Computer vision for the analysis of intra-cellular RNA localization patterns

Arthur Imbert-Meissirel

To cite this version:

Arthur Imbert-Meissirel. Computer vision for the analysis of intra-cellular RNA localization patterns. Bioinformatics [q-bio.QM]. Université Paris sciences et lettres, 2022. English. NNT : 2022UPSLM079 . tel-04097617

HAL Id: tel-04097617
https://pastel.hal.science/tel-04097617

Submitted on 15 May 2023

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

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

THÈSE DE DOCTORAT

DE L'UNIVERSITÉ PSL

Préparée à Mines Paris-PSL

Computer vision for the analysis of intra-cellular RNA localization patterns

Vision par ordinateur pour l'analyse des motifs de localisation intracellulaire de l'ARN

Soutenue par

Arthur Imbert-Meissirel

Le 15 Décembre 2022

Ecole doctorale n° 621

Ingénierie des Systèmes, Matériaux, Mécaniques, Énergétique

Spécialité

Bio-informatique

Composition du jury :

Étienne, DECENCIÈRE Directeur de recherche Mines Paris, France	*Président*
Carolina, WÄHLBY Full professor Uppsala University, Sweden	*Rapporteur*
Marcelo, NOLLMANN Directeur de recherche Centre de Biochimie Structurale, France	*Rapporteur*
Perrine, PAUL-GILLOTEAUX Ingénieur CNRS Université de Nantes, France	*Examinateur*
Florian, MUELLER Staff scientist Institut Pasteur, France	*Examinateur*
Thomas, WALTER Full professor Mines Paris, France	*Directeur de thèse*

Acknowledgement

Au moment d'écrire les derniers mots de ce manuscrit, l'occasion se présente d'apprécier le chemin parcouru. Si l'exercice d'une thèse reste une tâche solitaire, sa réussite n'en demeure pas moins liée aux rencontres que l'on fait. J'aimerais remercier ici toutes les personnes qui ont pu compter ces dernières années, par leur présence et leur soutien.

Mes premiers remerciements vont bien évidemment à Thomas et Florian. Un très grand merci pour m'avoir fait confiance et avoir accepté d'encadrer ma thèse. Merci d'avoir fait le chercheur que je suis aujourd'hui, merci pour votre bonne humeur, votre optimisme et votre écoute. Je trouve que nous avons formé une belle équipe et nos (parfois trop longues) réunions vont me manquer ! J'aimerais également exprimer ma gratitude envers Edouard Bertrand et son équipe, notamment Adham, Xavier et Emeline. Notre collaboration n'aurait pas été aussi réussie sans votre accueil enthousiaste à Montpellier, ni votre expertise. Enfin, c'est aux membres de mon jury que je souhaite adresser de sincères remerciements. Etienne Decencière, Carolina Wählby, Marcelo Nollmann et Perrine Paul-Gilloteaux, merci de me faire l'honneur de lire et de juger mes travaux de thèse.

Ces dernières années si particulières ont pu mettre en évidence à quel point la vie d'un laboratoire pouvait être importante pour notre recherche. Merci à la grande famille du CBIO pour votre enthousiasme, les discussions, les pauses cafés, et tout simplement pour partager les bons et les mauvais moments d'une thèse.

Mes pensées vont tout particulièrement à Chloé, Florian et Véronique pour leur leadership et leur bienveillance au quotidien. J'aimerais mentionner Asma, avec qui j'ai commencé cette aventure au CBIO, ainsi que la vieille garde composée de Benoit, Judith, Lotfi, Romain, Joseph et surtout Peter, pour votre accueil chaleureux à mon arrivée dans l'équipe. Merci à Thomas B. et Mélanie, pour cette improbable conférence en Terre Sainte, à Thomas D. qui, plus que tout autre dans l'équipe, a pu partager mes joies et mes peines du FISH et à Maxence, dont je garde un très bon souvenir de notre *paper reading group* du vendredi. Pouvoir encadrer ton stage a été une très belle expérience, malgré des conditions qui n'ont pas été toujours évidentes. Je remercie également Gwenaëlle, Julie, Aurélie et Philippe pour m'avoir fait découvrir vos talents de danseur, Matthieu N. (avec qui j'ai l'impression de partager 950 amis et connaissances) pour faire Movember sur 12 mois de l'année, Maguette pour tes précieux conseils en cuisine, Gwenn pour nos débats politiques et discussions hautement philosophiques, Tristan toujours prêt à "jouer" avec le nouveau modèle à la mode, Elise, dont je regrette de n'avoir pas pu assister à ta superbe soutenance, même si j'avais une très bonne raison ce jour là, Marvin, *a hell of a researcher*, Adeline pour ton énergie et tes *good vibes* au quotidien, Daniel R. pour ta motivation et ton assiduité à participer

aux book clubs et autres séminaires, Vincent pour ta curiosité, ton enthousiasme et ta patience lorsqu'il s'agit de nous (ré)apprendre la géométrie et Anne qui a décidé de mener de front une carrière de chercheuse et de médecin (parce qu'une seule des deux aurait été trop facile). La vie d'un laboratoire ne se limite pas à la recherche et c'est pourquoi je remercie grandement Caroline, Katy et Pamela pour leur aide quotidienne aux Mines ou à l'Institut Curie, et surtout pour m'avoir supporté avec ma phobie administrative.

Ma dernière salve de remerciements est réservée aux personnes extérieures au CBIO qui m'ont accompagné, formé et soutenu.

Merci Mr. Peltan et Mr. Brochier, je vous dois beaucoup. Quand je suis venu vous voir il y a quelques années avec l'idée un peu loufoque de passer un concours en candidat libre, à ma grande surprise, vous avez accepté de m'aider. Sans vous, mon parcours aurait été bien différent. Irène, Gaël et Catherine, merci de m'avoir mentoré. Tous les trois docteurs, tous les trois extrêmement compétents et bienveillants, c'est avec vos conseils et votre exemple en tête que je me suis lancé dans cette thèse.

Un grand merci enfin à mes amis et à ma famille. Merci à Thibaut toujours présent malgré l'éloignement et le temps qui passe et que je ne saurais classer parmi les amis ou la famille. Merci à Guillaume, Lucas, Théo, Florian V., Mathieu M., Hugo, Mathieu G., Adrien, Claire, Victor, Florian L. et Alexandre pour tous les weekends, les soirées (oubliées ou à oublier), les brunchs, les vacances et autres souvenirs partagés avec vous. Merci à la famille du Caire, Elisa et Gauthier, pour avoir rendu la vie parisienne si agréable. Merci à mes grand-parents. Néné, comme tu me l'as si bien dit, cette thèse c'est aussi un peu grâce à toi. Papa et Maman, merci pour votre soutien et votre amour. Merci Juna, merci Poppy, pour me donner la meilleure des raisons pour descendre les weekends à Marseille.

Finalement, ces remerciements ne seraient pas complets sans un dernier mot pour Mary, celle dont le soutien a été si important pour moi ces dernières années. L'aventure n'aurait pas été aussi belle sans toi. Du fond du coeur, merci.

Contents

Acknowledgement i

List of Figures viii

List of Tables ix

I Introduction 1

1 FISH-based transcriptomics 3
- 1.1 Gene expression 4
  - 1.1.1 Transcription 5
  - 1.1.2 mRNA transport and localization 7
  - 1.1.3 Translation 8
- 1.2 Imaging RNAs 8
  - 1.2.1 The smFISH experiment 9
  - 1.2.2 Large-scale FISH studies 10
- 1.3 Measuring images: from pixels to numbers 10
  - 1.3.1 Working with bioimages 10
  - 1.3.2 Machine Learning for bioimages 11
- 1.4 Goals and contributions 13
  - 1.4.1 Goals 13
  - 1.4.2 Contributions 14
  - 1.4.3 Manuscript summary 14

II Pipeline 17

2 FISH-quant 19
- 2.1 A Computational Framework for smFISH analysis 21
  - 2.1.1 Detection, segmentation and pattern recognition 21
  - 2.1.2 Related work 22
- 2.2 A new framework 23
  - 2.2.1 Scalability and modularity 24
  - 2.2.2 Big-FISH 25
  - 2.2.3 Sim-FISH 28
  - 2.2.4 ImJoy 29
- 2.3 Conclusion 30

3 Single RNA Detection **33**
3.1 Spot detection as a signal processing problem . . . 35
3.1.1 Problem description . . . 35
3.1.2 Related work . . . 36
3.2 Scaling mRNA detection . . . 38
3.2.1 Spot detection . . . 38
3.2.2 Managing high spot density . . . 41
3.2.3 Going beyond pixel accuracy . . . 43
3.3 Evaluation with simulated spots . . . 45
3.3.1 Simulations . . . 45
3.3.2 Results . . . 47
3.3.3 What if the PSF is not Gaussian? . . . 48
3.4 Conclusion . . . 48

4 Single-cell segmentation **51**
4.1 Segmentation of nuclei and cells in fluorescence microscopy . . . 53
4.1.1 Computer Vision for cell segmentation . . . 53
4.1.2 Related work . . . 54
4.2 Segmentation models . . . 57
4.2.1 A new multichannel dataset . . . 57
4.2.2 Nucleus segmentation . . . 57
4.2.3 Cell segmentation . . . 60
4.3 Improving cell segmentation . . . 62
4.3.1 Snake-like model . . . 63
4.3.2 In silico pre-training . . . 63
4.4 Conclusion . . . 64

5 Spatial Feature Engineering **67**
5.1 From images to coordinates . . . 69
5.1.1 Gathering all information at the single cell level . . . 69
5.1.2 Statistical description . . . 71
5.2 Hand-crafted localization features . . . 72
5.2.1 Related work . . . 72
5.2.2 Hand-crafted features to describe RNA localization patterns . . . 73
5.3 Learned localization features . . . 77
5.3.1 Related work . . . 78
5.3.2 Problem statement . . . 80
5.3.3 PointFISH . . . 83
5.3.4 Experiment . . . 85
5.3.5 Discussion . . . 89
5.4 Conclusion . . . 91

III Applications **93**

6 RNA localization by the numbers **95**
6.1 A systemic quantification of RNA localization . . . 97

6.1.1 Introduction 97
6.1.2 Materials and methods 98
6.1.3 Results 102
6.2 Local translation and translation factories 107
6.2.1 Introduction 107
6.2.2 Materials and methods 108
6.2.3 Results 109
6.3 A translation and cell cycle dependent centrosomal pattern 111
6.3.1 Introduction 111
6.3.2 Materials and methods 111
6.3.3 Results 114
6.4 The key role of KIF1C in protrusion pattern 116
6.4.1 KIF1C and protrusion mRNAs 116
6.4.2 Quantification of peripheral mRNAs 116
6.5 Conclusion 117

IV Conclusion 119

7 Conclusion and perspectives 121
7.1 Results of the thesis 122
7.2 Perspectives on the future of smFISH analysis 123
7.3 Publications 125

V Appendices 127

A Simulations 129
A.1 Spot and cluster simulations 130
A.2 Localization pattern simulations 132

B Detection and noise 135
B.1 Detection with different noise intensities 136

C Convolutional features 137
C.1 Localization features learned with convolutional neural network 138
C.2 Results 139

D Pattern classification 141
D.1 Supervised and unsupervised analyses 142
D.2 Cell-wise pattern heterogeneity 143

Bibliography 145

List of Figures

1.1 Illustration of an eukaryotic cell 5
1.2 Schema of gene expression process 6
1.3 Schema of RNA transport or locally enrichment mechanisms 8
1.4 Example of smiFISH image 9
1.5 Schema of smiFISH protocol 10
1.6 RNA localization patterns from pixels to numbers 13

2.1 Computational pipeline for a smFISH study 22
2.2 Overview of FISH-quant 24
2.3 Overview of *bigfish* 27
2.4 Overview of ImJoy 29

3.1 Example of spot detection results 36
3.2 Filtered images with LoG and DoG filters 39
3.3 Elbow curve 40
3.4 Reference spot 41
3.5 Example of dense region decomposition 42
3.6 Impact of threshold on colocalization 44
3.7 Simulation of noisy spots with *simfish* 46
3.8 Impact of noise on automated detection 48
3.9 Evaluation of dense region decomposition 49

4.1 Example of instance segmentation 54
4.2 Segmentation dataset 58
4.3 Example of in silico labeling 63
4.4 Results of in silico pre-training 64

5.1 Coordinate representation of a cell 70
5.2 Proportion of mRNAs in several subcellular regions 75
5.3 Centrosomal neighborhood 77
5.4 Simulated foci patterns 80
5.5 Simulated perinuclear patterns 81
5.6 Coordinate representations for different RNA localization patterns . . . 83
5.7 PointFISH model 84
5.8 Confusion matrix with simulated test set 86
5.9 UMAP embedding with exprimental dataset 87
5.10 F1-score distribution with hand-crafted and learned features 88

6.1 Contrasted image with Dapi and smFISH channels 99
6.2 RNA localization patterns 101
6.3 t-SNE embedding of experimental cells 103
6.4 Heat maps with localization pattern classification results 105
6.5 smFISH and SunTag images of DYNC1H1 109
6.6 Box plot with the number of detected RNA foci 110
6.7 Contrasted image with CellMask™, smFISH and GFP channels 112
6.8 RNA and centrosome detection results 113
6.9 Box plot with the proportion of centrosomal mRNAs 115
6.10 Histogram of the peripheral distribution index 117

A.1 Simulations of spots under different noise regimes 130
A.2 Simulations of clusters under different noise regimes 131
A.3 Simulation of random pattern 132
A.4 Simulations of localization patterns (first part) 132
A.5 Simulations of localization patterns (second part) 133

B.1 Elbow curves with different noise levels 136
B.2 Detection of 100 spots with different noise levels 136

C.1 Input images for the CNN 138
C.2 t-SNE projection of the features learned with CNN 139
C.3 Confusion matrix with CNN results 140

D.1 Cell-wise results of localization pattern classification 142
D.2 Heat maps with cell-wise classification results 143

List of Tables

4.1 Segmentation results 62

5.1 Annotated experimental dataset for RNA localization patterns 82
5.2 Impact of contextual inputs (PointFISH) 88
5.3 Ablation studies (PointFISH) 90

6.1 Count of annotated cells 102
6.2 Random forest accuracy 106

Part I

Introduction

1

FISH-based transcriptomics

Contents

1.1 Gene expression 4
  1.1.1 Transcription 5
  1.1.2 mRNA transport and localization 7
  1.1.3 Translation 8
1.2 Imaging RNAs 8
  1.2.1 The smFISH experiment 9
  1.2.2 Large-scale FISH studies 10
1.3 Measuring images: from pixels to numbers 10
  1.3.1 Working with bioimages 10
  1.3.2 Machine Learning for bioimages 11
1.4 Goals and contributions 13
  1.4.1 Goals 13
  1.4.2 Contributions 14
  1.4.3 Manuscript summary 14

My PhD thesis is devoted to the computational analysis of Fluorescence In Situ Hybridization (FISH) images in order to study subcellular localization of Ribonucleic Acid (RNA). In this introduction, I will start by briefly presenting the fundamentals on gene expression and the role of RNA localization therein. Then, I will present the experimental protocols that allows us to visualize and analyse RNA molecules as well as the Machine Learning methods I used. I will finish this introduction with an overview of my thesis objectives, a list of the contributions I made and a summary of this manuscript to guide the reader.

1.1 Gene expression

Cells are the basic unit of all living organisms. They display an astonishing diversity in virtually all relevant aspects, and even cells belonging to the same organism are very different in terms of appearance and specific function they fulfill. Yet, all cells belonging to the same organism share the same code, i.e. the same set of instructions, stored in the Deoxyribonucleic Acid (DNA). The DNA corresponds to a long sequence of nucleotides, where the information on the living system is stored in form of a sequence of a 4-letter alphabet. Genes are chunks of these long sequences that contain the blueprint for functional molecules. The diversity of cells in terms of appearance and function is implemented by the production of different sets of these functional molecules. This process is called gene expression.

Gene expression is the process by which genetic information is transformed into functional products. RNA molecules, or transcripts, are essential for this process. They are either the functional gene products themselves or they serve as an intermediate molecule prior to the production of other functional products, namely the proteins. A change in gene expression gives cells enough flexibility to adapt and react to different external stimuli. It also drives cells differentiation, allows them to run their basic functions, modulate their activities and therefore appears to be one of the most fundamental processes of life. On the contrary, its deregulation can lead to various diseases. This explains the interest of the scientific community to study and decode gene expression processes.

In the living world, cells can be divided in two categories: prokaryotic cells, which do not have a nucleus and typically form a unicellular organism, and eukaryotic cells, with a well-organized nucleus. An eukaryote (a living organism with eukaryotic cells) is usually composed of cells with a higher level of compartmentalization through membrane-bound organelles. The nucleus is the place where most of the DNA is stored. The rest of the organelles are located in the cytoplasm, such as the mitochondria, the lysosomes, or the Golgi apparatus (see Figure 1.1).

For an eukaryotic cell, the expression of a gene includes two main steps: the transcription of a DNA sequence into a RNA (in the nucleus) and the translation of a RNA into a protein. Between these two major steps, there is a phase of maturation for the transcript, before a potential export outside the nucleus and a transport somewhere in the cytoplasm. A cell can regulate either the transcription or the translation step to control the production of its functional molecules and thereby the biological process to which the proteins or RNA contribute.

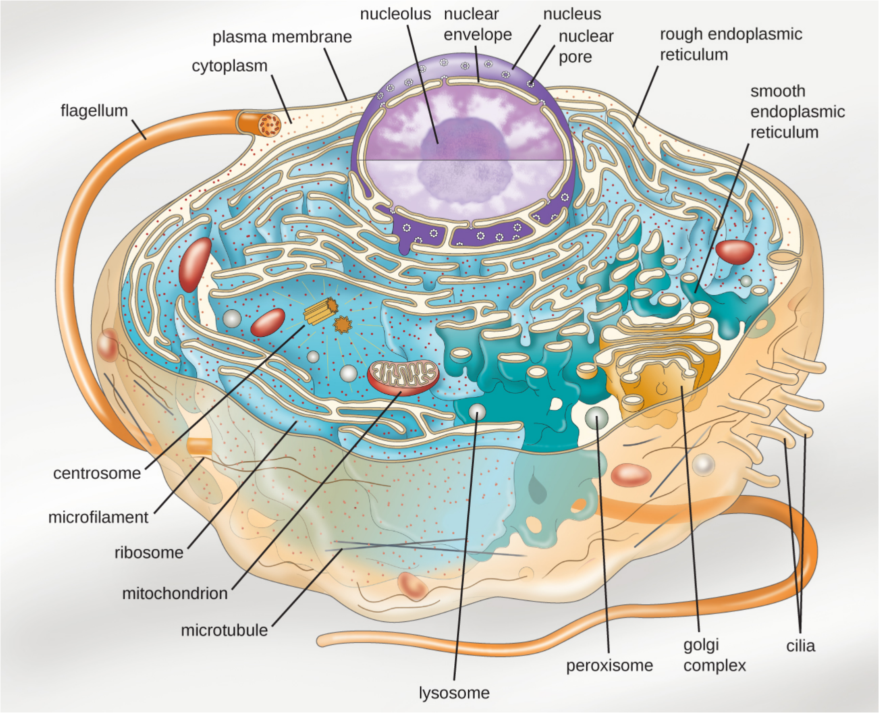

nucleolus
nuclear envelope
nucleus
nuclear pore
plasma membrane
cytoplasm
flagellum
rough endoplasmic reticulum
smooth endoplasmic reticulum
centrosome
microfilament
ribosome
mitochondrion
microtubule
lysosome
peroxisome
golgi complex
cilia

Figure 1.1: Illustration of an eukaryotic cell, from [Parker *et al.*(2017)]

1.1.1 Transcription

The primary transcripts

This step consists in copying a gene (a sequence of nucleotides along the DNA strands) into another biomolecule also composed of nucleotides: a RNA. While the DNA is located within the nucleus, the RNA can leave the nucleus by the nuclear pores. RNA can fulfill a number of different functions in the cell. We distinguish two broad categories: messenger RNA (mRNA) contain the blueprint for a protein that is synthesized outside the nucleus, while non-coding RNA are not translated into a protein. In a RNA, the nucleotide Uracil (U) replaces the Thymine (T) and a ribose sugar serves as backbone instead of a desoxyribose sugar. While both DNA and RNA contain genetic information, RNA is a shorter single-stranded molecule, compared to the double-stranded DNA, and is therefore less stable. RNA molecule also has two distinctive untranslated regions at its ends: 5'UTR and 3'UTR (corresponding to the number of carbon atoms in its sugar backbone extremities).

