, Manchester, UK, p. 50.

Hastings, W. K.

1970 "Monte Carlo sampling methods using Markov chains and their applications", *Biometrika*, 57, 1 [Apr.], pp. 97–109, DOI: `10.1093/biomet/57.1.97`. (Cited on p. 121.)

Hey, Tony, Stewart Tansley, and Kristin Tolle

2009 (ed.)*The Fourth Paradigm: Data-Intensive Sciencetific Discovery*, Microsoft Research, ISBN: 9780982544204. (Cited on pp. 165, 166.)

Hilbert, Martin and Priscila López

2011 "The World's Technological Capacity to Store, Communicate, and Compute Information", *Science*, 332, 6025 [Apr.], pp. 60–65, ISSN: 1095-9203, DOI: `10.1126/science.1200970`. (Cited on pp. 15, 165.)

Horn, Berthold K. P. and Brian G. Schunck

1993 ""Determining optical flow": A Retrospective", *Artif. Intell.*, 59, 1-2, pp. 81–87.

Huang, Timothy and Stuart Russell

1998 "Object identification: a Bayesian analysis with application to traffic surveillance", *Artificial Intelligence*, 103, 1-2, Artificial Intelligence 40 years later, pp. 77–93, ISSN: 0004-3702, DOI: `DOI:10.1016/S0004-3702(98)00067-8`.

Huttenlocher, D., J.J Noh, and W.J. Rucklidge

1993 "Tracking Nonrigid Objects in Complex Scenes", in *Proceedings of ICCV*, pp. 93–101. (Cited on p. 48.)

Isard, Michael and Andrew Blake

1998 "CONDENSATION - Conditional Density Propagation for Visual Tracking", *International Journal of Computer Vision*, 29, 1, pp. 5–28.

Jerrum, Mark, Alistair Sinclair, and Eric Vigoda

2004 "A polynomial-time approximation algorithm for the permanent of a matrix with nonnegative entries", *J. ACM*, 51 [4], pp. 671–697, ISSN: 0004-5411, DOI: http://doi.acm.org/10.1145/1008731.1008738. (Cited on p. 119.)

Jiang, Hao, Sidney Fels, and James J. Little

2007 "A Linear Programming Approach for Multiple Object Tracking", *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 0, pp. 1–8. (Cited on pp. 50, 168.)

Jonker, R. and A. Volgenant

1987 "A shortest augmenting path algorithm for dense and sparse linear assignment problems", *Computing*, 38 [4], pp. 325–340, ISSN: 0010-485X, DOI: 10.1007/BF02278710. (Cited on p. 100.)

Kang, Jinman, Isaac Cohen, and Gérard G. Medioni

2004 "Object Reacquisition Using Invariant Appearance Model", in *International Conference on Pattern Recognition*, vol. 4, pp. 759–762. (Cited on p. 48.)

Kanizsa, Gaetano

1980 *Grammatica del vedere*, Il Mulino, Bologna. (Cited on p. 51.)

Karp, R.M.

1972 "Reducibility among Combinatorial Problems", in *Complexity of Computer Computations*, ed. by R.E. Miller and J.W. Thatcher, Plenum Press, New York, pp. 85–103. (Cited on p. 7.)

Kass, Michael, Andrew Witkin, and Demetri Terzopoulos

1988 "Snakes: Active contour models", *International Journal of Computer Vision*, 1 [4], 10.1007/BF00133570, pp. 321–331, ISSN: 0920-5691. (Cited on p. 19.)

Khan, Zia, Tucker Balch, and Frank Dellaert

2005 "MCMC-Based Particle Filtering for Tracking a Variable Number of Interacting Targets", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27, pp. 1805–1918, ISSN: 0162-8828.

Koller, Dieter, Joseph Weber, and Jitendra Malik

1994 "Robust Multiple Car Tracking with Occlusion Reasoning", in *Computer Vision - ECCV'94, Third European Conference on Computer Vision, Stockholm, Sweden, May 2-6, 1994, Proceedings, Volume I*, ed. by Jan-Olof Eklundh, vol. 800, Lecture Notes in Computer Science, Springer, pp. 189–196, ISBN: 3-540-57956-7. (Cited on p. 48.)