Transcription proceeds as follows:

1. Proteins that serve as transcription factors bind to a specific sequence of DNA (the promoter sequence) and recruit an enzyme (the RNA polymerase) to initiate the transcription.
2. The RNA polymerase breaks the hydrogen bounds between the two DNA strands to separate them.
3. The RNA polymerase moves along one DNA strand, from 3' to 5' extremity,

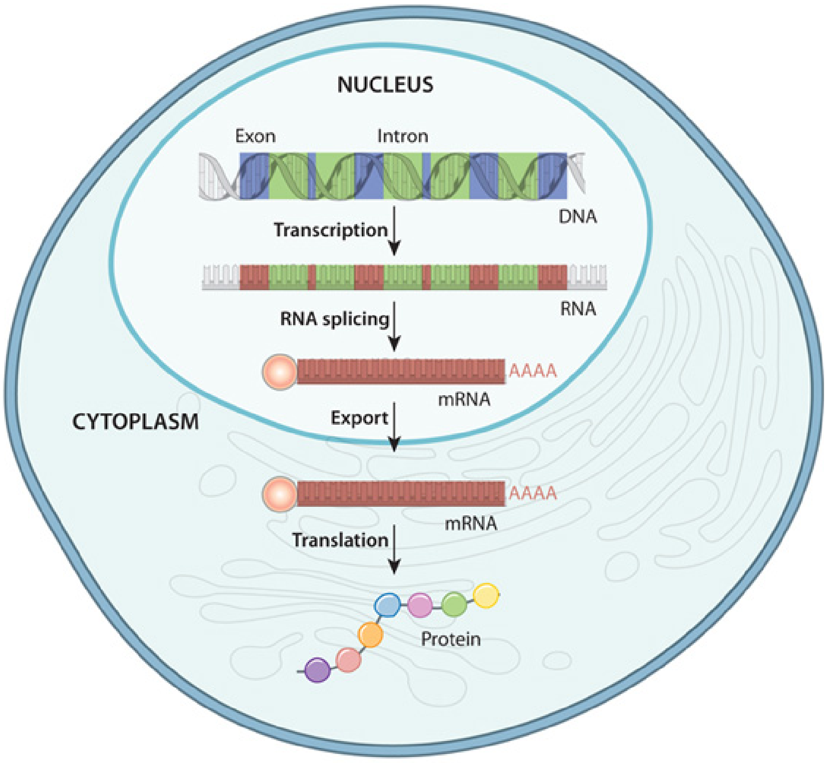


Figure 1.2: Overview of gene expression process, from [O'Connor et Adams(2010)]

synthesizing a complementary nucleotides sequence on the way.

4. The RNA polymerase disengages from the strand when it meets a specific sequence of DNA (the terminator sequence) and the newly synthesized nucleotides sequence (the primary transcribed RNA) is released.

At this point, different sorts of RNA are transcribed, depending of RNA polymerase recruited:

- mRNA, composed of coding (exon) and non-coding (intron) nucleotides, which conveys the blueprint of a future protein.
- ribosomal RNA (rRNA) which, associated with ribosomal proteins, forms ribosomes, a macromolecular machine used by the cell to synthesize proteins.
- transfer RNA (tRNA), the most abundant RNA molecule, which carries amino acids to the ribosome.

RNA maturation

For the rest of the manuscript, I mainly focus on the mRNAs. They do not fit translation requirements yet and need to undergo three processes before leaving the nucleus:

- the 5' end is transformed (RNA capping)
- a poly(A) tail (repeated adenine-based molecules) is added at the 3' end (polyadenylation)

- the non-coding parts of the RNA sequence are removed (splicing)

Those transformations enable to move the mRNA out of the nucleus, regulate its degradation and promote the translation. Figure 1.2 shows a simplified illustration of the gene expression process.

1.1.2 mRNA transport and localization

When an mRNA is ready for export, it is moved out of the nucleus through the nuclear pore complex. This complex recognizes a mature mRNA if a specific set of proteins is bound to it - poly(a) binding proteins, cap binding proteins, and proteins related to the splicing step. Once in the cytoplasm, an mRNA is not necessarily exploited by a ribosome for translation. It can also be silenced by translational repressors and stored, transported in a specific region of the cell or degraded.

Until recently, the scientific community thought translation mostly occurred at the endoplasmic reticulum and then proteins were transported where they were needed. New evidence suggests on the contrary that mRNA localization within the cell is not always random and mRNA-protein colocalization could be an important aspect of cell organization and gene expression regulation [Lécuyer *et al.*(2007)]. However, the involved mechanisms are not well understood yet and the extend of mRNAs concerned by a specific localization pattern is still unknown. Beyond the number of mRNA molecules within a cell (the expression level of a gene), researchers manifest now an increasing interest for mRNA localization [Chin et Lécuyer(2017)].

mRNA transport

There is a specific sequence in the 3'UTR of the mRNA molecule that acts like a *zipcode* [Hervé *et al.*(2004)]. This sequence can be recognized by a RNA-binding protein (RBP) and starts the formation of a ribonucleo-protein (RNP) complex. This structure is then central to coordinate the transport, the translation and the degradation of the mRNA molecule throughout the cell.

Three mechanisms to transport mRNA or locally enrich them are identified (see Figure 1.3 for illustration):

1. Active transport of mRNAs along the cytoskeleton, potentially coupled with an anchoring mechanism. This is the most common transport mechanism observed. Motor proteins are recruited by the RBPs, bind to the mRNA complex and transport it along actin filaments or microtubules.
2. At specific places, mRNA molecules are protected from degradation complexes. This localization is then locally enriched and the spatial distribution of the transcript is biased in favor of these protected localizations.
3. Transcripts diffuse randomly across the cell, but local entrapment make them accumulate with an asymmetric distribution. This mechanism is notably observed in bacteria, where no active transport of RNA has been observed [Das *et al.*(2021)].

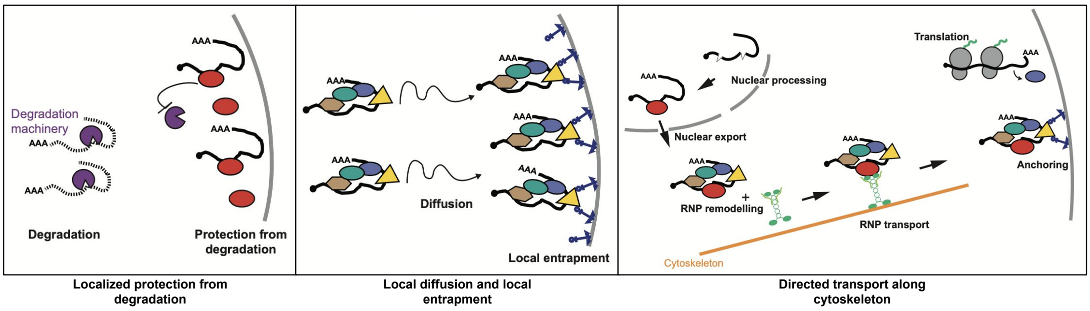

AAA
Degradation machinery
AAA
AAA
AAA
Degradation
Protection from degradation
Localized protection from degradation
AAA
AAA
AAA
AAA
AAA
Diffusion
Local entrapment
Local diffusion and local entrapment
AAA
Nuclear processing
Nuclear export
AAA
+
RNP remodelling
AAA
RNP transport
Cytoskeleton
Translation
AAA
AAA
Anchoring
Directed transport along cytoskeleton

Figure 1.3: Schema of RNA transport or locally enrichment mechanisms, adapted from [Medioni *et al.*(2012)]. (*Left*) RNAs are locally protected from degradation. (*Center*) RNAs diffuse randomly, then they are locally entrapped. (*Right*) RNAs are actively transported along cytoskeleton, thanks to RNP complexes and molecular motors. Directed transport can be coupled with an anchoring mechanism

mRNA localization

The localization of mRNA can be see as an elegant and efficient mechanisms for the spatial regulation of gene expression. By transporting one mRNA molecule and promoting a local translation, several proteins can be directly produced at a desired place. It saves the transport of a large amount of proteins [Medioni *et al.*(2012)]. It also enables the cell to quickly react to external stimuli. This mechanism could be critical to modulate the synaptic plasticity in neuron cells for example [Jung *et al.*(2012)]. Also, RNA localization provides a mechanism to bring interaction partners in close proximity. More generally, mRNA localization will increase the cell compartmentalization and enable a higher flexibility and ”fine-tuning of gene expression in both space and time” [Medioni *et al.*(2012)].

1.1.3 Translation

Finally, a cell synthesizes a protein through the translation process. A ribosome is assembled around a mRNA and moves along its nucleotides. At the same time, tRNAs carry amino acids matching the mRNA sequence. According to the genetic code, three successive nucleotides of the mRNA (a codon) correspond to an amino acid. The ribosome sequentially chains the amino acids, which then fold into a functional protein with a specific 3-dimensional structure.

1.2 Imaging RNAs

Traditional techniques to study gene expression are based on biochemical bulk-measurements, like microarrays [Schena *et al.*(1995)] or qRT-PCR [Bustin(2000)]. They aim at identifying and quantifying the presence of RNAs in a sample. These methods could not operate at the single cell level and did therefore not allow to assess expression heterogeneity between cells. Subsequent developments in next generation sequencing (NGS) allowed to address this limitation, by the advent of single cell RNA sequencing [Hedlund et Deng(2018)] or fluorescent in situ RNA sequencing (FISSEQ) [Lee *et al.*(2014)].

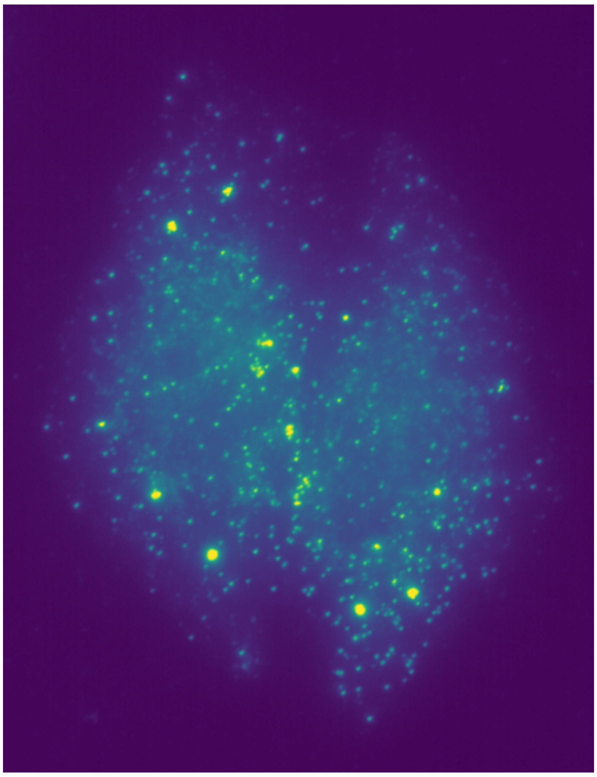


Figure 1.4: Example of smiFISH images with a maximum intensity projection. Each spot is a single RNA molecule

An alternative option is to rely on images, which naturally provide insights regarding the spatial distribution of RNAs and, with sufficient resolution, enable single cell analysis. Coupled with techniques from High Content Screening — a largely automatized experimental setup — this approach can be applied at a large scale, probing thousands of RNAs with respect to their spatial localization.

1.2.1 The smFISH experiment

Notably, single molecule FISH (smFISH) technique [Femino *et al.*(1998)], is the first to reach single molecule resolution in fixed cells. It uses multiple fluorescent oligonucleotides (or probes) that hybridize to the target RNA. Importantly, this technique preserves the cell environment, and thus enriches the contextual information available in the images. However, probes need to be specifically designed against the transcript of interest, making this solution costly to scale. A smFISH experiment includes four steps:

1. Cell fixation and permeabilization
2. Hybridization of the fluorescent probes for several hours
3. Cell washing to remove unbound probes
4. Image acquisition under a wide field microscope

As RNA molecules are smaller than the diffraction limit of the microscope, the final image results in diffraction limited spots over a fluorescent background (see Figure 1.4). In smFISH images, we do not observe the actual molecule, but its Point-Spread-Function (PSF). The background comes from both the cellular autofluorescence and the unbound probes. Single RNA molecule can be resolved because of the hybridizations of multiple fluorescent probes on the same molecule. This technique increases the Signal-to-Noise Ratio (SNR) and ensures that the RNA signal is brighter than the background.

In [Tsanov *et al.*(2016)], a cheaper and simpler version of smFISH is presented: single molecule inexpensive FISH (smiFISH). The fluorescent probe is built through a pre-hybridization step that matches a primary (unlabeled) probe with a secondary fluorescent oligonucleotide (see Figure 1.5). This design with indirect labeling presents a higher flexibility and saves cost.

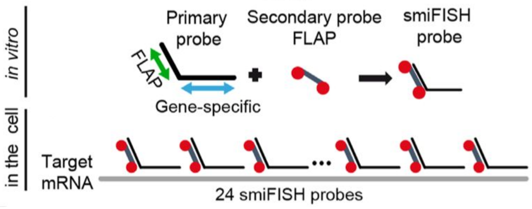


Figure 1.5: Schema of smiFISH protocol from [Tsanov *et al.*(2016)]

1.2.2 Large-scale FISH studies

FISH is an experimental technique compatible with large-scale studies. The first large scale screen focusing on RNA localization was performed in Drosophila embryos [Lécuyer *et al.*(2007)] and revealed a surprising number of mRNAs that have preferential localization during embryogenesis (71% out of more than 3000 genes tested).

A large-scale screen in mammalian cell lines was proposed a few years later [Battich *et al.*(2013), Battich *et al.*(2015)], assessing the transcriptional noise and their relationship to cellular phenotypes, as well as subcellular localization of mRNAs for a large number of genes ($N = 1000$).

Our collaborators also adapted our smiFISH protocol to 96-well plates, permitting robotized experiments and automated imaging to study hundreds of RNA species [Tsanov *et al.*(2016), Safieddine *et al.*(2021)].

One of the limitations of these smFISH approaches with respect to single cell RNAseq is that they only visualize the localization and expression of transcripts one RNA species at a time. More recent developments relying on sequential hybridization steps and barcoding RNAs, allow now the joint analysis of hundreds of RNAs in the same cells [Lubeck *et al.*(2014), Chen *et al.*(2015), Eng *et al.*(2019), Fazal *et al.*(2019)]. These techniques were not available in practice at the time of the data collection for this PhD thesis, but in principle, they represent a very interesting alternative to smFISH.

1.3 Measuring images: from pixels to numbers

1.3.1 Working with bioimages

Over the past decades, a number of technological breakthroughs and new experimental techniques have transformed biology in depth. The availability of large, systematically acquired datasets allow today to perform integrated systems-level analyses. This transformation also significantly increases the importance of data collection and maintenance, as well as robust quantitative approaches allowing to extract valuable insights from these datasets.

In bioimaging specifically, current experimental platforms allow us to perform a large number of experiments under varied conditions and highly informative content. These techniques are referred as High Content Screening and pave the way for a systematic analysis of living systems. Finally, the increasing degree of automation of experimental protocols justifies the development of bioimage informatics as a discipline in its own right to assist biologists.

The use of images in biology brings several advantages: (1) images are informative about morphological phenotypes, (2) they preserve the spatial dimension of the living system studied, as well as the temporal dimension in the case of repeated image acquisitions in live imaging experiments and (3) they allow to access different scales of organization, from the molecular scale to the tissue architecture.

Bioimage Informatics deals with all computational aspects of images in the life sciences. There is a large variety of different tasks, such as image restoration, denoising, segmentation, tracking, registration and recognition of specific phenotypes. Usually, image datasets are characterized by an important variability: living systems can be imaged at different scales, with different modalities of acquisition and evolving technology. In addition, the fluorescent markers used in microscopy or the question investigated by researcher may differ, making images of the same object from two different studies not necessarily compatible. As a consequence, the field often requires tailored solutions to the problem at hand. However, there can also be for some projects a common algorithmic backbone, that just needs to be adapted to new projects.

1.3.2 Machine Learning for bioimages

A Machine Learning boom

The recent years have seen a renewed interest for Machine Learning research, the cornerstone of current artificial intelligence era, driven by successful applications in vision, language understanding, speech recognition, etc... This term of Machine Learning was coined to describe the set of techniques and models that make automatic decision based on patterns *learned* and found in data. For some specific tasks (like image classification) these algorithms have reached human-like capabilities. For other fields, the gap between computer and human performances keeps decreasing. In computer vision, the success of Machine Learning models makes their use legitimate for bioimage informatics applications.

Overall, three ingredients make these successes possible:

1. methodological advances regarding the predictive models
2. large annotated datasets
3. increased computing power to process an ever growing number of operations

While new developments certainly are important for the performance of Machine Learning, it must be noted that many of the principles in Machine Learning and Deep

Learning in particular (a family of Machine Learning models based on artificial neural networks) have been studied and developed for decades (see [Bishop(2006), Hastie *et al.*(2009)] for a review of different Machine Learning methods). Convolutional Neural Networks, which are the most widely used neural networks in Computer Vision, were developed more than 25 years ago [LeCun *et al.*(1989)], but became the state-of-the-art method only after 2012 [Krizhevsky *et al.*(2012)], when they won a classification challenge on a large dataset of natural images [Deng *et al.*(2009)]. In biomedical community, Machine Learning techniques are disseminating and address more challenges every year (see [Jumper *et al.*(2021)] for an example with the protein folding problem).

Neural networks as the next generation tool

In case of Deep Learning, model architectures are based on neural networks with successive layers of artificial neurons. Each neuron is defined by a set of weights and perform a linear combination of the input signal with its weights, followed by a potential non linear transformation (usually a *maxout*, *sigmoid* or *tanh* function). A complete network usually combines these layers of neurons with normalization steps. It can be seen as a parametrized model, whose weights need to be optimized to minimize for a specific task.

Given an input, a loss function enables to compare the output signal of the model with a ground truth (the expected output signal) and thus compute a gradient (the first order derivative of the loss function). The backpropagation of this gradient to the rest of the network makes it possible to update the weights of the layers in order to minimize the loss [Rumelhart *et al.*(1986)]. This process defines a training step. After several iterations, the loss and the gradient should decrease, and the network progressively stabilizes its weights. The weights are ultimately frozen and the model is trained.

Such architecture is highly flexible and different variants have been developed [LeCun *et al.*(2015)]. For a general introduction to Deep Learning techniques, one can refer to [Goodfellow *et al.*(2016)].

Limitations and caveats

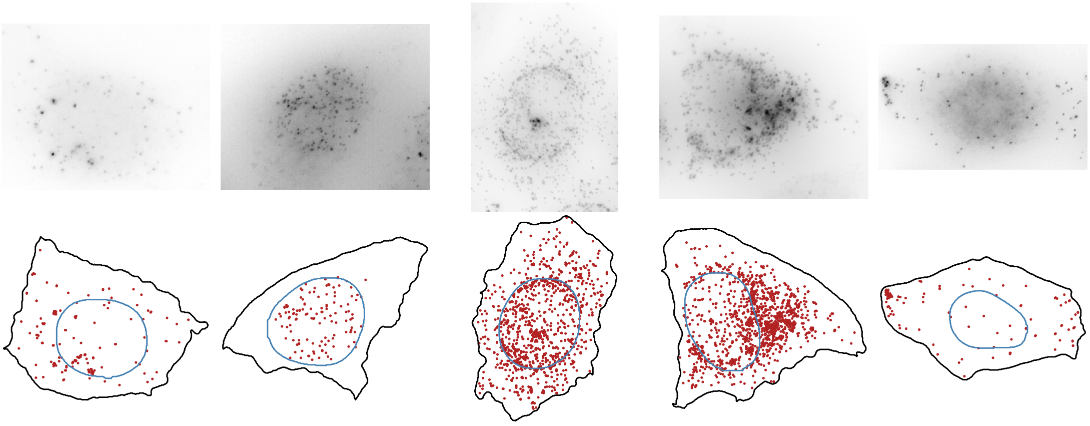
Beyond the great performances returned by Machine Learning models, a number of limitations and caveats should be considered.

First, they require annotated datasets to train and evaluate the models. Compared to natural images, this ground truth is particularly costly to obtained with biomedical images as it often involves the assignation of experts (medical doctor or biologists) to a time-consuming, repetitive and annoying task. As a consequence, datasets available for a given biomedical problem are often too small, highly diversified or released without any useful manual annotations.

Second, Machine Learning algorithms require robust evaluation protocols, strong benchmarks and a rigorous choice of evaluation metrics[1]. It is necessary to prevent data leakage and firmly separate the training set from the test set in our dataset, otherwise models overfit to the training samples and generalize poorly to unknown data. If the

---

[1] One can see [Varoquaux et Cheplygina(2022)] for a description of these evaluation caveats when Machine Learning models are applied to medical images.



(a) Foci (b) Intranuclear (c) Nuclear edge (d) Perinuclear (e) Protrusion

Figure 1.6: RNA localization patterns from [Imbert *et al.*(2022a)]. (*Top*) Typical smFISH images with different RNA localization patterns. (*Bottom*) Coordinate representations with RNA spots (*red*), cell membrane (*black*) and nuclear membrane (*blue*). Detection and segmentation results are extracted and visualized with FISH-quant [Imbert *et al.*(2022b)]

test set is not representative of the data distribution, or too small, the evaluation will be biased and hardly reproducible [Varoquaux(2018)]. Obviously, all these limitations are worsened by the inherent heterogeneity of bioimages and the difficulty to assemble manual annotations. Working with Machine Learning models, one needs strong and diversified baselines that reflect the state-of-the-art of performances, as well as the right choice of metrics to benchmark the developed methods with respect to approaches published in the literature.

Third, flexibility of some models comes at a price: a sensitivity to the selected hyperparameters, which makes the evaluation all the more important.

Last but not least, most of the Deep Learning models appear like a black box solution with an internal process that cannot be easily or directly interpreted. While this problem is more critical for direct medical applications than biological ones, it requires an important pedagogical effort to diffuse Deep Learning techniques to the rest of the scientific community.

1.4 Goals and contributions

1.4.1 Goals

My PhD had three main objectives:

1. Development of a full workflow for the analysis of smFISH images, containing state-of-the-art methods for image segmentation (nucleus and cytoplasm), a robust method for RNA detection applicable to large-scale screens and classification of RNA localization patterns.

2. Implementation of the methods in a python based open source tool, usable by the scientific community.

3. Application of the developed methods and tools to real experimental datasets in ongoing screens on subcellular RNA localization.

Figure 1.6 illustrates the gap that needs to be bridged between the input images and an adequate representation of the RNA localization, making their quantitative analysis possible.

1.4.2 Contributions

Beyond this manuscript and the publications, my contributions are essentially lines of code. In an effort of transparency, reproducibility and documentation, I developed computational tools as useful as possible for any biologist interested in FISH experiments. My PhD results in three major contributions.

- My first and main contribution is FISH-quant V2. This online framework gathers Python packages and a Graphical User Interface (GUI) to process FISH images, build robust analysis pipelines and perform simulations.
- My second contribution is an alternative method to compute relevant features in order to discriminate RNA localization patterns. This method implied the development of PointFISH, a dedicated Deep Learning model operating directly on point clouds.
- My third contribution was the participation in biological studies, where I could apply the computational methods and tools I had developed. These studies leveraged high content screening assays and smFISH techniques to perform a systemic analysis of RNA localization and investigate local translation.

Additional contributions are mentioned throughout this manuscript, including the annotation of a dataset for nucleus and cell segmentation with thousands of instances and the supervision of an intern. The work performed during this internship was the seed for the first publication of another PhD student in the team, about in silico labeling and segmentation. Lastly, a list of my publications are available at the very end of the manuscript.

1.4.3 Manuscript summary

The manuscript is composed of two parts. The first four chapters are a presentation of the analysis pipeline with a focus on every critical stage. It includes systematic reviews of the existing methods and details about solutions I implemented. In addition, code snippets, ready to be imported and run, are interspersed with descriptions of the related algorithms. The second part details several applications of my tools, where quantification of RNA localization patterns provides biological insights.

In **Chapter 2**, I give an overview over the general computational framework and the design of the software I have developed and published during my PhD thesis: FISH-quant v2. It includes methods for every stage of a FISH-based analysis, with an effort to make them scalable and modular. I also describe the improvements from the original FISH-quant version in MATLAB, and how they address the requirements of a modern software tool.

In **Chapter 3**, I focus on RNA detection. With smFISH experiments, the RNA molecules are spotted and reduced to discrete data points in space. I describe detection algorithms and their extensions available in FISH-quant. In the end, the set of all RNA molecules is reduced to a point cloud with spatial coordinates.

In **Chapter 4**, I review and describe algorithms for nucleus and cell segmentation. Most of them are Deep Learning models, trained on vast datasets of annotated images. Beside my own implemented methods, I also present refinement techniques to improve and format segmentation masks. Two other projects for which I contributed (including one published) are also discussed at the end of the chapter. They aim at improving the efficiency and consistency of segmentation on specific aspects.

In **Chapter 5**, I compare two different approaches to quantify RNA localization patterns. Once the RNA molecules are detected and cell morphology is segmented, the pixel domain gives way to Euclidean space and cells can be represented with a coordinate representation. Then, spatial features are computed in order to discriminate relevant localization patterns. I present two different methods of feature engineering. The first method consists in manually designing features to characterize specific localization patterns. I then list and describe every hand-crafted feature implemented in FISH-quant. The second (published) method consists in learning features. They are extracted from a Deep Learning model fed with the RNA point cloud as input and trained on a simulated pretext task.

In **Chapter 6**, I present several applications of my quantitative pipeline. These applications come from three publications where we study different RNA localization patterns. I designed a classification pipeline to recognize a variety of localization patterns. An additional series of experiments which I computationally analyzed, led to the discovery of translation factories, a novel mechanism for the spatial control of gene expression. In a different screen, I developed additional features to quantify a cell cycle dependent pattern, which is related to centrosomes. Finally, I also contributed to a study focused on the protrusion pattern, where RNA localize in cell extensions.

Part II

Pipeline

2

FISH-quant

Abstract:

FISH-quant v2 is a Python-based computational framework dedicated to the analysis of smFISH images. The framework includes a Python package with a fully automated spot detection algorithm as well as pre-trained deep learning models for cell and nucleus segmentation. It also includes a set of hand-crafted features to quantitatively describe the spatial distribution of RNAs within cells. A Python simulation package, a graphical user interface, and interactive examples complete the framework. Finally, the software architecture allows for the development of modular and scalable workflows, is open-source and documented.

Résumé:

FISH-quant v2 est un logiciel Python dédié à l'analyse d'images smFISH. Le logiciel comprend une bibliothèque Python avec un algorithme de détection de spots entièrement automatisé ainsi que des modèles d'apprentissage profond pré-entraînés pour la segmentation des cellules et des noyaux. Il comprend également un ensemble d'indicateurs permettant de décrire quantitativement la distribution spatiale des ARNs dans les cellules. Une bibliothèque de simulation Python, une interface graphique et des exemples interactifs complètent le logiciel. Enfin, l'organisation de FISH-quant permet le développement de pipeline modulaire et à grande échelle. L'outil est open-source et documenté.

Contents

2.1 A Computational Framework for smFISH analysis 21
- 2.1.1 Detection, segmentation and pattern recognition 21
- 2.1.2 Related work 22

2.2 A new framework 23
- 2.2.1 Scalability and modularity 24
- 2.2.2 Big-FISH 25
- 2.2.3 Sim-FISH 28
- 2.2.4 ImJoy 29

2.3 Conclusion 30

In this chapter I present FISH-quant v2, a computational framework for the analysis of smFISH images. FISH-quant v2 contains methods for every step of a FISH-based study. Based on a previous MATLAB package [Mueller *et al.*(2013)], I developed an improved and extended version, that is both scalable and modular and thus fulfills the requirement of a modern software tool. The chapter mainly describes the work presented in the paper [Imbert *et al.*(2022b)]:

A. Imbert, W. Ouyang et al. (2022), *FISH-quant v2: a scalable and modular tool for smFISH image analysis*, RNA, pp. 786–795, iSSN: 1355–8382, 1469–9001.

2.1 A Computational Framework for smFISH analysis

In this section, I present the functionalities expected from a modern and efficient computational analysis framework for smFISH images.

2.1.1 Detection, segmentation and pattern recognition

smFISH permits the visualization of single RNAs in cells and tissues and thus allows the exploration of the subcellular spatial distribution of RNA molecules. The analysis of smFISH images aims at localizing and counting individual RNAs in single cells and analyze their spatial configuration with respect to subcellular landmarks (e.g. nuclear and cytoplasmic membranes). It typically encompasses a sequence of interconnected steps, as illustrated in Figure 2.1:

- detecting isolated and clustered RNA molecules
- segmenting cells and the relevant cellular compartments such as nuclear and cytoplasmic membranes, mitochondria, centrosomes, etc. (depending on the focus of the study and the markers used)
- performing cell-level analysis of expression levels and RNA localization patterns

With recent technological advances, FISH can be scaled up and therefore be applied in screening projects, where hundreds or thousands of experiments can be performed using a high degree of automation [Safieddine *et al.*(2022)]. This then results in large and complex image datasets.

While such large-scale imaging methods can provide experimental data to understand RNA localization at a systems level, they come at a price: the need for fully automated, robust image analysis and user-friendly software tools to analyze such data sets and to fully exploit their potential. Several specifications can be defined a priori for such an analysis tool. It should be simple enough to be mastered by non-experts, especially noncoders. Yet, it should be flexible enough to address different experimental designs and rely on a common algorithmic backbone. With the same modules, users should be able to both perform a high content screening analysis on a remote cluster, and a local analysis of a single image. Finally, the software should integrate the latest generation of computer vision algorithms, in particular deep-learning-based methods for image segmentation.

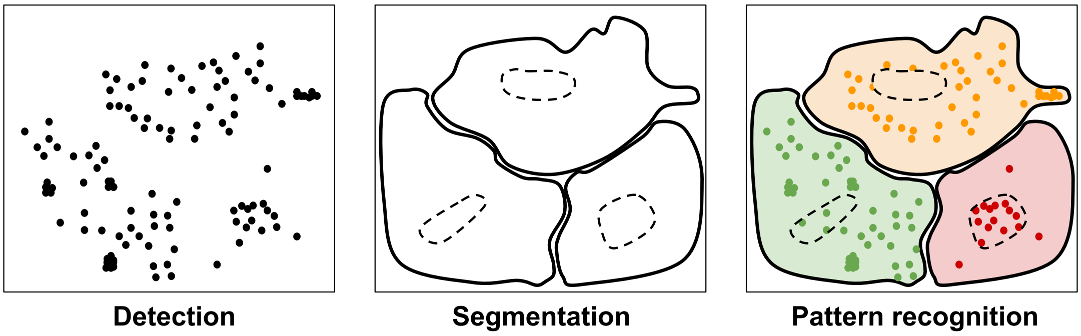


Figure 2.1: Computational pipeline I use as reference for a smFISH study

2.1.2 Related work

Processing a large-scale smFISH study requires both specific methods to analyze RNA distribution and more general techniques to read and manipulate images based on robust computer vision algorithms. A FISH analysis pipeline hence has to include several inter-dependent subtasks: an accurate RNA detection method, the segmentation of relevant subcellular regions such as the cytoplasm and the nucleus, and preprocessing techniques to prepare the images (such that filtering, denoising or projection algorithms). The obtained results can then be exploited to compute gene expression levels, by counting RNA molecules, or develop localization features, to study non-random RNA localization.

Specialized solutions for many of these subtasks exist, like BlobFinder [Allalou et Wählby(2009)] for spot detection, but using them requires the end-user to assemble a complex analysis workflow, often across different programming languages. This can be a daunting task, especially for wet-lab biologists with no formal training in computer science.

As an alternative, the development of complete analysis workflows can provide a modular and extensible pipeline for the analysis of RNA abundance and localization. As an example, the Pelkmans Lab published several papers illustrating the need for such complete and robust pipelines. In [Battich *et al.*(2013), Stoeger *et al.*(2015)], the authors develop a modular analysis pipeline permitting to analyze subcellular RNA localization from their high-content screen. The latter paper is a detailed protocol, that permits new users to perform their analysis quickly by providing a detailed description of the implemented approaches. Further, the advantage of their modular approach, is that additional analysis tasks can be integrated to study related biological questions. In latter work [Battich *et al.*(2015)], they reused parts of this pipeline to infer RNA abundance per cell, and completed this analysis with morphological and phenotypic analysis to explain the observed heterogeneity in RNA abundance.

These examples can provide several guidelines for future developments. First, it is preferable to have a modular framework addressing every step in the smFISH analysis pipeline, to facilitate usage. Second, there is a balance to find between the scope of a framework, its flexibility, its scalability and the amount of user input required. Lastly, to increase usability of a flexible framework, the analysis steps should be kept modular and be clearly explained.

While several tools exist for each stage illustrated in Figure 2.1, there was - to our knowledge - no tool available that permits performing the entire analysis in one framework. A complete analysis pipeline has then to be built by mixing these tools and requires some in-house developments, which can be daunting for non-specialists and may provide solutions that are unstable and difficult to scale. For the segmentation, deep learning has become the method of choice with dramatic improvements in segmentation accuracy as compared to traditional methods. Spot detection can be performed with different Python or Matlab packages. However, assigning spot counts to segmentation results and the subsequent analysis of RNA levels and RNA localization require custom-written code [Stoeger *et al.*(2015),Samacoits *et al.*(2018)]. General image analysis tools, such as CellProfiler [McQuin *et al.*(2018)], CellCognition [Held *et al.*(2010)] or Trackmate [Ershov *et al.*(2022)], permit us to establish an analysis framework daisy-chaining some of these analysis steps, but do not permit us to perform the entire analysis. Steps are missing, usually the ones focused on the RNA distribution and localization features computation.

A first attempt to build a software self-sufficient for a smFISH analysis was the Matlab version of FISH-quant [Mueller *et al.*(2013)]. Coupled with a flexible smiFISH approach [Tsanov *et al.*(2016)], FISH-quant progressively integrated detection and segmentation solutions, in addition to dedicated localization features [Samacoits *et al.*(2018)]. Today, a number of approaches specifically dedicated to the analysis of smFISH are available or in development. DypFISH [Savulescu *et al.*(2019),Savulescu *et al.*(2021b)] permits the study of the spatial distribution of mRNAs and proteins of micropatterned cells. They mix tools implemented in Python and Icy [de Chaumont *et al.*(2012)]. Recently, Bento [Mah *et al.*(2022)] was released to process Sequential FISH (SeqFISH)+ and Multiplexed Error-Robust FISH (MERFISH) images, explore and compute spatial features like FISH-quant, with a single-molecule accuracy. Lastly, StarFISH [Perkel(2019)] is an ongoing software development mainly aiming at solving problems related to multiplex smFISH data for application in spatial transcriptomics. They made a significant effort to be able to read and process images generated by any FISH method.

2.2 A new framework

While an impressive range of computational methods already exists, a unified framework dedicated to smFISH experiments was lacking. This prevents users, especially non-specialist, from performing an accurate and large-scale analysis. To address this, I designed a Python-based version of FISH-quant to fulfill the above-described requirements in a flexible and efficient way [Imbert *et al.*(2022b)]. Contrary to the first version of FISH-quant in Matlab, I address and improve on each of the specifications mentioned in section 2.1.1. The switch to Python allows me to develop a flexible, free and fully open-source software. FISH-quant v2 enjoys a better integration to other open source tools and frameworks, from data analysis to web-based user interaction. Importantly, FISH-quant v2 facilitates the use of machine learning or deep learning algorithms with the import of dedicated packages, such as scikit-learn [Pedregosa *et al.*(2011)] or TensorFlow [Abadi *et al.*(2015)]. I also improve the scalability and the modularity of the package: the software has now been applied to several High Content Screening projects [Chouaib *et al.*(2020),Safieddine *et al.*(2021),Pichon *et al.*(2021)]. Lastly, by

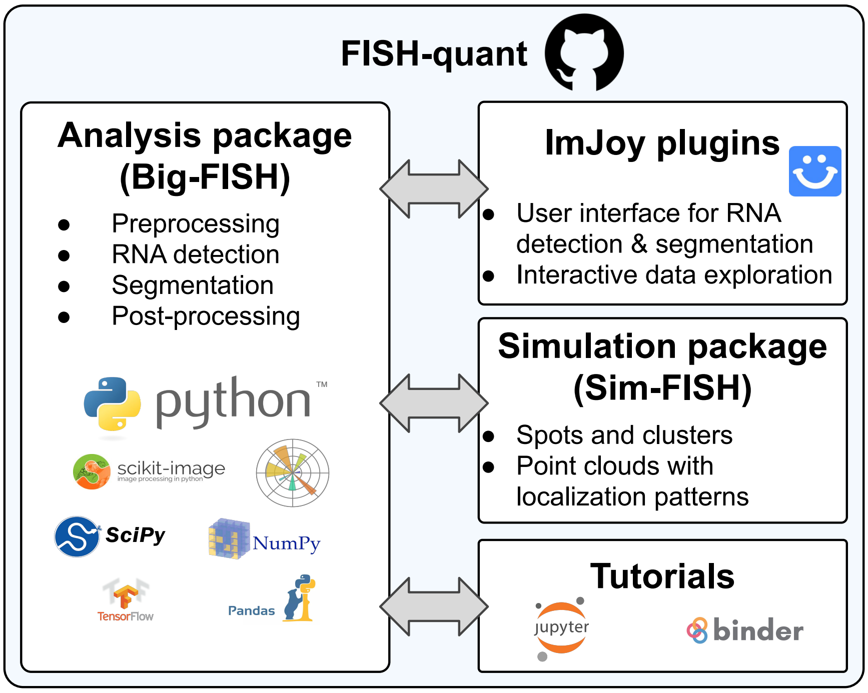

FISH-quant
Analysis package (Big-FISH)
Preprocessing
RNA detection
Segmentation
Post-processing
python
scikit-image
SciPy
NumPy
TensorFlow
Pandas
ImJoy plugins
User interface for RNA detection & segmentation
Interactive data exploration
Simulation package (Sim-FISH)
Spots and clusters
Point clouds with localization patterns
Tutorials
jupyter
binder

Figure 2.2: Overview of FISH-quant. The software is hosted on GitHub and consists of several interconnected repositories. The Python core package *bigfish* contains the entire analysis code, which is used by the ImJoy GUIs, the tutorial repository and a simulation package *simfish*

using ImJoy [Ouyang *et al.*(2019a)], a recently developed data analysis framework, we provide browser-based GUI for both launching image analysis and downstream analysis of the results, and the computation can be performed locally or seamlessly scale to powerful remote computing servers.

In this section, I will describe the overall organisation of FISH-quant v2 and the architecture of its main components: *bigfish*, *simfish* and the ImJoy plugins.

2.2.1 Scalability and modularity

FISH-quant v2 is entirely open-source and hosted on GitHub under the FISH-quant organization[1]. Using a GitHub organization allowed me to provide dedicated repositories with well defined and dedicated scope (see Figure 2.2). Further, it gives the flexibility for future extension where new projects can be integrated as new, independent repositories, without affecting and complexifying the already existing code. The user can choose the adequate code for the analysis needs, without the overhead of installing unnecessary packages. This GitHub organization is organized in several resources with dedicated repositories and documentation.

First, I implemented a Python package (*bigfish*) providing the core code for performing scalable computation and analysis. Second, I provide detailed interactive examples with test data for each analysis step that are available in Jupyter notebooks. These examples can also be run directly on Binder [Jupyter *et al.*(2018)], a free and reproducible

---

[1] `https://github.com/fish-quant`

Jupyter notebook service, without local installation. Third, a Python package (*sim-fish*) allows the simulation of different subcellular RNA localization patterns. Fourth, ImJoy plugins [Ouyang *et al.*(2019a)] provide a GUI for the most commonly used workflows, and an interactive tutorial that can also run directly without local installation. Lastly, a landing page[2] centralizes implemented tools and directs new users to the most relevant resource for their analysis needs.

Dependencies for the Python packages are limited to standard Python scientific libraries: scientific computing (numpy [Harris *et al.*(2020)], SciPy [Virtanen *et al.*(2020)]), data wrangling (pandas [Wes McKinney(2010)]), image analysis (scikit-image [Walt *et al.*(2014)]), visualization (matplotlib [Hunter(2007)]), parallel computing (joblib[3])and machine learning (scikit-learn [Pedregosa *et al.*(2011)], TensorFlow [Abadi *et al.*(2015)]). The GitHub repositories are using continuous integration providing increased robustness of the released code, through unitary testing, version control and automatically generated up-to-date documentation. Finally, packages are hosted under a BSD 3-Clause License.