Kuhn, H W

1955 "The Hungarian method for the assignment problem", *Naval Research Logistic Quarterly*, 2, pp. 83–97. (Cited on pp. 7, 35, 98–100.)

Lee, Chan-Su and Ahmed Elgammal

2010 "Style adaptive contour tracking of human gait using explicit manifold models", *Machine Vision and Applications*, 10.1007/s00138-010-0303-y, pp. 1–18, ISSN: 0932-8092.

Li, Kang and Takeo Kanade

2009 "Nonnegative Mixed-Norm Preconditioning for Microscopy Image Segmentation", in *Information Processing in Medical Imaging, 21st International Conference, IPMI 2009, Williamsburg, VA, USA, July 5-10, 2009. Proceedings*, ed. by Jerry L. Prince, Dzung L. Pham, and Kyle J. Myers, vol. 5636, Lecture Notes in Computer Science, Springer, pp. 362–373, ISBN: 978-3-642-02497-9. (Cited on p. 18.)

Li, Kang, E. Miller, L. Weiss, P. Campbell, and T. Kanade

2006 "Online Tracking of Migrating and Proliferating Cells Imaged with Phase-Contrast Microscopy", in *Proc. of CVPRW'06*, p. 65.

Li, Kang, Eric D. Miller, Mei Chen, Takeo Kanade, Lee E. Weiss, and Phil G. Campbell

2008 "Cell population tracking and lineage construction with spatiotemporal context", *Medical Image Analysis*, 12, 5, Special issue on the 10th international conference on medical imaging and computer assisted intervention - MICCAI 2007, pp. 546–566, ISSN: 1361-8415, DOI: DOI:10.1016/j.media.2008.06.001. (Cited on pp. 5, 19, 21.)

Liu, Yiu T. and Paul K. Warme

1977 "Computerized evaluation of sperm cell motility", *Computers and Biomedical Research*, 10, 2, pp. 127–138, ISSN: 0010-4809, DOI: DOI:10.1016/0010-4809(77)90030-1. (Cited on pp. 4, 19.)

Lowe, David G.

2004 "Distinctive Image Features from Scale-Invariant Keypoints", *International Journal of Computer Vision*, 60 [2], pp. 91–110, ISSN: 0920-5691.

Markoff, John

2010 "Google Cars Drive Themselves, in Traffic", *New York Times website* [Oct.], http://www.nytimes.com/2010/10/10/science/10google.html.

Meijering, Erik, Oleh Dzyubachyk, Ihor Smal, and Wiggert A. van Cappellen

2009 "Tracking in cell and developmental biology", *Seminars in Cell & Developmental Biology*, 20, 8, Imaging in Cell and Developmental Biology; Planar Cell Polarity, pp. 894 –902, ISSN: 1084-9521, DOI: DOI:10.1016/j.semcdb.2009.07.004. (Cited on pp. 2, 15.)

Metropolis, Nicholas, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller

1953 "Equation of State Calculations by Fast Computing Machines", *The Journal of Chemical Physics*, 21, 6, pp. 1087–1092, DOI: 10.1063/1.1699114. (Cited on p. 121.)

Moeslund, Thomas B. and Erik Granum

2001 "A Survey of Computer Vision-Based Human Motion Capture", *Computer Vision and Image Understanding*, 81, 3, pp. 231–268.

Moeslund, Thomas B., Moritz Störring, and Erik Granum

2001 "A Natural Interface to a Virtual Environment through Computer Vision-Estimated Pointing Gestures", in *Gesture and Sign Languages in Human-Computer Interaction, International Gesture Workshop, GW 2001, London, UK, April 18-20, 2001, Revised Papers*, ed. by Ipke Wachsmuth and Timo Sowa, vol. 2298, Lecture Notes in Computer Science, Springer, pp. 59–63, ISBN: 3-540-43678-2.

Moisan, Lionel and Bérenger Stival

2004 "A probabilitic criterion to detect rigid point matches between two images and estimate the fundamental matrix", *International Journal of Computer Vision*, 57, 3, pp. 201–218. (Cited on p. 51.)

Monasse, P. and F. Guichard

2000 "Fast computation of a constrast-invariant image representation", *IEEE Trans. on Image Processing*, pp. 860–872. (Cited on p. 24.)

Munkres, James

1957 "Algorithms for the Assignment and Transportation Problems", *Journal of the Society for Industrial and Applied Mathematics*, 5, 1, pp. 32–38. (Cited on p. 74.)

Murty, Katta G.

1968 "An Algorithm for Ranking all the Assignments in Order of Increasing Cost", *Operations Research*, 16, 3, pp. 682–687. (Cited on p. 75.)

Musé, Pablo, Frédéric Sur, Frédéric Cao, Yann Gousseau, and Jean-Michel Morel

2004 *Accurate estimates of false alarm number in shape recognition*, Research Report RR-5086, INRIA.

Otsu, N.

1979 "A threshold selection method from gray level histograms", *IEEE Trans. Systems, Man and Cybernetics*, 9 [Mar.], pp. 62–66. (Cited on p. 18.)

Padfield, Dirk R., Jens Rittscher, and Badrinath Roysam

2008 "Spatio-temporal cell segmentation and tracking for automated screening", in *Proceedings of the 2008 IEEE International Symposium on Biomedical Imaging: From Nano to Macro, Paris, France, May 14-17, 2008*, IEEE, pp. 376–379. (Cited on pp. 5, 19.)

Papageorgiou, Constantine P., Michael Oren, and Tomaso Poggio

1998 "A General Framework for Object Detection", *Computer Vision, IEEE International Conference on*, 0, p. 555.

Paragios, Nikos and Rachid Deriche

2000 "Geodesic Active Contours and Level Sets for the Detection and Tracking of Moving Objects", *IEEE Trans. Pattern Anal. Mach. Intell.*, 22, 3, pp. 266–280. (Cited on p. 48.)

Pighin, Frederic H., Richard Szeliski, and David Salesin

1999 "Resynthesizing Facial Animation through 3D Model-based Tracking", in *International Conference on Computer Vision*, pp. 143–150. (Cited on p. 48.)

Rangarajan, Krishnan and Mubarak Shah

1991 "Establishing motion correspondence", *Computer Vision and Pattern Recognition, 1991. Proceedings CVPR '91., IEEE Computer Society Conference on*, pp. 103–108. (Cited on pp. 49, 98.)

Rasmussen, C. and G.D. Hager
2001 "Probabilistic data association methods for tracking complex visual objects", *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23, 6, pp. 560–576.

Reid, D.
1979 "An algorithm for tracking multiple targets", *Automatic Control, IEEE Transactions on*, 24, 6, pp. 843–854. (Cited on pp. 6, 7, 48.)

Robin, Amandine, Lionel Moisan, and Sylvie Le Hégarat-Mascle
2010 "An A-Contrario Approach for Subpixel Change Detection in Satellite Imagery", *IEEE Trans. Pattern Anal. Mach. Intell.*, 32, 11, pp. 1977–1993. (Cited on p. 51.)

Roerdink, J. and A. Meijster
2000 "The watershed transform : definitions, algorithms and parallelization strategies", *Fundamenta Informatica*, 41, 1-2, pp. 187–228.

Ronfard, Rémi
1994 "Region-based strategies for active contour models", *International Journal of Computer Vision*, 13, 2, pp. 229–251. (Cited on p. 48.)

Rudin, L. and S. Osher
1994 "Total variation based image restoration with free local constraints", *Proc. IEEE Int. Conf. on Image Processing*. (Cited on p. 22.)

Rudin, L., S. Osher, and E. Fatemi
1992 "Nonlinear total variation based noise removal algorithms", *Physica D*. (Cited on p. 22.)

Salari, V. and I.K. Sethi
1990 "Feature point correspondence in the presence of occlusion", *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 12, 1, pp. 87–91. (Cited on pp. 49, 98, 100.)

Sato, Koichi and Jake K. Aggarwal
2004 "Temporal spatio-velocity transform and its application to tracking and interaction", *Computer Vision and Image Understanding*, 96, 2, pp. 100–128. (Cited on p. 48.)

Sbalzarini, I.F. and P. Koumoutsakos
2005 "Feature point tracking and trajectory analysis for video imaging in cell biology.", *Journal of Structural Biology*, 151, 2, pp. 182–95, ISSN: 10478477. (Cited on p. 48.)