2.2.2 Big-FISH

pip install big-fish

We chose to implement the core analysis package *bigfish* in Python. Compared to MATLAB, Python allows the development of a free and fully open-source software. It also provides access to established libraries for data and image analysis, in addition to the most popular deep learning frameworks. Lastly, Python packages can be interfaced with other tools and frameworks, from data analysis to web design, to provide interactive tools for user interaction and data inspection.

As shown in Figure 2.3, *bigfish* includes several independent subpackages for the different steps in a smFISH analysis workflow: preprocessing, detection, segmentation, and analysis. I designed each subpackage with clearly defined input and output data formats, which are automatically checked. Each of these packages can be used independently in a modular fashion. Users can thus create a customized analysis workflow, starting from preprocessing of images to statistical interpretation of results. These workflows can be implemented in Python and Bash scripts and run both on local and remote computational resources. The modular design also permits the easy integration of external methods, for instance, a new segmentation method can be combined with this spot detection algorithm.

More specifically, the Python code used in *bigfish* package is organized in 7 subpackages performing dedicated steps:

- *bigfish.stack* - I/O operations and images preprocessing
- *bigfish.detection* - mRNA spot detection
- *bigfish.segmentation* - nucleus and cell segmentation
- *bigfish.multistack* - post-processing and analysis of results from different channels, such as the merging of RNA detections and segmentation masks or colocalization

---

[2] `https://fish-quant.github.io/`

[3] `https://github.com/joblib/joblib`

analysis

- *bigfish.classification* - localization feature computation
- *bigfish.plot* - visual reports of the obtained results
- *bigfish.deep_learning* - deep learning algorithms and pretrained models for segmentation or point cloud analysis

In this chapter, I provide only an overview of these subpackages. More details can be found in subsequent chapters where I discuss particular algorithms that I have designed or implemented, or both. I would also like to refer interested readers to the package documentation[4] or the GitHub repository[5]. Dedicated notebook tutorials are also available[6]. They can be run directly in the browser with Binder [Jupyter *et al.*(2018)] and provided sample data, and thus allows new users to immediately test the package.

Building blocks for a full analysis pipeline

The package *bigfish* aims at addressing the main challenges in smFISH analysis.

First, for image handling and preprocessing, I implemented a number of different utility functions to read, write, normalize, cast, filter, and project images. Different image file formats are natively supported and both 2D and 3D images can be processed. These methods are gathered in **bigfish.stack**.

Second, the detection subpackage (**bigfish.detection**) provides methods required to detect spots in 2D or 3D images. An important aspect of my implementation is its ability to detect spots without setting any pixel intensity threshold. I propose a method to automatically infer this threshold from the image. Such automatization overcomes human intervention and allows scaling to large data sets, such that the subpackage can process thousands of images. While initially designed to detect individual mRNAs, this subpackage permits the detection of larger spot-like structures like P-bodies or centrosomes (see applications in chapter 6 for more details). Furthermore, I provide a fitting method to localize spots with a subpixel accuracy, a colocalization algorithm and cluster detection. Strong local accumulation of RNAs can lead to an underdetection since such accumulations are counted as single RNAs. For such cases, I also provide tools to decompose these dense regions and estimate the right number of spots. The chapter 3 gives a comprehensive description of the implemented detection methods.

Third, the segmentation subpackage (**bigfish.segmentation**) contains several algorithms and utility functions for segmentation and post-processing. It provides a simple deep-learning-based approach to segment cells and nuclei, but also traditional methods such as thresholding or watershed. Furthermore, different post-processing tools are available to refine and clean the segmentation result, such as boundary smoothing, removal of small objects or filling of small holes. These algorithms are presented with more details in the chapter 4.

Forth, detected RNA point clouds need to be matched to their cellular or subcellular environment. For this, the subpackage **bigfish.multistack** enables to combine

---

[4] `https://big-fish.readthedocs.io/en/stable/`

[5] `https://github.com/fish-quant/big-fish`

[6] `https://github.com/fish-quant/big-fish-examples`

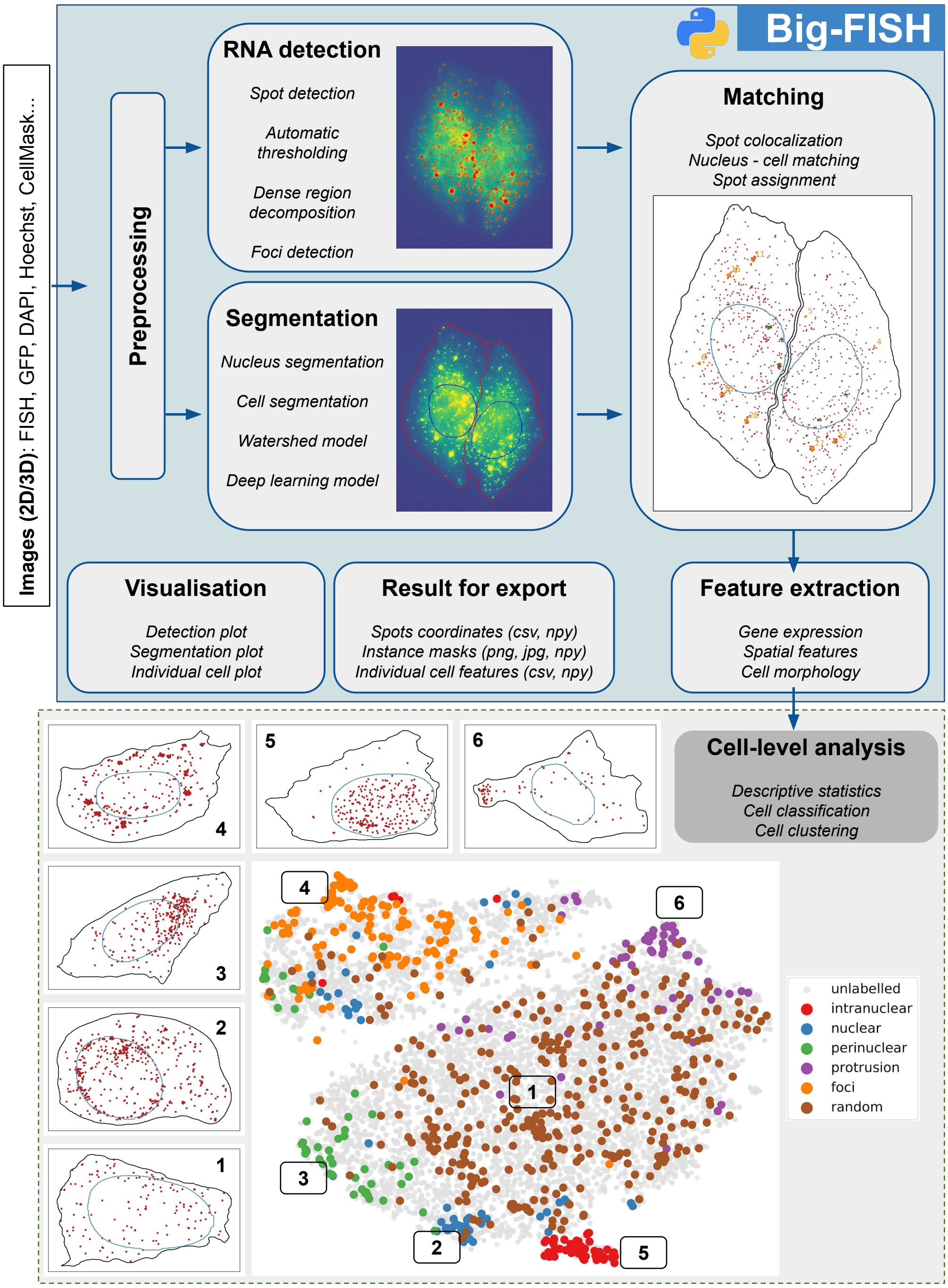

Big-FISH
Images (2D/3D): FISH, GFP, DAPI, Hoechst, CellMask...
Preprocessing
RNA detection
Spot detection
Automatic thresholding
Dense region decomposition
Foci detection
Segmentation
Nucleus segmentation
Cell segmentation
Watershed model
Deep learning model
Matching
Spot colocalization
Nucleus - cell matching
Spot assignment
Visualisation
Detection plot
Segmentation plot
Individual cell plot
Result for export
Spots coordinates (csv, npy)
Instance masks (png, jpg, npy)
Individual cell features (csv, npy)
Feature extraction
Gene expression
Spatial features
Cell morphology
Cell-level analysis
Descriptive statistics
Cell classification
Cell clustering
1
2
3
4
5
6
unlabelled
intranuclear
nuclear
perinuclear
protrusion
foci
random

Figure 2.3: Overview of *bigfish*. (*Upper part*) Main modules illustrated with a typical analysis workflow. Shown are also the inputs and outputs that are created at the different steps. (*Lower part*) Not directly included in *bigfish*. As a final result, each cell is described with a set of features reflecting RNA abundance and localization. These features can then be used to perform analysis on the cell population. Results from [Chouaib *et al.*(2020)] are illustrated as example (see chapter 6 for more details). The t-SNE plot projects 15 localization features for smFISH experiments against 27 different genes. Each dot is one cell. The color-coded dots are manual annotations of six different localization patterns. Images are examples of individual cells displaying a typical localization pattern of this region of the t-SNE plot

detection and segmentation results, obtained from different input channels. It permits the analysis of RNA abundance and distribution at the single-cell level. Detected spots can be assigned to a specific region of interest, for instance, a cell or a nucleus. Using the same method, RNA clusters can be assigned to a nucleus and thus be considered as transcription sites. RNA expression levels are computed within this subpackage, as this is usually the minimum information that is extracted from a spatial transcriptomics study.

Finally, the subpackage **bigfish.classification** computes features at the single cell level from the spot positions and the coordinates of cellular landmarks. These features allow a statistical description of the cell population or can feed a classification model allowing for the discrimination of individual cells according to their RNA localization patterns. More information about the cell matching step and the feature engineering are presented in chapter 5.

In *bigfish*, I also provide a subpackage (**bigfish.plot**) to visualize the results of each intermediate step in the analysis workflow and thus provide valuable visual quality control. A last subpackage (**bigfish.deep_learning**) gathers the utility functions and model architectures to run deep learning solutions. The use of such techniques implies more complex dependencies in the back-end like TensorFlow [Abadi *et al.*(2015)]. By isolating pieces of code related to deep learning, I make the import of these frameworks optional for the user. Indeed, the majority of the methods provided by *bigfish* does not require artificial neural networks.

2.2.3 Sim-FISH

The Python package *simfish* is dedicated to the simulation of FISH images. These simulations come in two flavors: simulation of smFISH image simulation and simulation of point cloud coordinates with a specific localization patterns.

Spots or clusters of spots with various intensities and shapes can be simulated, in 2D or 3D, optionally with subpixel accuracy. Different parameters, such as the number of spots, the size of the clusters and even the background noise level or randomness in the image can be controlled by the user. These simulated images can be used to validate spot or cluster detection methods, for instance to evaluate the impact of varying noise levels. I refer to the chapter 3 for the details about the generation of these images.

The simulated point clouds are used to build a large dataset with different RNA localization patterns. They can be used for benchmarking and ultimately for the training of neural networks for classification of localization patterns. The output of the simulation is not an image, but a list of 3D RNA coordinates as well as the 2D cell and nuclear boundaries. The localization pattern can be chosen among 9 predefined patterns, and the strength of the pattern (and thus the difficulty of recognizing the pattern) can be controlled by one parameter. Chapter 5 provides a thorough description of the simulations.

More generally, such simulations can be helpful to validate, calibrate or pre-train algorithms. This is particularly true when we deal with experimental data difficult to generate or annotate. Lastly, online information about the simulation package are

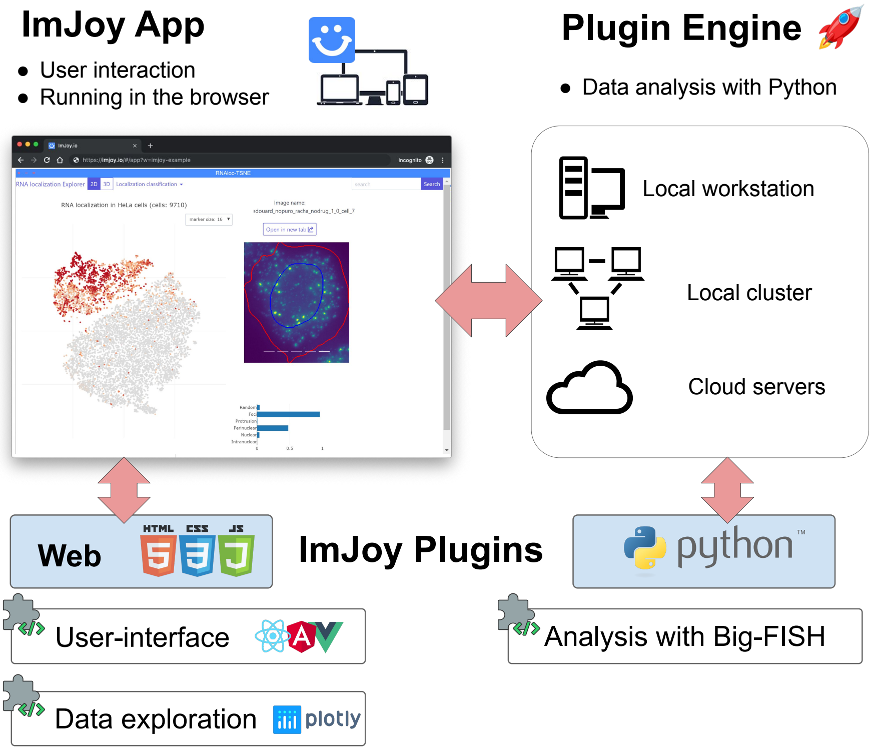

ImJoy App
User interaction
Running in the browser
Plugin Engine
Data analysis with Python
Local workstation
Local cluster
Cloud servers
Web
HTML
CSS
JS
ImJoy Plugins
python
User-interface
Analysis with Big-FISH
Data exploration
plotly

Figure 2.4: Overview of ImJoy. (*Top*) ImJoy's core is a Progressive Web App whose functionalities are provided by plugins run directly in the browser or on a plugin engine (local or remote). Plugin engines support the scientific Python ecosystem and dependency management is handled with Conda. (*Bottom*) Browser plugins are implemented with HTML, CSS or JavaScript. They allow user interaction, data visualization and computation in the browser. Heavier computations can be run from a Python plugin engine before being displayed in the web app

available in the documentation[7] or in the GitHub repository[8].

2.2.4 ImJoy

The *bigfish* package provides flexibility and scalability since its components can be adapted to the specific analysis needs of a given project. However, it requires at least a minimum knowledge of Python to establish a complete workflow by using the provided tutorials. We thus implemented several plugins with a GUI for ImJoy [Ouyang *et al.*(2019a)]. A general organization of ImJoy is illustrated in Figure 2.4. It provides simpler access for users with no computational background and no programming skills. These plugins provide the most commonly used analysis workflow, as determined from the usage of the Matlab version of FISH-quant [Mueller *et al.*(2013)], and will thus be suited for a large number of use cases. It mostly includes segmentation and detection tools.

First, we implemented a plugin to perform segmentation on top of CellPose model [Stringer

---

[7] `https://sim-fish.readthedocs.io/en/stable/`

[8] `https://github.com/fish-quant/sim-fish`

et al.(2021)]. Thanks to the modular design, this model can be easily exchanged if more performant methods are available in the future. A documentation is available online to launch and run the plugin[9].

We also implemented a detection plugin based on *bigfish* modules. Both isolated and clustered RNA can be detected and detection results can be inspected with the Kaibu image viewer plugin in ImJoy. While *bigfish* is developed to be scalable without the need to fine tune parameters, different detection settings can be interactively investigated through the GUI. Batch processing of entire folders is also possible. Lastly, detection results can be assigned to segmented cells and nuclei if segmentation masks are available. An online documentation is available for this plugin[10], and also an interactive demo[11]. This demo can be run directly in the browser without any local installation.

Using ImJoy provides several advantages compared to a stand-alone GUI. Due to its distributed design that separates GUI from computation plugins, it natively supports user-friendly remote computing which allows access to massive data storage and powerful computation resources including GPUs. ImJoy is a Progressive Web App where the user interface plugin is implemented with front-end languages like HTML, CSS or JavaScript. It then transparently calls the *bigfish* methods running on a Python plugin engine (for example a Jupyter notebook server) to perform the actual smFISH analysis task. While this plugin can run on a local workstation, it can be executed on a computational cluster or even in the cloud or seamlessly switching between them. For example in our demo version, the engine is running on Binder [Jupyter *et al.*(2018)]. Once the plugin engine is installed on the remote resource, the end-user can connect with ImJoy and will be confronted with the same interface (the browser plugin), independently of where the analysis is actually performed. Interestingly, this front-end interface can also be opened with mobile devices. ImJoy plugins implemented in JavaScript not only provide modern and reactive user interfaces, but also profit from the extensive JavaScript resources in terms of data visualization and interactivity. Such interactivity is becoming increasingly important, especially when large and complex data sets are analyzed where static plots are too limited. As a case example, we provide an interactive t-SNE plot[12] for the data analyzed in [Chouaib *et al.*(2020)] and detailed in chapter 6. This plugin can be run without local installation and enables the user to explore and interact with these complex data.

2.3 Conclusion

In this chapter I present the second version of FISH-quant, a user-friendly Python based framework for the complete analysis of smFISH images. It is built around *bigfish*, a core-analysis package, implemented following rigorous software development guidelines, with detailed interactive documentation and tutorials. This package consists of several interchangeable modules whose organization matches key steps in smFISH image analysis: preprocessing, RNA detection, cell segmentation and analysis. Its modularity permits the creation of flexible workflows ranging from the analysis of small data sets with the help of a GUI to custom-tailored investigation of large-scale screens requiring computational clusters. Indeed, I also provide user interfaces in ImJoy accessible to biologists without programming skills, which can be used locally or scaled to larger remote computational resources, and displayed in the browser. A last package, *simfish*, allows the simulation of smFISH images with non random RNA localization patterns. These simulations can be used to develop and evaluate analysis pipelines. Lastly, the use of Python scientific ecosystem, as well as strict version control and minimal dependencies, facilitate installation, maintenance and integration with other analysis or visualization frameworks. All dependencies, as well as FISH-quant v2, are open-source, thus can be used free of charge.

---

[9] `https://fq-segmentation.readthedocs.io/en/latest/`

[10] `https://fq-imjoy.readthedocs.io/en/latest/`

[11] `https://fish-quant.github.io/fq-interactive-docs/#/fq-imjoy`

[12] `https://fish-quant.github.io/fq-interactive-docs/#/rnaloc-tsne`

In the chapters 3, 4 and 5, I detail the different functions available in FISH-quant. Throughout the manuscripts, I thus include several snippets with the few lines of code related to the described methods.

3

Single RNA Detection

Abstract:

RNA molecules appear as diffraction limited spots on smFISH images. Their spatial coordinates are extracted using spot detection algorithms and the resulting point cloud can be processed for quantitative analysis. An existing spot detection technique is improved to increase its scalability and evaluated on simulated spot images. In addition, methods for detecting RNA clusters and decomposing these dense areas are provided with FISH-quant. Spot detection can be applied to high content screening studies: no manual intervention is required and the performance is robust enough for different noise levels.

Résumé:

Les molécules d'ARN apparaissent comme des spots fluorescents sur les images smFISH. Leurs coordonnées spatiales sont extraites à l'aide d'algorithmes de détection de spots et le nuage de points qui en résulte peut être utilisé pour une analyse quantitative. Une technique existante de détection de spots est améliorée pour permettre son usage à grand échelle, puis évaluée sur des images de spots simulées. En outre, FISH-quant inclut une solution pour détecter et décomposer les clusters d'ARNs. La détection de spots peut s'appliquer à des études de criblage à haut débit : aucune intervention manuelle n'est nécessaire et la performance reste suffisamment robuste pour différents niveaux de bruit.