Science
2011 "Challenges and Opportunities", *Science*, 331, 6018, pp. 692–693, DOI: `10.1126/science.331.6018.692`, eprint: `http://www.sciencemag.org/content/331/6018/692.full.pdf`. (Cited on pp. 15, 166.)

Serby, David, Esther Koller-Meier, and Luc J. Van Gool
2004 "Probabilistic Object Tracking Using Multiple Features", in *International Conference on Pattern Recognition*, vol. 2, pp. 184–187. (Cited on p. 48.)

Sethi, I. K. and R. Jain
1987 "Finding trajectories of feature points in a monocular image sequence", *IEEE Trans. Pattern Anal. Mach. Intell.*, 9, 1 [Jan.], pp. 56–73, ISSN: 0162-8828. (Cited on p. 49.)

Sethian, J. A.

1996 "A fast marching level set method for monotonically advancing fronts", *Proceedings of the National Academy of Sciences*, 93, 4 [Feb.], pp. 1591–1595. (Cited on p. 28.)

Shafique, Khurram and Mubarak Shah

2003 "A Non-Iterative Greedy Algorithm for Multi-frame Point Correspondence", in *9th IEEE International Conference on Computer Vision (ICCV 2003), 14-17 October 2003, Nice, France*, IEEE Computer Society, pp. 110–115, ISBN: 0-7695-1950-4. (Cited on p. 48.)

Shen, Hailin, Glyn Nelson, Stephnie Kennedy, David Nelson, James Johnson, David Spiller, Michael R.H. White, and Douglas B. Kell

2006 "Automatic tracking of biological cells and compartments using particle filters and active contours", *Chemometrics and Intelligent Laboratory Systems*, 82, 1-2, Selected Papers from the International Conference on Chemometrics and Bioinformatics in Asia - CCBA 2004, pp. 276–282, ISSN: 0169-7439, DOI: DOI:10.1016/j.chemolab.2005.07.007. (Cited on p. 19.)

Shi, Jianbo and Carlo Tomasi

1994 "Good features to track", in *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94., 1994 IEEE Computer Society Conference on*, Seattle, WA, USA, pp. 593–600, DOI: 10.1109/CVPR.1994.323794, http://dx.doi.org/10.1109/CVPR.1994.323794.

Smal, Ihor, Wiro J. Niessen, and Erik H. W. Meijering

2008 "A new detection scheme for multiple object tracking in fluorescence microscopy by joint probabilistic data association filtering", in *Proceedings of the 2008 IEEE International Symposium on Biomedical Imaging: From Nano to Macro, Paris, France, May 14-17, 2008*, IEEE, pp. 264–267. (Cited on p. 48.)

Smith, Kevin and Alan Carleton Vincent Lepetit

2008 "General constraints for batch multiple-target tracking applied to largescale videomicroscopy", in *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2008*, pp. 1–8. (Cited on pp. 5, 20.)

Stephens, David J. and Victoria J. Allan

2003 "Light Microscopy Techniques for Live Cell Imaging", *Science*, 300, 5616 [Apr.], pp. 82–86, ISSN: 1095-9203, DOI: 10.1126/science.1082160. (Cited on p. 15.)

Stewart, Eric J, Richard Madden, Gregory Paul, and François Taddei

2005 "Aging and Death in an Organism That Reproduces by Morphologically Symmetric Division", *PLoS Biol*, 3, 2 [Feb.], e45, DOI: 10.1371/journal.pbio.0030045. (Cited on p. 16.)

Streit, Roy L. and Tod E. Luginbuhl

1994 "Maximum likelihood method for probabilistic multihypothesis tracking", in *Signal and Data Processing of Small Targets 1994*, ed. by Oliver E. Drummond, vol. 2235, SPIE, Orlando, FL, USA, pp. 394–405. (Cited on p. 48.)

Tang, C. and E. Bengtsson

2005 "Segmentation and Tracking of Neural Stem Cell", in *International Conference on Intelligent Computing (2)*, pp. 851–859.

Tao, Hai, Harpreet S. Sawhney, and Rakesh Kumar

2002 "Object Tracking with Bayesian Estimation of Dynamic Layer Representations", *IEEE Trans. Pattern Anal. Mach. Intell.*, 24, 1, pp. 75–89. (Cited on p. 48.)