Contents

3.1 Spot detection as a signal processing problem 35
  3.1.1 Problem description 35
  3.1.2 Related work 36
3.2 Scaling mRNA detection 38
  3.2.1 Spot detection 38
  3.2.2 Managing high spot density 41
  3.2.3 Going beyond pixel accuracy 43
3.3 Evaluation with simulated spots 45
  3.3.1 Simulations 45
  3.3.2 Results 47
  3.3.3 What if the PSF is not Gaussian? 48
3.4 Conclusion 48

In smFISH images, RNA molecules appear as diffraction limited spots For this reason, the detection of individual RNA molecules boils down to spot detection, a problem that has been largely addressed by different scientific communities.

After a review of different techniques for spot detection, I will discuss shortcomings of existing methods when applied to High Content Screening by FISH and describe the solution I have developed and implemented in *bigfish.detection*. Finally I evaluate robustness and accuracy of this implementation by applying the algorithm on simulated data under different noise conditions. These results are published in the paper [Imbert *et al.*(2022b)]:

A. Imbert, W. Ouyang et al. (2022), *FISH-quant v2: a scalable and modular tool for smFISH image analysis*, RNA, pp. 786–795, iSSN: 1355–8382, 1469–9001.

3.1 Spot detection as a signal processing problem

This introductory section is devoted to a description of the spot detection problem, and a review of solution proposed in the literature.

3.1.1 Problem description

From an image with identifiable point sources, the aim of spot detection is to extract an array of spatial coordinates corresponding to the spot centers. This task is performed in two or three dimensions, depending of the input image.

Detecting an object as small as an RNA molecule faces several challenges:

1. The optical system does not capture the original light signal emitted by the fluorescence probes, but its convolution with a PSF. In this thesis, I will assume that the PSF can be approximately described by a Gaussian in 2 or 3 dimensions. Such simplification is reasonable for a detection with pixel accuracy, given a good SNR.
2. As we can observe in Figure 3.1, a smFISH image often presents a background fluorescence, stemming from reporter molecules that are not specifically binding to target RNA and which are uniformly distributed in the cytoplasm. In 2D images, we thus observe that the background signal is roughly proportional to the local height of the cell.
3. We can observe noise that actually resembles the signal we wish to detect, but at lower intensity.
4. Both signal and noise can be highly heterogeneous and vary between images and between cells inside the same image.
5. Depending on the density and the localization patterns, RNA molecules might be in close proximity. They might form clusters, for which no individual spots are distinguishable. In this case, the detection of individual RNA molecules might not be possible without simplifying assumptions.

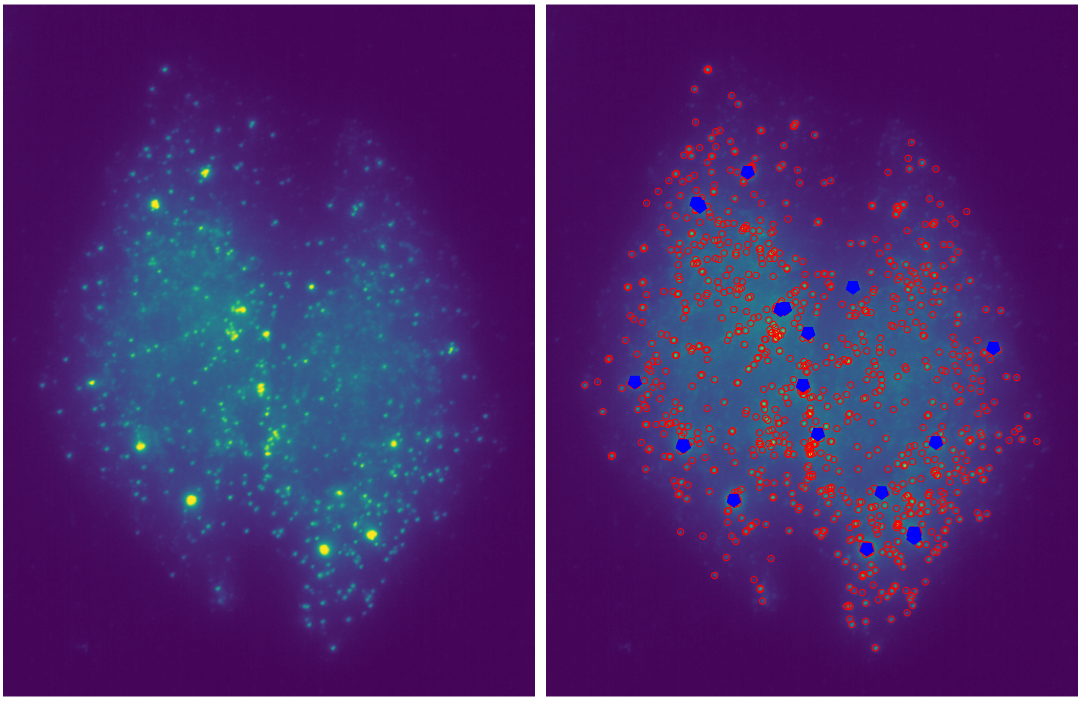


Figure 3.1: Detection results. (*Left*) Original smFISH image. (*Right*) Spots in *red* and clusters in *blue*, detected with *bigfish.detection*

In the context of a high content screening, a large number of images are acquired, with various biological samples or conditions. For this reason, heterogeneity regarding pixel intensity and spot distributions represent a serious problem. At the same time, the number of images is such that we cannot reasonably manually adapt parameters for individual images (let alone individual cells). It thus requires a robust approach to tackle this problem and extract accurate coordinates of the spot positions.

3.1.2 Related work

Traditional approaches based on filtering and thresholding

Spot detection, especially for fluorescent images, has been addressed by the scientific community through different approaches. Classical approaches usually rely on filtering approaches that specifically enhance spot-like features, and binarization by thresholding or maxima detection.

Methods Most spot detection methods can be decomposed into 3 steps: reduction of noise, signal enhancement and signal thresholding [Smal *et al.*(2010)].

A popular method to reduce noise while enhancing spot-like structures is the convolution with a Gaussian filter. Filtering enhances the structures that resemble the filter itself, while attenuating high-frequency noise. Under the assumption of a Gaussian PSF, a Gaussian filter is supposed to be particularly efficient. Alternative methods for noise reduction and signal enhancement include wavelet-based filtering, where the signal is decomposed with the Wavelet transform and non-significant Wavelet coefficients are discarded [Olivo-Marin(2002)].

Signal enhancement refers to the specific enhancement of spot-like structures. As the spots we aim at detecting are small with respect to all other structures in the image, a popular strategy is to remove spots by some filter operation ψ and to consider the residue $g - \psi(g)$ for further processing, where g is the prefiltered image from the first step. Here, ψ might be a Gaussian filter (thus leading to the classical Difference of Gaussians (DoG) filter [Lindeberg(2015)]) or a morphological filter, such as the h-dome image [Smal *et al.*(2010)], or diameter openings [Walter *et al.*(2007)]. An alternative consists in evaluating the second order derivative by calculating the Laplacian of Gaussian (LoG). It can be shown that this approach is closely related to the DoG filter [Lindeberg(2015)].

The third step is binarization. This is usually achieved by either thresholding the enhanced image or identification of local maxima, usually followed by application of additional criteria, such as intensity (which is also a form of thresholding).

These three steps can be complemented by additional post-processing methods to disentangle close spots and refine the positions of the detected spots by PSF fitting or exploitation of radial symmetry [Bahry *et al.*(2021)]. Overall, it can be said that smFISH usually provides high SNR images for cell culture, and differences between these detection methods are often marginal [Smal *et al.*(2010)].

Implementations In general, software for RNA detection can rely on several spot detection algorithms that are already implemented and available in popular Python packages like *scikit-image* [Walt *et al.*(2014)] or *astropy* [Price-Whelan *et al.*(2018)]. Astropy community developed an affiliated package *photutils* [Bradley *et al.*(2020)] with helpful functions for photometry of astronomical sources like PSF matching or source detection. Software tools for RNA detection in Python can thus rely on these libraries.

There are also existing dedicated software solutions for the detection of RNAs, implemented in the three ecosystems used in bioimage informatics: Java, Python and MATLAB. For instance, Icy (implemented in Java), proposes a wavelet-based method for spot detection [de Chaumont *et al.*(2012)], the ImageJ/FIJI plugin Trackmate (also implemented in Java) implements a blob detection pipeline with LoG filters [Ershov *et al.*(2022)], the recently published RS-FISH [Bahry *et al.*(2021)] proposes DoG filters with refined localizations as a FIJI plugin [Schindelin *et al.*(2012)] (implemented in Java). The recent Python library *starfish* [Perkel(2019)] wraps existing *scikit-image* functions and the first FISH-quant version [Mueller *et al.*(2013)] (implemented in MATLAB) includes a blob detection pipeline with a LoG filter. This last pipeline is the one I implemented and improved in my work for *bigfish.detection*.

Besides the processing capabilities and the specific implementations, most of the cited solutions also come with a GUI, namely Icy, Trackmate, FISH-quant and RS-FISH. One important aspect of these GUIs is to manually adapt algorithmic parameters, in particular the detection threshold.

This last point is a major limitation for me. The need for parameter tuning is a critical bottleneck for scaling detection. When we apply an algorithm to thousands of images, with noise and intensity heterogeneity, most parameters need to be set once (or automatically) and not recalibrated between images. In particular most of the presented methods require a threshold or a size parameter at some point.

Learning-based methods

Spot detection has also been addressed by recent deep learning methods. Deep learning (or more generally computer vision), allows one to perform spot detection without the a-priori definition of filters, models and criteria, and prevent parameter tuning on a cell-by-cell or image-by-image basis at the cost of a training stage, thus implying the establishment of ground-truth data.

A first study [Bouilhol *et al.*(2022)] proposes to preprocess images to make spot intensity homogeneous. A convolutional network (named DeepSpot) is trained to enhance spot signal to the same intensity. The network has two main components: a Context Aggregation for Small Objects (CASO) module followed by a customized ResNet [He *et al.*(2016)]. CASO mixes standard convolution blocks with strided and atrous (or dilated) convolutions [Hamaguchi *et al.*(2018)]. Small objects like spots do not contain enough semantic information to be captured. Standard convolution blocks (with max pooling) are great to learn semantic information, but at the expense of lower intensity spots and a potential spatial information loss. One the one hand, replacing the pooling layer with strided convolution compensates the lower intensity. On the other, the use of atrous convolution increases receptive field and thus improves context information with a minimal spatial information loss.

Another model, DeepBlink [Eichenberger *et al.*(2021)], is based on a U-Net architecture [Ronneberger *et al.*(2015)] and directly detects spots. The U-Net component extracts intermediate features. Then it maps the original image into *grid-cells* (small bounding-boxes) for which model predicts the probability of a spot localization and the potential 2D coordinates. Size of the *grid-cell* is critical. With a small size, too many cells might be empty, resulting in an imbalanced dataset for the classification head of the model. On the opposite, with a larger size, one cell could contain multiple spots (for only one prediction per *grid-cell*). This process is directly inspired by the detection model YOLO [Redmon *et al.*(2016)].

3.2 Scaling mRNA detection

I now describe at depth the algorithms currently implemented in *bigfish.detection*.

3.2.1 Spot detection

The method I use is directly adapted from the original version of FISH-quant [Mueller *et al.*(2013)] and the blob detection algorithms [Walt *et al.*(2014)]. Detection is performed in 2D or 3D. Images are filtered in order to increase SNR, then each spot is defined as a local maximum above a specific threshold.

Filtering

I apply a LoG filter (Gaussian filter followed by a Laplacian). This is a two-step algorithm that enhances the spot signals.

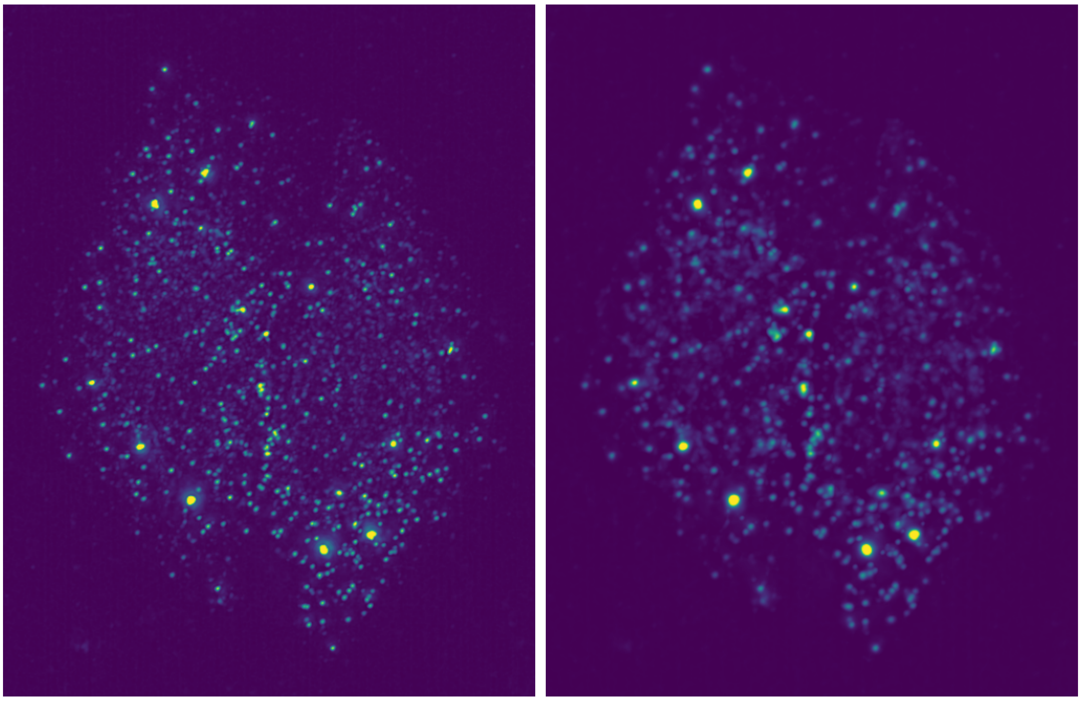


Figure 3.2: Filtered images with LoG filter (*Left*) and DoG filter (*Right*)

The Gaussian filter smooths the image and removes the high frequency noise. By choosing the Gaussian weights, it gives maximal output if the image structure under the filter is perfectly correlated to a Gaussian PSF. Consequently, I apply this operator at a single scale because I assume a unique size for the spots, defined by the optical system. By default, the size of the Gaussian kernel is thus set to match the expected size of the spot which is assumed to be known for a given experiment.

The Laplacian filter approximates the second derivative of the image. Indeed, spots are supposed to correspond to local minima of the second derivative. If I consider a 2D image $f(x, y)$, the LoG filter consists in computing the second derivative of the smoothed image $L(x, y, \sigma^2)$:

$$\nabla^2 L(x, y, \sigma^2) = \frac{\partial^2 L(x, y, \sigma^2)}{\partial x^2} + \frac{\partial^2 L(x, y, \sigma^2)}{\partial y^2} \tag{3.1}$$

with $L(x, y, \sigma^2)$ the convolve image:

$$L(x, y, \sigma^2) = g(x, y, \sigma^2) * f(x, y) \tag{3.2}$$

and $g(x, y, \sigma^2)$ the Gaussian kernel with a scale σ^2:

$$g(x, y, \sigma^2) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{3.3}$$

An alternative filter is the DoG filter. I estimate the background of the image with a large Gaussian kernel, then I subtract it from the original image or one smoothed with a narrower Gaussian kernel. LoG and DoG filters are closely related. As illustrated in

Figure 3.2, both methods aim to remove the background noise and enhance the spot signal.

Peak detection

A Local Maximum detection algorithm follows the filtering. I apply a maximum filter on the LoG-filtered image and compare the result to the original one. A pixel with the same value in the original and filtered images is defined as a local maximum. If, by chance, a spot has several identical pixels at its peak, I only keep one to define the spot coordinate.

Thresholding

At this stage, actual spots and noisy fluorescent blobs (e.g. off-site binding of oligos) are both detected. From all previously detected local peaks, I only keep those above a specific threshold. The problem is how to set this threshold. Manual setting of the threshold does not allow application of the detection method at a large scale, as the signal intensities can be very different between different images. Indeed, while image acquisition can be homogenized, the efficiency of the probes is necessarily heterogeneous. Thus, I use a heuristic technique to set a threshold per image in a automated way.

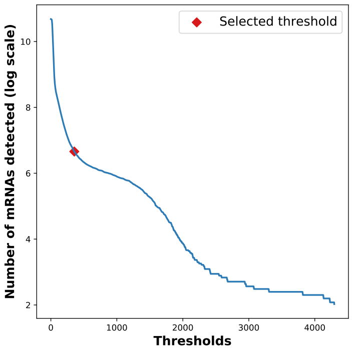


Figure 3.3: Elbow curve

I know that the major discriminative feature between spots originating from mRNA and those from off-site binding of oligos is their intensity: real mRNAs have significantly higher intensity values since an mRNA molecule is targeted by multiple oligos. Conversely, the actual shape and size of true mRNAs and false positives is not necessarily different (see Figure 3.2). In the Figure 3.3, I plot the relation between different thresholds and the number of selected spots (with a log scale). We observe a sharp and monotone decrease in the number of detected spots with increasing detection threshold. The *spots* removed are mostly background noise at these low threshold levels. Actual spots are too bright to be filtered out. At the opposite, if I increase the threshold too much, I start removing real spots and the sensitivity of the detection decreases.

In addition, I assume that wrong detections due to noise are much more frequent than actual mRNAs. This is a reasonable assumption, as off-site binding of a low number of probes is frequent. This means that the decrease in spot number when the threshold increases should be large and relatively steady. As soon as the number of removed spots by a further increase drops dramatically (see red point in Figure 3.3), I can assume that I am reaching a point where I have removed most of the low-intensity noise and start removing spots originating from real mRNAs. I can thus argue that the right threshold value is reached when the reduction in spot number is beyond this first steep decrease and reaches a *plateau*. This plateau is reached when the tangent's slope equals the average slope of the curve (for the first time). Even if this plateau is less pronounced than in Figure 3.3, for instance if there is a particularly high level of noise, an abrupt change in the slope of the curve is still identifiable. This difference of slope describes a clear separation between the regimes of overdetection and underdetection. Additional elbow curves can be observed in appendix B.1, with different conditioning.

```
import bigfish.detection as detection

# spot detection with automated thresholding
spots, threshold = detection.detect_spots(
    images=smfish,
    return_threshold=True,
    voxel_size=(300, 103, 103),  # in nanometer
    spot_radius=(350, 150, 150))  # in nanometer
```

3.2.2 Managing high spot density

A second challenge in spot detection is the presence of clustered spots and high density areas, like active transcription sites or RNA foci. The method described above in 3.2.1 works well with isolated spots. When spots are agglomerated, they might not be possible resolve anymore. In practice, an accumulation of spots looks like a large and bright fluorescent region where my detection will underestimate the number of individual spots.

One option might be to use blob detection at different scales [Walt *et al.*(2014)], which would allow to identify these clusters as one spot. However, I could still not represent them as an agglomeration of individual spots. In *bigfish.detection* I adapt the solution proposed in a previous version of MATLAB FISH-quant [Mueller *et al.*(2013), Samacoits *et al.*(2018)]. I handle high spot density regions in two independent steps:

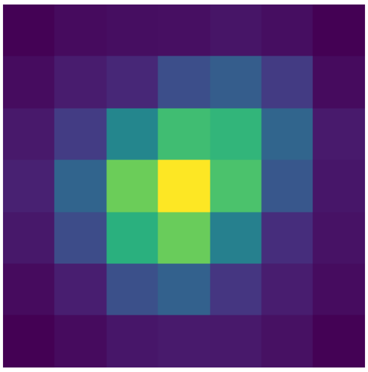


Figure 3.4: Reference spot

- In order to deal with RNA underdetection in dense regions, I identify potential dense regions and decompose them into individual spots. This step increases the number of detected spots in the image (see figure 3.5).
- For subsequent analysis of the presence of clusters, I detect RNA clusters by applying a clustering algorithm to the RNA point cloud. This step can be performed with or without the dense region decomposition.

While somehow related, I stress that the first step tackles a technical issue of underdetection, while the second step allows me to separately define RNA clusters for further analysis, as the number of clusters is an important feature of RNA localization.

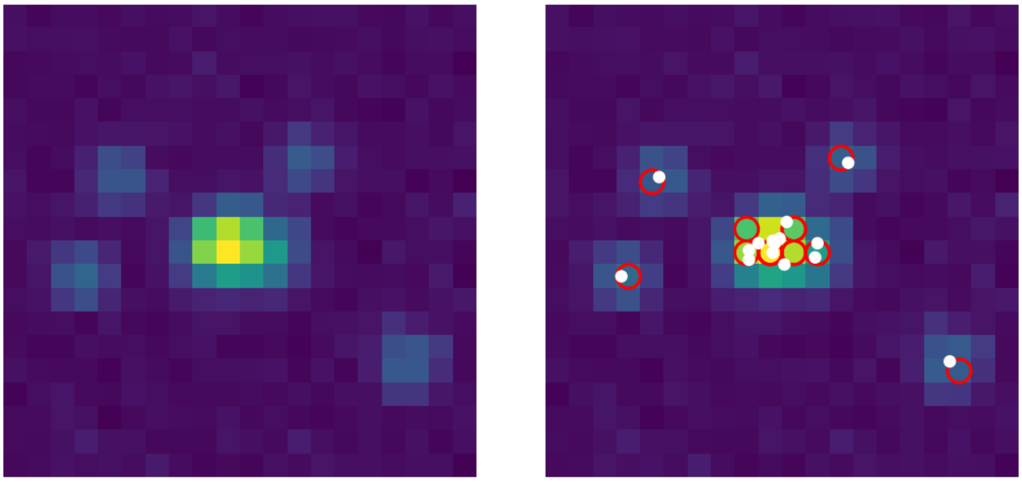


Figure 3.5: Decomposition results. (*Left*) Original smFISH image zoomed in a dense region. (*Right*) Detected spots in *red* and actual clustered spots in *white*

Dense region detection

The first step consists in localizing regions in the image with a high density of spots. I know that for such regions my detection might miss several spots.

I remove the low-frequency noise from the image by subtracting its background intensity. The latter is approximated with a large Gaussian filtering. From this image I then extract the detected spots and compute the median spot signal. This median signal allows me to define criteria for the identification of high density regions. I expect high density regions to be brighter than individual spots, so they should at least be brighter than the median spot intensity. A second criterion is the size of the regions. Furthermore, they should be larger than an individual spot. To match these criteria, I first threshold the denoised image with the median spot intensity, then I apply a connected component algorithm [Wu *et al.*(2005)] to the binary mask obtained. Each group of connected pixels represents a region. Because the mask is the result of a thresholding above the median spot signal, every region (or connected component) with at least 2 pixels are larger and brighter than the median spot for this image.

Dense region decomposition

The candidate regions can contain one individual RNA spot brighter than the average, but they can also contain one large spot, corresponding to an agglomeration of very close RNAs. I reuse the denoised image and aggregate the detected spots to compute a reference spot like in Figure 3.4. By default this reference is the median spot, but another percentile can be chosen. I fit a Gaussian signal on the reference spot. This model can then be used to simulate new spots.

The decomposition process consists in populating my candidate regions by simulating as many spots as possible until I match the pixel intensity observed in the region. Starting with an empty image, I iteratively add a new simulated spot in the region until I minimize the residual sum of square (RSS):

$$\text{RSS} = \sum_{x,y} (\hat{f}(x,y) - f(x,y))^2 \tag{3.4}$$

with $\hat{f}(x,y)$ the simulated image intensity and $f(x,y)$ the denoised image.

Cluster detection

The second (independent) step consists in applying a clustering algorithm on the spatial positions, in order to identify clusters according to a clearly understandable, biological meaningful definition, only based on spatial coordinates.

To this end I use a DBSCAN algorithm [Ester *et al.*(1996),Pedregosa *et al.*(2011)]. Two parameters need to be set: a minimum number of spots k and a threshold distance d. Every pair of RNAs closer than d are connected. If an RNA is connected to at least k neighbors RNA, it's a *core sample* and with its connections it defines a cluster. Such a method allows me to detect clusters as "areas of high density separated by areas of low density"[1].

Different users might may have a different definition of what they expect to be a cluster. For the rest of the manuscript and throughout my studies, I usually consider a minimum group of 4 or 5 RNAs within a radius of 350nm. These are the default parameters in *bigfish.detection* and the ones I use in Figure 3.1.

```python
import bigfish.detection as detection

# dense decomposition
spots_post_decomposition, _, _ = detection.decompose_dense(
    image=smfish,
    spots=spots,
    voxel_size=(300, 103, 103),  # in nanometer
    spot_radius=(350, 150, 150))  # in nanometer

# cluster detection
spots_post_clustering, clusters = detection.detect_clusters(
    spots=spots_post_decomposition,
    voxel_size=(300, 103, 103),  # in nanometer
    radius=350,  # in nanometer
    nb_min_spots=4)
```

3.2.3 Going beyond pixel accuracy

Two additional methods for reaching subpixel accuracy have been integrated in FISH-quant. They were already present in the first version of FISH-quant [Mueller *et al.*(2013)].

Subpixel fitting

So far, the spot detection and the dense region decomposition and the cluster detection return coordinates with pixel accuracy. In my applications, I was only interested in overall RNA distribution, but for some applications, subpixel accuracy is critical. Such error can be observed in the Figure 3.5 between the detected spots (with pixel accuracy) and the ground truth (with subpixel accuracy).

---

[1] `https://scikit-learn.org/stable/modules/clustering.html`

The possibility to refine the coordinate on individual spots solves this limitation. For this, I loop over the detected spots, crop the image and fit a gaussian signal on each of them individually. I then correct the spot coordinates with the coordinates of the fitted gaussian. Obviously, such method might return inaccurate results in high density areas when spots can't be resolved.

```
import bigfish.detection as detection

# subpixel fitting
spots_subpixel = detection.fit_subpixel(
    image=smfish,
    spots=spots,
    voxel_size=(300, 103, 103),  # in nanometer
    spot_radius=(350, 150, 150))  # in nanometer
```

Spot colocalization

Another method requested by the community is the possibility to detect adjacent spots in different channels then match their coordinates. This could be the same RNA detected with two different fluorescent probes or techniques in order to validate experimental protocols, or different RNA for which I would like to investigate the co-localization. As an example, in the Figure 3.6, I detect colocalized spots between a sample of spots detected with pixel accuracy and the same sample with subpixel accuracy.

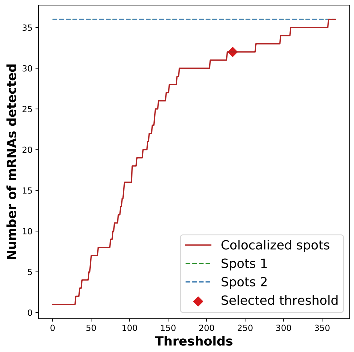


Figure 3.6: Impact of threshold on colocalization

My implementation is based on the methods published by [Cornes *et al.*(2022)]. First I compute the euclidean distance matrix between the two sets of spot coordinates, then I solve a linear sum assignment problem [Crouse(2016), Virtanen *et al.*(2020)]. I obtain a matching between the two sets of spots, that minimizes the overall euclidean distance between assigned pairs. Finally, I only keep pairs with a distance below a specific threshold. The Figure 3.6 illustrates the impact of the threshold parameter on the number of colocalized spots. Like for the spot detection, I implement my heuristic 3.2.1 to infer an optimal threshold if none is provided.

```
import bigfish.multistack as multistack

# spot colocalization
(spots_1_colocalized, spots_2_colocalized,
 distances) = multistack.detect_spots_colocalization(
    spots_1=spots_crop,
    spots_2=spots_subpixel_crop,
    voxel_size=(300, 103, 103))  # in nanometer
```

3.3 Evaluation with simulated spots

Finally I describe my evaluation of *bigfish.detection*. In addition to qualitative assessment of spot detection throughout my studies, I quantify the error from simulated images. Performances for both spot and cluster detections are measured with different image qualities.

3.3.1 Simulations

To measure the error of a spot detection I need a ground truth. A manual annotation of a regular 3D smFISH image is intractable: such a strategy would be extremely time-consuming and prone to human error. The alternative is to simulate realistic images of spots under different noise conditions in order to assess the performance of my algorithms and to study its limitations depending on image quality. To this end, I built the simulation package *simfish* that allows me to precisely control the level of noise and the number of spots I want to simulate in the image.

Spot simulation

The simulation process aims to return both an image and the ground truth coordinates of the spots I simulated.

My images are generated with three main steps:

1. I randomly draw the number of spots and their localization. This is my ground truth. The number of spots is sampled from a Poisson distribution and the localizations from a uniform distribution all over the frame. Alternatively the number of spots can be set manually.

2. For each spot in the image I simulate its pixel intensity. Instead of directly sample the intensity value from a Gaussian distribution, I reuse the simulation process from [Bahry *et al.*(2021)]. With a Gaussian distribution centered on every spot, I simulate the average number of photons collected by each pixel in the image. The amplitude and the standard deviation of this Gaussian signals are manually or randomly predetermined. The final intensity of every pixel is then sampled from a Poisson distribution with the number of photons as expectation.

3. (Optional) I add a background white noise to the entire image. It follows a Normal distribution centered around a predefined noise level.

The process detailed above generate spots with pixel accuracy. A simulation with a subpixel accuracy follows the same steps, but with a larger image (4 to 20 times larger). Before saving the image and the ground truth, it is downsized by local averaging. The ground truth coordinates are adapted accordingly.

In order to produce a noisier image I can decrease the ratio between the spot amplitudes and the background noise. A second option is to increase the variance of the different parameters: spot standard deviation and amplitude, background noise standard deviations.

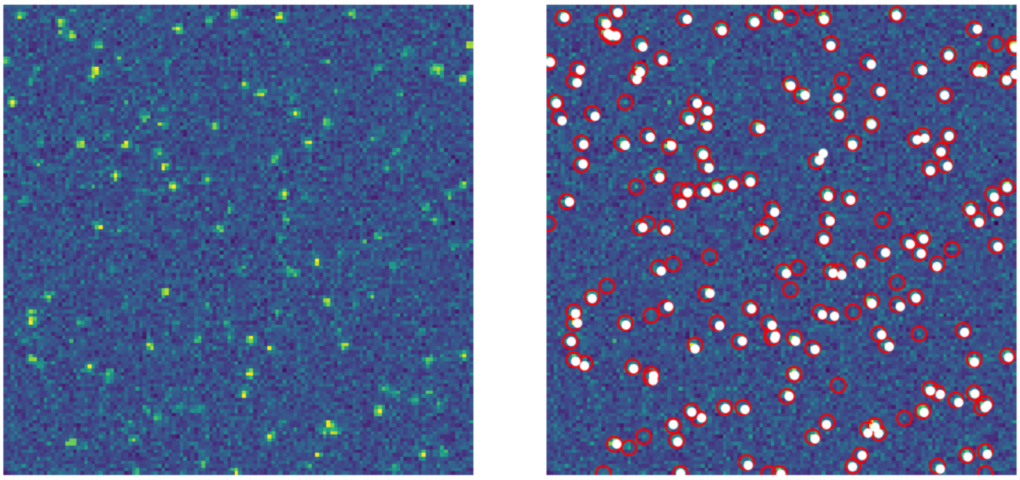


Figure 3.7: Simulation of noisy spots with *simfish*. (*Left*) Original simulated image. (*Right*) Detected spots in *red* and actual spots in *white*

Signal-to-Noise Ratio

I can tune the different parameters mentioned to simulate spots with an increasing level of difficulty to detect. To quantify the noise of an image and graduate the challenge it offers in terms of detection, I compute its SNR for every spot.
For a 2D image, I define the SNR of an image as the median of the $\mathrm{SNR_i}$ I compute for each spot i, such that:

$$\mathrm{SNR_i} = \frac{\max(a(x,y)) - \mu(b(x,y))}{\sigma(b(x,y))} \tag{3.5}$$

with $\mu(.)$ the mean, $\sigma(.)$ the standard deviation, $a(x,y)$ the cropped spot image and $b(x,y)$ the spot background (a crop twice larger than the spot image).

In consequence, my measure of noise is based on the spot coordinates. I quantify how distinct the spot is from its background. A spot with a low amplitude or a noisy background will decrease the SNR of the image. To correctly quantify the image noise during my evaluation, I use the ground truth coordinates of the spots to compute the SNR. Indeed, a noisy image would impact the detection and bias the measure of noise itself. I simulate different images with a range of SNR between 2 and 26. The higher the SNR is, the better.

For example, in Figure 3.7 I simulate an extreme case with a highly noisy image (SNR below 5). On the right panel we can observe some contrasted background blobs misdetected as spot. Despite a small amount of false positives, the detection remains correct in this badly conditioned image.

Cluster simulation

The user can also decide to simulate clusters. In this case, the first step of my simulation process is adapted. First the number of clusters and the number of spots per cluster are drawn from a Poisson distribution (or manually set). The cluster centers are then localized like spots, with a uniform distribution over the entire image, or fixed at the center of the frame. Finally, different spot localizations are randomly generated around the centers, using polar coordinates. The result can be observed in Figure 3.5 which is actually a simulated cluster. In addition, several simulations are available in appendix A.1, with different conditioning.

```
import simfish as sim

# image simulation
image, ground_truth = sim.simulate_image(
    ndim=3,
    n_spots=100,
    image_shape=(128, 128),
    voxel_size=(100, 100, 100),  # in nanometer
    sigma=(150, 150, 150),  # in nanometer
    amplitude=5000,
    noise_level=300)
```

3.3.2 Results

I simulated batches of 100 images with high, medium and low noise levels (roughly, with a SNR below 5, between 5 and 15 and above 15).

Impact of noise

My method is overall pretty robust. If the image quality deteriorates, with a lower SNR for example, my algorithms can return a moderate overestimation of detected spots. This overestimation is estimated below 5% and 10% for images with a low or medium SNR value, respectively.

These measures are illustrated on the left panel of Figure 3.8. I compared the number of spots detected with the actual number of spots simulated. Each dot corresponds to one image. For each noise regime, 100 images are generated. Above the plain line, the detection overestimates the number of spots. Logically, these overestimations increase with the noise level (and decrease with increasing SNR).

On the right panel, I summarize the impact of noise with my detection technique. Each dot represents again an image, with 100 simulated spots and a varying level of noise. Below a SNR of 5, with poorly contrasted spots, a fully automated detection might remain challenging.

Accuracy of the cluster detection

I simulate images with a unique cluster in order to test the performance of my method to decompose the dense regions and detect the clusters. Such images can be observed in appendix A.1. In Figure 3.9 I report the number of spots I estimate in the clusters. Each dot represents an image with a cluster. My decomposition is quite robust with the noise level, even with the lower SNR values. However, for the largest clusters, with 15 spots or more, I tend to slightly underestimate the number of spots. In this case, the average error is 1.6. In comparison, with 5 spots and 10 spots per cluster, I measure an average error of 0.83 and 0.82, respectively.

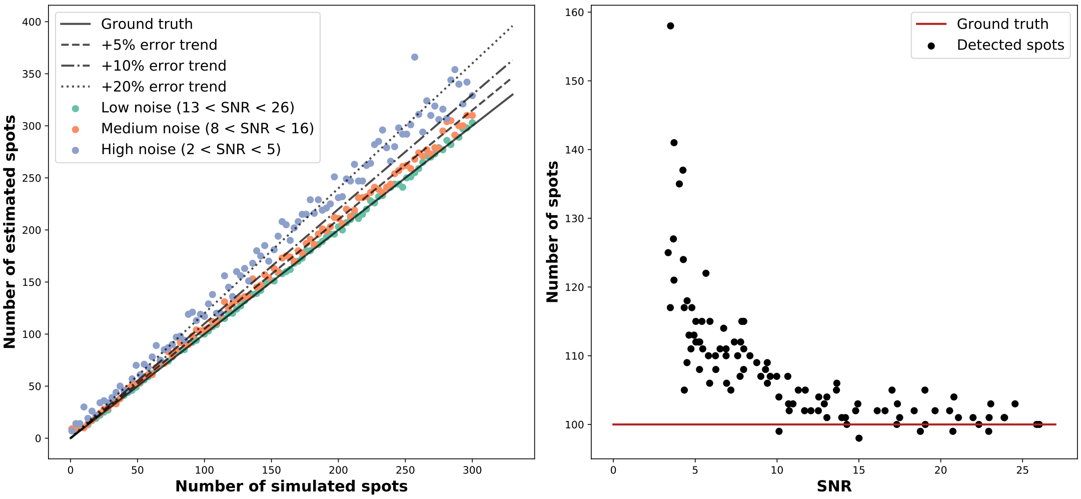


Figure 3.8: Impact of noise on automated detection. (*Left*) Number of detected spots for different numbers of simulated spots. (*Right*) Number of detected spots for different levels of noise (with 100 simulated spots)

3.3.3 What if the PSF is not Gaussian?

In this chapter I assume my mRNA spots can be modelled with a Gaussian signal. This simplification was relevant during my PhD and did not lead to exaggerated errors in my different analysis. However, it is always possible to exploit a more complex PSF to fit with a specific spot signal. To this end, the former *psf* package[2] allows to generate more complex PSF for specific fluorescent microscopic experiments. Letting the user choose between different PSF is an improvement that could be implemented in a future version of *bigfish.detection* and *simfish*. In particular, this would modify the way I generate spot signal in my image simulations or in the decomposition of dense regions.

Eventually, if the PSF differs greatly from a Gaussian signal, the spot detection itself could be impacted. Indeed, two PSFs from close spots could interfere and make the local peaks less distinct.

3.4 Conclusion

This chapter presents different methods from *bigfish.detection* subpackage to perform spot detection in smFISH images. For this, I have re-implemented and extended algorithms first published in FISH-quant [Mueller *et al.*(2013)]. In particular, I have added a new scheme for thresholding that allows me to perform analysis of FISH images without any manual intervention — a pre-requisite for application at large scale. Furthermore, I have provided methods and implementations for cluster detection and decomposition. And finally, I have provided a framework for smFISH image simulation in order to assess the performance of the various detection algorithms under different noise conditions.

Performance assessment of my methods for spot and cluster detection gave overall

---

[2] `https://github.com/cgohlke/psf/`

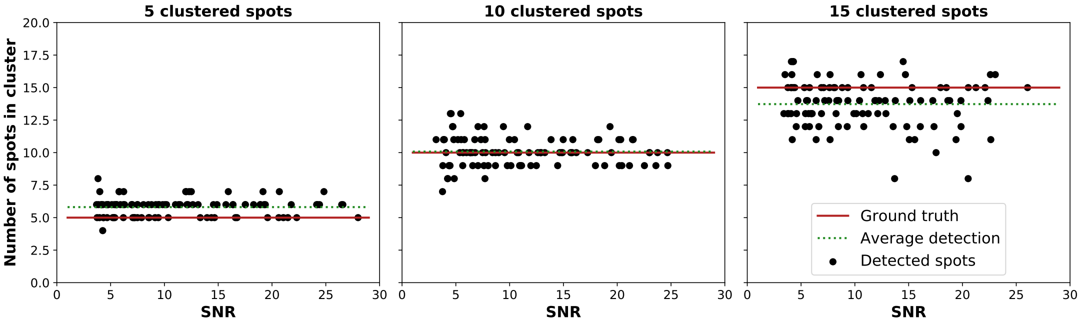


Figure 3.9: Evaluation of dense region decomposition. Number of spots estimated per cluster for different levels of noise and respectively 5 (*left*), 10 (*center*) or 15 (*right*) simulated spots per cluster

satisfying results, even for higher noise levels. One limitation of the performance assessment I provide is the fact that it is based only on simulations. Simulations clearly have many advantages for performance assessment, but also the inconvenience that they might not sufficiently represent domain shifts present in real data. On the other hand, manual annotation is not a feasible alternative for this. First, it is extremely time-consuming, and — more importantly — human annotators are also not necessarily in a position to decide whether a spot is a true mRNA or not. Another alternative that might provide additional insights and that I have not provided in this thesis, is a double-color FISH against the same mRNA. However, also this assessment strategy is not perfect, as there is no guarantee that a given mRNA will necessarily be marked in both channels.

With all the imperfections, I can nevertheless conclude that the performance of my detection algorithms is robust enough to be applied on a high content screening experiment. Of course, several improvements are still possible. I can expand the PSF options as explained in 3.3.3. Some alternative detection techniques that were published recently claim a better accuracy or faster runs like [Bahry *et al.*(2021), Bouilhol *et al.*(2022)]. However, I feel that today, it is difficult to conclude on this, as performance assessment is not standardized, and the field is clearly lacking a general benchmarking strategy. But in general, it would be another important step to provide Python implementations of these methods, and integrate them in FISH-quant v2.