Tomaževič, D., B. Likar, and F. Pernuš

2002 "Comparative evaluation of retrospective shading correction methods", *Journal of Microscopy*, 208, 3, pp. 212–223, ISSN: 1365-2818, DOI: 10.1046/j.1365-2818.2002.01079.x. (Cited on p. 18.)

Vachier, Corinne and Fernand Meyer

2005 "The Viscous Watershed Transform", *Journal of Mathematical Imaging and Vision*, 22, 2-3, pp. 251–267. (Cited on p. 26.)

Veenman, Cor J., Emile A. Hendriks, and Marcel J. T. Reinders

1998 "A Fast and Robust Point Tracking Algorithm", in *International Conference on Image Processing*, vol. 3, pp. 653–657.

Veenman, Cor J., Marcel J. T. Reinders, and Eric Backer

2001 "Resolving Motion Correspondence for Densely Moving Points", *IEEE Trans. Pattern Anal. Mach. Intell.*, 23, 1, pp. 54–72. (Cited on p. 155.)

2003a "Establishing motion correspondence using extended temporal scope", *Artif. Intell.*, 145, 1-2, pp. 227–243. (Cited on p. 76.)

2003b "Motion tracking as a constrained optimization problem", *Pattern Recognition*, 36, 9, pp. 2049–2067. (Cited on pp. 7, 48–50, 73, 75, 76, 155.)

Veit, Thomas, Frédéric Cao, and Patrick Bouthemy

2007 "Space-time A Contrario Clustering for Detecting Coherent Motions", in *2007 IEEE International Conference on Robotics and Automation, ICRA 2007, 10-14 April 2007, Roma, Italy*, pp. 33–39. (Cited on p. 51.)

Vincent, Lee and Pierre Soille

1991 "Watersheds in digital spaces: An efficient algorithm based on immersion simulations", *IEEE PAMI, 1991*, 13, 6, pp. 583–598. (Cited on p. 19.)

Wertheimer, Max

1922 "Untersuchungen zur Lehre von der Gestalt", *Psychologische Forschung*, 1, 1 [Jan.], pp. 47–58. (Cited on p. 51.)

Xie, Jun, Shahid Khan, and Mubarak Shah

2008 "Automatic Tracking of Escherichia Coli Bacteria", in *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2008, 11th International Conference, New York, NY, USA, September 6-10, 2008, Proceedings, Part I*, ed. by Dimitris N. Metaxas, Leon Axel, Gabor Fichtinger, and Gábor Székely, vol. 5241, Lecture Notes in Computer Science, Springer, pp. 824–832, ISBN: 978-3-540-85987-1. (Cited on p. 19.)

Yilmaz, Alper, Omar Javed, and Mubarak Shah

2006 "Object tracking: A survey", *ACM Comput. Surv.*, 38, 4. (Cited on pp. 2, 48.)

Yilmaz, Alper, Xin Li, and Mubarak Shah

2004 "Contour-Based Object Tracking with Occlusion Handling in Video Acquired Using Mobile Cameras", *IEEE Trans. Pattern Anal. Mach. Intell.*, 26, 11, pp. 1531–1536. (Cited on p. 48.)

Zhang, Bo, Jost Enninga, Jean-Christophe Olivo-Marin, and Christophe Zimmer

2006 "Automated super-resolution detection of fluorescent rods in 2D", in *Proceedings of the 2006 IEEE International Symposium on Biomedical Imaging: From Nano to Macro, Arlington, VA, USA, 6-9 April 2006*, IEEE, pp. 1296–1299. (Cited on p. 19.)

Zhu, Song Chun and Alan L. Yuille

1996 "Region Competition: Unifying Snakes, Region Growing, and Bayes/MDL for Multiband Image Segmentation", *IEEE Trans. Pattern Anal. Mach. Intell.*, 18, 9, pp. 884–900. (Cited on p. 48.)

Zimmer, C., B. Zhang, A. Dufour, A. Thebaud, S. Berlemont, V. Meas-Yedid, and J.-C. Olivo-Marin

2006 "On the digital trail of mobile cells", *IEEE Signal Processing Magazine*, 23 [May], pp. 54–62, DOI: `10.1109/MSP.2006.1628878`. (Cited on p. 15.)