4

Single-cell segmentation

Abstract:

Current solutions for nucleus and cell segmentation are mainly based on deep learning models. FISH-quant includes such pre-trained models, in addition to postprocessing algorithms to format segmentation masks and refine them. It also allows the integration of external and potentially more specialized resources. Two main limitations are then discussed. First, the need for a large training dataset, which can be mitigated by the use of in-silico techniques. Second, the lake of consistency between nucleus and cell segmentation. This consistency can be enforced by the way an algorithm operates, as in the case of a watershed transformation, or learned through a specific training strategy.

Résumé:

Les solutions actuelles pour la segmentation des noyaux et des cellules sont principalement basées sur des modèles d'apprentissage profond. FISH-quant inclut de tels modèles pré-entraînés, en plus d'algorithmes de post-traitement pour formater les masques de segmentation et les affiner. Il permet également l'intégration de ressources externes et potentiellement plus spécialisées. Deux limites principales sont ensuite discutées. Premièrement, la nécessité d'un grand ensemble de données d'entraînement, limite qui peut être atténuée par l'utilisation de techniques in-silico. Deuxièmement, le manque de cohérence entre la segmentation des noyaux et des cellules. Cette cohérence peut être imposée par le mode de fonctionnement d'un algorithme, comme dans le cas d'une segmentation par ligne de partage des eaux, ou apprise par le biais d'une stratégie d'entraînement spécifique.

Contents

4.1 Segmentation of nuclei and cells in fluorescence microscopy . . 53
4.1.1 Computer Vision for cell segmentation 53
4.1.2 Related work 54
4.2 Segmentation models 57
4.2.1 A new multichannel dataset 57
4.2.2 Nucleus segmentation 57
4.2.3 Cell segmentation 60
4.3 Improving cell segmentation 62
4.3.1 Snake-like model 63
4.3.2 In silico pre-training 63
4.4 Conclusion 64

In this chapter, I review different techniques for nucleus and cell segmentation. Recently, the solutions proposed to solve this problem have considerably improved, driven by the wave of deep learning models.

After a brief review of the literature in the first part, I describe the methods implemented in *bigfish.segmentation* in a second part. These methods are published in the paper [Imbert *et al.*(2022b)]:

A. Imbert, W. Ouyang et al. (2022), *FISH-quant v2: a scalable and modular tool for smFISH image analysis*, RNA, pp. 786–795, iSSN: 1355–8382, 1469–9001.

In a third part, I briefly present two projects for which I have contributed with the objective to boost segmentation efficiency, either by improving the consistency of segmentation masks or by reducing the number of training examples needed. The former is a preliminary work with incomplete results and the latter was published at the ECCV Bioimage Computing Workshop (BIC) [Bonte *et al.*(2022)]:

T. Bonte, M. Philbert et al. (2022), *Learning with minimal effort: leveraging in silico labeling for cell and nucleus segmentation*, in 2022 European Conference on Computer Vision (ECCV 2022) Workshop on BioImage Computing.

4.1 Segmentation of nuclei and cells in fluorescence microscopy

First, I describe the segmentation task that comes in different flavors. Then, I review different methods that address nucleus and cell segmentation, especially deep learning based models.

4.1.1 Computer Vision for cell segmentation

The most useful tasks of Computer Vision in bioimage analysis are classification, object detection and object segmentation. Classification is concerned with predicting a label for a given input image. Object detection aims at both classifying and localizing objects inside an image, where localization normally amounts to returning the coordinates of the bounding box of the object. Of note, object detection is designed to detect several instances of the same object class. Finally, a segmentation model returns returns a mask for the targeted objects. If the model does not distinguish between instances of the same object class, it is a semantic segmentation model. Semantic segmentation resumes to pixel classification, where each individual pixel is classified into object or background (or potentially different foreground classes). In contrast, the most important segmentation problem in bioimage analysis is instance segmentation, where the model not only distinguishes between foreground and background but also between different objects. This kind of models is of particular interest for cell and nuclei segmentation, as both cells and nuclei often touch each other.

An example of instance segmentation with nuclei and cells is shown in Figure 4.1. A unique identifier and segmentation mask are returned for every instance. In Figure 4.1,

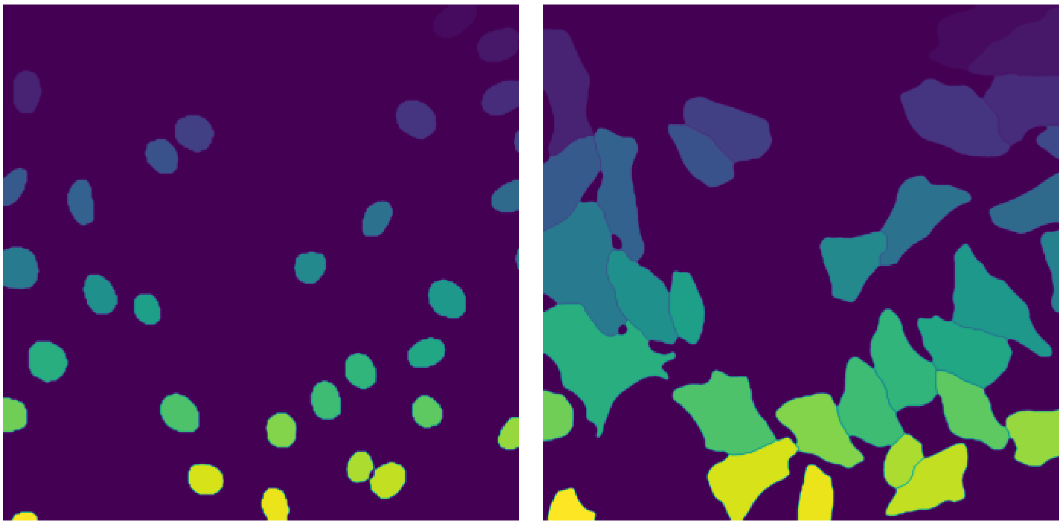


Figure 4.1: Example of instance segmentation. (*Left*) Nucleus masks. (*Right*) Cell masks. Every nucleus and cell instance has a different colored mask. Plot built with *bigfish*

segmentation masks have been postprocessed such that a nucleus has the same identifier as its corresponding cell.

For the rest of the chapter, I only consider 2D segmentation. Models for 3D segmentation are emerging in the literature, but as manual annotation in 3D is complex and time-consuming, it does not seem an interesting option for our applications.

A specific difficulty with instance segmentation, compared to semantic segmentation, is the need to discriminate between two adjacent instances. In case of cell segmentation, this is particularly important. For instance, if we wish to quantify cell shape properties or assess the number of RNA inside a cell, it is essential to be able to extract the contours of the cells with high accuracy.

Even though it might be tempting in some applications to omit the cell segmentation step altogether and to classify images of entire cell populations [Godinez *et al.*(2017)], I believe that cell and nucleus segmentation add great value to the analysis pipelines as they enable one to assign every phenotype or pattern detected in an image to an individual cell. Segmentation is thus the corner stone of any single-cell analysis. Instance segmentation is a critical step to capture information at the cell level, especially when the biological mechanism studied exhibits a high intracellular heterogeneity.

4.1.2 Related work

From mathematical morphology...

Fluorescence Microscopy has been specifically designed to provide images, where structures of interest appear with high contrast, while irrelevant structures are invisible. This can greatly simplify the segmentation task, in particular when objects are isolated and can be easily discriminated from the background, for instance DAPI stained nuclei. Therefore, a first approach, simple but often successful, is to threshold the image to discriminate foreground from background, and then identify each disconnected mask with a unique identifier to label the objects. To avoid manual setting of the threshold for each image, automatic methods based on histogram intensity analysis can be exploited, such as Otsu thresholding [Otsu(1979)]. For clustered nuclei or images with crowded cells, this method does not work and more complex algorithms are needed.

A popular method for segmentation of nuclei and cells is the watershed algorithm [Beucher et Lantuéjoul(1979), Serra(1983), Vincent et Soille(1991)]. An image is interpreted as a local topography (higher values are peaks and crests, lower values valleys and basins). There are several variants of this method, but basically the algorithm requires three elements: the starting points from which we flood the image (the markers or seeds; the local minima as default), a relevant topographic representation of our image where the boundaries of the object have higher pixel values, and optionally a binary mask limiting the flooding area (the mask). The Watershed algorithm simulates a flooding of the surface where pixels are subsequently added to the extending seed regions in such a way that pixels with lower value are always processed first. Object boundaries are found where the extending seed regions meet.

This is an attractive strategy for cell segmentation [Wählby *et al.*(2002)]. We can use the previously detected nuclei as seeds, which guarantees that all detected cells will have one nucleus, which is inside the cellular region, by design of the algorithm. Furthermore, there are numerous possibilities to construct an image to be flooded, depending on the employed marker: either the image itself (e.g. in case of a membrane marker) is used, a gradient image (e.g. in case of a cytoplasmic marker with different expression levels) or the distance map of a binary image (which was the original idea). Unfortunately, if we miss some nuclei at the beginning, the error impacts the rest of the computational pipeline. Not only do we miss potentially interesting cells, but we also wrongly assign cytoplasmic regions to cells where they do not belong. For this reason, it is important that the nuclei detection returns accurate results.

Many other segmentation methods have been proposed in the computer vision literature. Superpixel approaches, for instance, partition the image into multiple homogeneous regions with enforced compactness [Ren et Malik(2003)]. Low-level segmentation algorithms such as watershed itself can help form these regions by clustering pixels together [Machairas *et al.*(2014)]. These superpixel can then be merged by heuristics or graph-based approaches. The segmentation task can also be addressed from the boundaries point of view. Active contour models (or snakes) rely on energy minimization techniques to deform a spline curve until it delineates the object outline [Kass *et al.*(1988)]. The advantage of this method is that a priori knowledge on the cell or nuclear geometry can be incorporated into the segmentation algorithm [Dufour *et al.*(2005)].

For most segmentation benchmarks, these methods have been outdated in recent years, with the emergence of deep learning models trained on large and diverse datasets. However, mathematical morphology methods remain relevant for some use cases. Most importantly, these traditional methods do not require manual annotation, which is often a bottleneck for medical or biological image segmentation. Eventually, such algorithms can also inspire new learning-based models, like Deep watershed model that predicts a watershed energy map before extracting object instances from it [Bai et Urtasun(2017)].

. . . to deep learning models

Deep learning literature for image segmentation has provided several powerful and consistent models. Mostly based on Convolutional Neural Network (CNN), they have dramatically boosted computer vision applications. This trend keeps influencing bioimage informatics as well.

One of the seminal neural networks proposed for biomedical segmentation is U-Net [Ronneberger *et al.*(2015)]. The network has a U-shaped architecture with an encoder and a decoder. The former combines convolutional layers, non linear activation functions (ReLU) and max pooling operations to reduce spatial information and expand feature information. The latter includes upsampling layers to return an output segmentation map that is postprocessed in order to build a (semantic) segmentation mask. This architecture is now a classic and it is vastly reused by more recent work like StarDist [Schmidt *et al.*(2018)], which is trained to predict star-convex polygons for each nucleus or cell instance.

One the main limitations of deep learning methods in general is the need for a large amount of annotated images to train the models. For image classification, this problem was addressed by the publication of large annotated dataset of natural images like ImageNet [Deng *et al.*(2009)] or COCO dataset [Lin *et al.*(2014)]. Following this example, recent publications in biomedical segmentation include sometimes both a trained model and a release of an important dataset with segmented nuclei or cells. For example, the research community benefits from an online challenge organized in 2018 about nucleus segmentation: the 2018 Data Science Bowl[1]. This competition includes a large collection of images with different modalities (histopathology, fluorescence microscopy).

NucleAIzer [Hollandi *et al.*(2020)] proposes a model inspired by the winning solutions of the 2018 challenge and trained on the released dataset. It combines Mask R-CNN [He *et al.*(2017)] and U-Net. Post competition, it outperforms all the submitted methods of the competition.

Cellpose [Stringer *et al.*(2021)] proposes a U-Net based model for nucleus and cell segmentation. The network is trained to predict horizontal and vertical gradients of the topological cell map. These gradients form a vector field where each pixel "belonging to a given cell can be routed to its center". By grouping pixels converging to the same regions, individual instances can be identified. Most importantly, authors have collected and manually annotated 608 cell images with various modalities. Currently, this solution appears to be one of the most efficient, even though it remains unclear whether this is attributed to the dataset, the architecture or the formulation of the prediction task. The authors have extended their method by training it with several specialized datasets like TissueNet [Greenwald *et al.*(2022)] or bacteria images [Cutler *et al.*(2022)].

Another important factor for progress in segmentation is the state of the literature on natural image segmentation. Indeed, several methods published for biomedical segmentations are inspired from a previous model trained on natural images. For example, NucleAIzer reuses Mask R-CNN, a CNN that combines ideas from object detection and image segmentation. Mask R-CNN suggests regions of interest, detects object instances within these regions and performs instance segmentation with a fully convolutional branch. When released in 2017, the model was state-of-the-art on the COCO dataset. This relationship can also be observed with EmbedSeg [Lalit *et al.*(2021)] directly inspired by [Neven *et al.*(2019)].

---

[1] `https://www.kaggle.com/c/data-science-bowl-2018`

4.2 Segmentation models

In this section, I first describe a dataset I have annotated in order to train deep learning segmentation models. Then, I detail the solutions implemented in *bigfish.segmentation* to segment both nuclei and cells. In addition, several postprocessing functions are presented to refine any segmentation results.

4.2.1 A new multichannel dataset

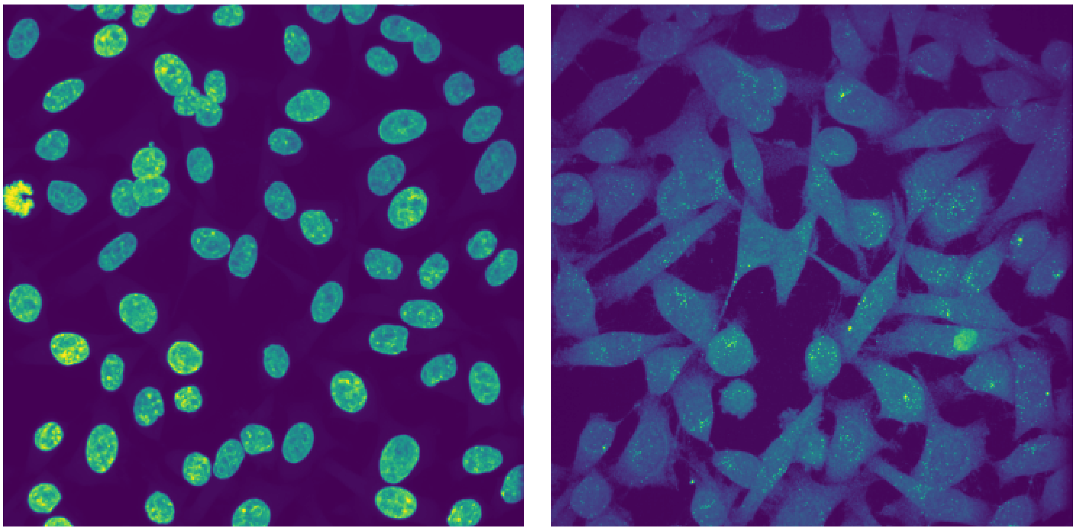
I use the 4-channel images from one of our applications [Safieddine *et al.*(2021)] presented in Chapter 6 to build an annotated dataset for the segmentation task. I randomly sample 180 Field of View (FoV)s. Each image includes one channel adapted for nucleus segmentation (DAPI) and three channels adapted for cell segmentation (smFISH, CellMask™ and a Green Fluorescent Protein (GFP) marker for the centrosome). An example of these images is illustrated in Figure 4.2. The ground truth annotation is obtained in two steps: I pre-segment nuclei and cell with a Cellpose model [Stringer *et al.*(2021)], then I manually correct these predictions. In total, 4,026 cell instances are segmented, with their relative nucleus.

The initial goal for this dataset is to train segmentation models and perform some experiments to evaluate their consistency. In particular, I am interested in:

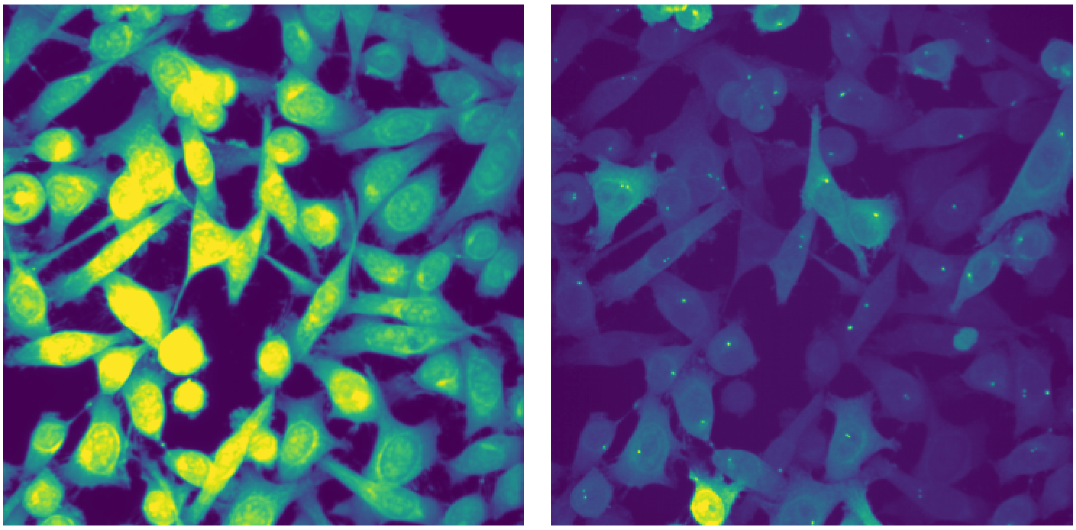
1. Consistency between nucleus and cell segmentation, two tasks often performed independently. The dataset is annotated such that each instance has a nucleus and a cell mask, matching together. This also enables a potential joint training of nucleus and cell segmentation.
2. Input heterogeneity. Here, the cell related channels come from three different modalities of acquisition. A priori, a model trained on CellMask™ should be more efficient, but because this label is not always available in an experimental setup, it is interesting to train models robust enough to identify cells from different channels.

4.2.2 Nucleus segmentation

Nucleus segmentation is usually the first task in cellular image analysis, applied on a DAPI channel for example. An error during this step can propagate to the rest of the analysis, if the cell segmentation and identification is based on an initial nucleus segmentation. In particular, it is important to avoid missed nuclei and nuclei that are split in several parts. In *bigfish*, I have implemented simple thresholding techniques, and a simple deep learning model for nucleus segmentation.



(a) DAPI channel (c) smFISH channel



(b) CellMask™ channel (d) GFP channel

Figure 4.2: Multichannel annotated images for nucleus and cell segmentation. Images are projected in 2D. Plot built with *bigfish*

A 3-class deep learning model

The model is described in [Imbert *et al.*(2022b)]. It uses an encoder-decoder architecture like U-Net [Ronneberger *et al.*(2015)], with 4 downsampling stages. In total, the spatial resolution of the input image is divided by 16 at the bottom of the model. For each stage, I implement residual blocks, mimicking Cellpose model [Stringer *et al.*(2021)]. Lastly, the upsampling stage is implemented following deconvolution techniques from [Odena *et al.*(2016)] to prevent any checkerboard artifacts.

The nucleus segmentation problem is address like a pixel-wise classification problem with three classes: background, foreground and nuclear boundary. My model assigns one of these three classes to each pixel. The returned final mask is the foreground predicted surface, postprocessed with a dilation of 1 pixel. This model is trained on the DAPI channel from the annotated dataset presented in 4.2.1, with a categorical cross-entropy loss.

```
import bigfish.segmentation as segmentation

# load pretrained model
model_nuc = segmentation.unet_3_classes_nuc()

# instance segmentation
nuc_label = segmentation.apply_unet_3_classes(
    model=model_nuc,
    image=image_nuc,
    target_size=256,
    test_time_augmentation=True)
```

Multiple rounds of segmentation

Despite overall good results, I noted that some (dim) nuclei are missed altogether by the algorithm. In *bigfish.segmentation*, I implement a method to remove segmented nuclei from a DAPI channel, in order to perform a second round of segmentation. This strategy is useful if the employed segmentation method misses too many nuclei at first try. Such failures can be due to a heterogeneous DAPI signal between cells, to the presence of adjacent nuclei or instances with unusual shape.

Removal of segmented nuclei is based on morphological reconstruction and based on the following steps:

1. The binary mask of the segmented nuclei is dilated.
2. From the original DAPI image f, an image g is generated where all pixels outside of the dilated mask are set to zero. This includes the background and the potentially missed nuclei.
3. A morphological reconstruction $R_f^\delta(g)$ by dilation [Serra(1983),Soille(2003),Robinson et Whelan(2004)] of g under f allows to reconstruct the background signal, but not the missed nuclei. As a consequence the reconstructed image only differs from the original one where the nuclei have been missed in the first place.
4. The residue image $\Delta = f - R_f^\delta(g)$ thus contains the missed nuclei and can then be thresholded to complete the detected nuclei.

Finally, after a second round of segmentation, the two obtained nucleus segmentation masks can be merged together. Surprisingly, repetitive application of the same segmentation model seems to help improving the final segmentation. In particular, I applied this technique in [Chouaib *et al.*(2020)].

```
import bigfish.segmentation as segmentation

# first attempt of segmentation (with missing nuclei)
#nuc_label_1 = model(nuc_image)

# remove segmented nuclei
remaining_nuc_image = segmentation.remove_segmented_nuc(
    image=nuc_image,
    nuc_mask=nuc_label_1)

# second attempt of segmentation
#nuc_label_2 = model(remaining_nuc_image)

# merge nucleus labels
nuc_label = segmentation.merge_labels(nuc_label_1, nuc_label_2)
```

4.2.3 Cell segmentation

Cell segmentation is often more difficult. It can involve images with cluttered cells, fluorescent labels with different quality or labels not initially designed to visualize the cytoplasm (for example, the smFISH channel). In addition, cells can exhibit more diversity in their morphological shapes than nuclei, and a successful method with HeLa cells could fail in other cell lines or bacteria images.

A watershed algorithm is available in *bigfish.segmentation* to discriminate adjacent cells, where a previous nucleus segmentation mask is used as marker. The Watershed transform is applied to the input channel, potentially regularized with the distance map from nuclei (an image where each pixel takes the distance to the nucleus).

A revisited deep watershed model

I also implement a deep learning solution for cell segmentation, inspired by Deep Watershed [Bai et Urtasun(2017)] and the use of distance maps for nucleus segmentation [Naylor *et al.*(2019)]. Two models are tested, with the same backbone architecture previously presented for nucleus segmentation: an encoder-decoder convolutional neural network with residual blocks.

A first model uses only one input image with cell information (in my case, CellMask™, smFISH or GFP). This model predicts two outputs: a binary mask of cell surface (like in semantic segmentation tasks) and a distance map to cell edges. The idea is that forcing the network to predict a distance map will ultimately push it to learn the specific local patterns indicating the position of the plasma membrane, a pre-requirement for instance segmentation. These outcomes are then used in a watershed algorithm with the segmented nuclei as seeds in order to return a segmentation mask for every cell instance. The model is trained with a combined loss averaging a binary cross-entropy loss for the surface prediction and a mean absolute loss for the distance map.

A second model uses two channels as input images, one for the nuclei and one for the cells. In addition to the two previous output images, it predicts a third output: a distance map to nucleus edges. Cell instances are obtained with the same strategy as above, namely by application of a watershed algorithm applied on the predicted distance map with the segmented nuclei as seeds. The idea is to add input information about the nuclei and to force the model to take it into account by returning a nucleus related prediction. I hypothesized that this would make cell segmentation more accurate, as more local cell and nuclei features are predicted. In *bigfish.segmentation* this second model is implemented, with a double input strategy.

```
import bigfish.segmentation as segmentation

# load pretrained model
model_cell = segmentation.unet_distance_edge_double()

# instance segmentation
cell_label = segmentation.apply_unet_distance_double(
    model=model_cell,
    nuc=image_nuc,
    cell=image_cell,
    nuc_label=nuc_label,
    target_size=256,
    test_time_augmentation=True)
```

Segmentation evaluation

The main advantages of the available deep learning models in *bigfish.segmentation* is to offer an efficient in-house segmentation solution, without the need to use another API, package or framework. It is fast to apply and can be a relevant first solution to try. However, for more challenging segmentation problems, FISH-quant v2 still enables the use of external resources like Cellpose or StarDist.

Both nucleus and cell segmentation models are trained with Adam optimizer [Kingma et Ba(2015)] until validation loss does not improve anymore. Performance is assessed with the mean Average Precision. The Intersection over Union (IoU) score is computed for each pair of predicted and ground truth instances (whose value ranges between 0 and 1). Prediction matches the ground truth if the IoU is above a specific threshold. Therefore, for a given threshold, I can compute True Positives (instances matched correctly), False Positives (predicted instances matching nothing), False Negatives (ground truth instances missed) and the Average Precision (AP) score such that:

$$\mathrm{AP} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP} + \mathrm{FN}} \tag{4.1}$$

The mean Average Precision is the average of the AP score for different IoU thresholds between 0.5 and 0.95. A higher value indicates a better agreement between prediction and ground truth instances.

Results for my deep learning implementations are shown in Table 4.1. Evaluation is performed on the multichannel dataset extracted from [Safieddine *et al.*(2021)]. As expected, best results for cell segmentation are obtained with CellMask™ input, while smFISH and GFP channels yielded similar AP scores. Interestingly, the addition of

Model	DAPI	CellMask™	smFISH	GFP
3-class U-Net	**0.6**	-	-	-
Distance map U-Net	-	**0.66**	0.59	0.58
Distance map U-Net (double input)	-	0.65	**0.63**	**0.62**

Table 4.1: Segmentation results for different input channels. Computed score is the mean Average Precision, the higher, the better. Best models are bold. Evaluation is performed over 19 images

nucleus information in input slightly improves cell segmentation results for the smFISH and GFP channels.

Postprocessing and refinement

Regardless of the segmentation method applied, *bigfish.segmentation* includes different methods to clean and refine the segmentation masks. The most simple operations consist in smoothing the mask boundaries with a median filter, removing the small disjoint masks or filling holes within the segmented areas.

It can also be useful to separate the labeled instances in the labelled image y. For this, I can simply calculate the morphological gradient $\rho = \delta(y) - \varepsilon(y)$ (the difference between the dilation δ and the erosion ε of a given image) and then separate the instance by setting $y(x) = 0$, $\forall x : \rho(x) > 0$. An example of such refined results is illustrated in Figure 4.1.

Lastly, if nucleus and cell segmentation maps have been generated by independent algorithms, they need to be matched. For this, I also propose a small helper function in *bigfish.segmentation*.

```
import bigfish.segmentation as segmentation
import bigfish.multistack as multistack

nuc_label = segmentation.clean_segmentation(
    nuc_label=nuc_label,
    delimit_instance=True)
cell_label = segmentation.clean_segmentation(
    cell_label=cell_label,
    smoothness=7,
    delimit_instance=True)
nuc_label, cell_label = multistack.match_nuc_cell(
    nuc_label=nuc_label,
    cell_label=cell_label,
    single_nuc=False,
    cell_alone=True)
```

4.3 Improving cell segmentation

I have contributed to two projects with the aim to improve specific aspects of biomedical segmentation. The first project consisted in developing a deep learning version of the active contour algorithm. However, this work is not finished and it yielded incomplete results so far, so here I only present the main idea. The second project aimed at

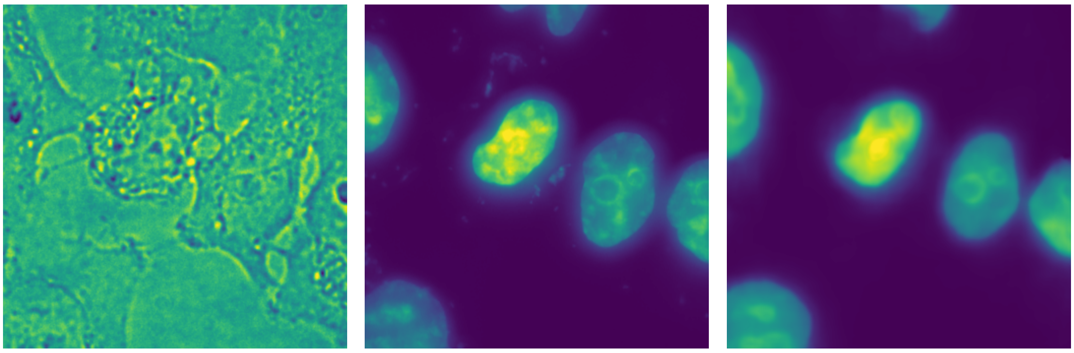


Figure 4.3: Example of in silico labeling, from [Bonte *et al.*(2022)]. (*Left*) Bright-field input image. (*Center*) Ground truth DAPI channel. (*Right*) Predicted DAPI

leveraging in-silico labeling in order to pre-train segmentation models without manual annotation and resulted in a publication [Bonte *et al.*(2022)].

4.3.1 Snake-like model

This project tries to address potential inconsistencies between nucleus and cell segmentation when these two tasks are performed independently. Sometimes the cell is correctly segmented, but not the nucleus, sometimes it is the opposite. If I cannot match a nucleus with a cell, the whole instance is discarded. In addition, nucleus is often far easier to segment than the cell.

The idea is to propagate or extend the cell outline from the nuclear region, which would perfectly solve this problem of inconsistency. The popular watershed approach does this, but it has a number of well-known limitations in the presence of strongly anisotropic cells and noisy boundaries (leakage).

Inspired by active contour models, the goal is to deform a spline curve around cell instance, but instead of solving an energy minimization problem a neural network would predict the deformation of the polygon. Such method has already been proposed for natural image segmentation like Deep Snake [Peng *et al.*(2020)] or DANCE [Liu *et al.*(2021)]. However, these models need to first detect the candidate object instance and then initialize the deformable polygon from the detected bounding box. Their entire pipeline relies on the performance of a detection model. In my case, the cell outline would be initialized from the nucleus outline (much easier to segment), before being iteratively distorted by a fully convolutional network. Even though the method seemed appealing at first sight, I did not obtain convincing results.

4.3.2 In silico pre-training

In a joint work with a master student and another PhD student, we aimed at leveraging In Silico labeling (ISL) task to pre-train segmentation model and mitigate the need for annotated training images. ISL consists in predicting fluorescent labels from bright-field or transmitted-light images [Christiansen *et al.*(2018),Ounkomol *et al.*(2018)]. The idea is to replace some fluorescence microscopy channels by their predictions (illustrated in Figure 4.3), in order to free channels for other analyses.

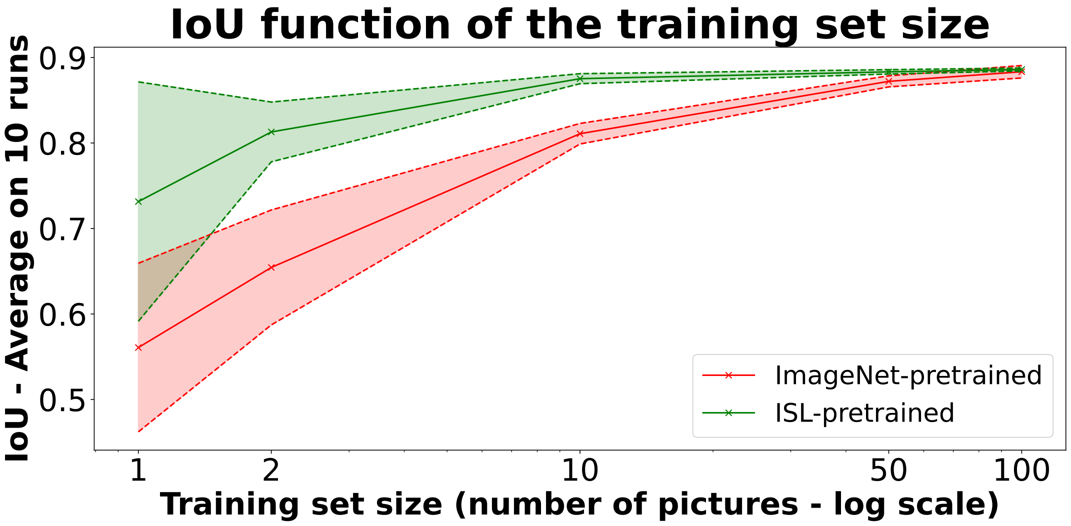


Figure 4.4: Results of in silico pre-training for a nucleus segmentation task, from [Bonte *et al.*(2022)]. IoU score for different training set size (the higher the better). Segmentation model with in silico pre-training (*green*) is more efficient in low training dataset regime than model with ImageNet pre-training (*red*)

In [Bonte *et al.*(2022)], we use the same model to perform ISL and segmentation. It is based on a U-Net architecture, with a DenseNet encoder branch [Huang *et al.*(2017)]. The first training on ISL makes the model learn intermediate visual representations of the cell. We speculate that these learned features are relevant for the segmentation task. In this case, the following segmentation training should be more efficient.

In Figure 4.4 we compare the IoU score of the trained models for different training set size. A second model pre-trained with a natural image classification task (from ImageNet [Deng *et al.*(2009)]) is also displayed for comparison purpose. The difference in terms of pre-training vanishes when the size of the training set increases. However, pre-training a segmentation model with ISL clearly compensates a low number of training images.

4.4 Conclusion

This chapter details the different algorithms available in *bigfish.segmentation* subpackage to segment nuclei and cells from fluorescent microscopy images. The main interest of these methods is to propose in-house segmentation solutions, easy to use in a few lines of code, and therefore complete the quantitative pipeline defined in FISH-quant. Yet, the research community is quite dynamic concerning biomedical segmentation. A lot of publications are released every year, with new datasets and improved techniques. Beyond developing a potential state-of-the-art model (which would be surpassed a few months later), the ability to easily integrate results from external resources is essential for FISH-quant. To this end, postprocessing methods are also implemented to format and refine segmentation masks.

Even if the segmentation has not been my main axis of research, I have explored some methods to alleviate limitations observed in cell segmentation. During my PhD I have mainly worked with multichannel images considering both nuclei and cells. For this reason, the consistency between nucleus and cell segmentation is critical. Such consistency can be directly enforced by the way an algorithm works, like in a watershed transform, or learned through a specific training strategy. Additionally, the need for a large amount of annotated images is still a hot topic in deep learning research. The use of in silico techniques to pre-train models seems to mitigate this limitation.

5

Spatial Feature Engineering

Abstract:

Each identified element in a cell is represented by its spatial coordinates, resulting in a coordinate representation of the cell. Two approaches are then presented to study RNA localization patterns. The first methods is to design hand-crafted features to characterize specific patterns. These features are implemented in FISH-quant. In the second approach, a vector representation of the RNA point cloud is learned and extracted from a neural network trained on a simulated task. Both techniques allow for downstream analyses such as quantification of the RNA distribution, supervised and unsupervised analysis.

Résumé:

Chaque élément identifié dans une cellule est représenté par ses coordonnées spatiales, ce qui permet de représenter la cellule comme un nuage de points. Deux approches sont ensuite présentées pour étudier les schémas de localisation de l'ARN. La première méthode consiste à développer manuellement des indicateur statistiques pour caractériser ces schémas. Ces indicateurs sont implémentés dans FISH-quant. Dans la seconde approche, une représentation vectorielle du nuage de points d'ARN est apprise et extraite à partir d'un réseau de neurones entraîné sur des données simulées. Les deux techniques permettent des analyses en aval telles que la quantification de la distribution de l'ARN, l'analyse supervisée et non supervisée.

Contents

5.1 From images to coordinates 69
5.1.1 Gathering all information at the single cell level 69
5.1.2 Statistical description 71
5.2 Hand-crafted localization features 72
5.2.1 Related work 72
5.2.2 Hand-crafted features to describe RNA localization patterns 73
5.3 Learned localization features 77
5.3.1 Related work 78
5.3.2 Problem statement 80
5.3.3 PointFISH 83
5.3.4 Experiment 85
5.3.5 Discussion 89
5.4 Conclusion 91

In this chapter, I present and discuss different approaches to derive quantitative profiles describing RNA localization patterns. By combining the outputs from detection and segmentation modules (Chapters 3 and 4 respectively), each cell is characterized by a point cloud (RNA), a set of landmarks (e.g. the nucleus or other cellular compartments) and potentially by some pre-digested subpatterns (such as RNA clusters). The challenge is then to derive a vector that is suitable for downstream analysis, namely quantification, supervised and unsupervised analysis.

In the first part, I describe the preparation of the input data, starting from segmentation and detection results obtained by the methods implemented in *bigfish.multitask*. In the second part, I manually select or design a set of localization features. These features are implemented in *bigfish.classification* and described in [Imbert *et al.*(2022b)]. In the third part, I present a different approach to compute spatial features. By training a point cloud model on a pretext task, I build a feature extractor that learns a relevant embedding [Imbert *et al.*(2022a)]. This approach is described in the paper:

A. Imbert, F. Mueller, et al. (2022), *PointFISH: learning point cloud representations for RNA localization patterns*, in 2022 European Conference on Computer Vision (ECCV 2022) Workshop on BioImage Computing.

5.1 From images to coordinates

The segmentation and detection methods presented in chapters 3 and 4 provide us with a list of detected objects (e.g. RNA and clusters) and segmentation masks (e.g. nucleus and cytoplasmic region), as illustrated in Figure 5.1. Here, the coordinates of the detected objects and segmentation masks are extracted and form the input data for which we will seek a quantitative description, amenable to machine learning.

We recall from the previous chapter that *bigfish* proposes 2D and 3D detection, but only 2D segmentation. This is not a conceptual limitation, but rather a pragmatic choice in our projects. Also, any external methods for detection and segmentation could be used, as long as output formats are compatible with *bigfish*.

5.1.1 Gathering all information at the single cell level

To extract and summarize FISH results at the single cell level, the only requirement is a segmentation mask of the cell, that allows us to associate each detected RNA and other objects to one cell. Segmentation of additional compartments is optional, but greatly improves the amount of relevant information on RNA localization assigned to each cell.

Labeling detected objects according to their localization in the cell

The nuclei segmentation masks allows us to identify the intranuclear RNAs. Technically, this amounts to assigning a region label to each of the detected RNAs or clusters. In general, the more subcellular compartments are available, the more detailed the description of RNA localization can become in the following.

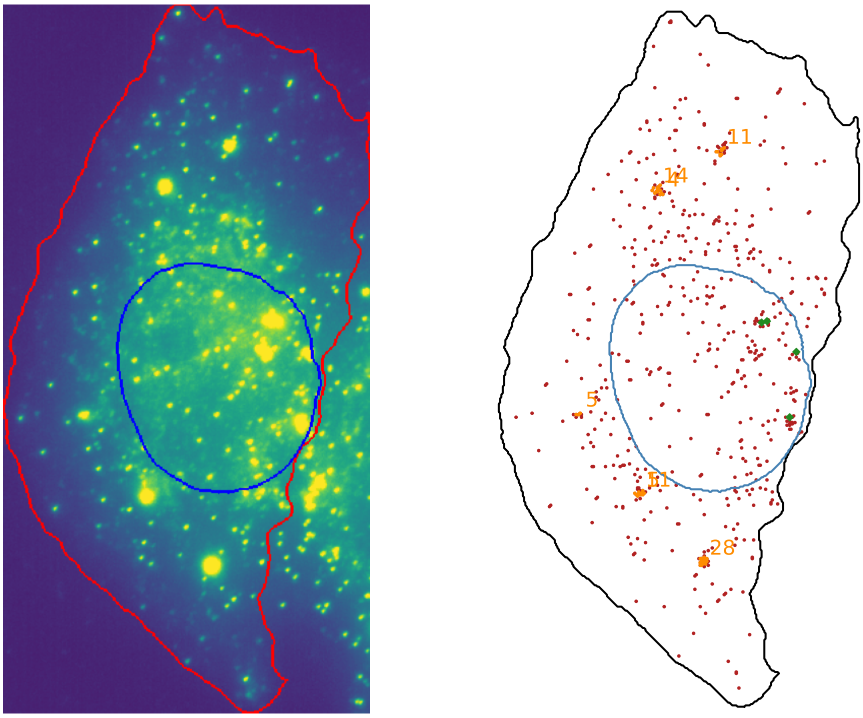


Figure 5.1: Contrasted original image with segmented boundaries (*left*) and coordinate representation (*right*). Plot build with *bigfish*

This compartmentalization might also change the interpretation we are giving to detected clusters. For instance, a cluster inside a nucleus might be interpreted as transcription site, while a cluster in the cytoplasm can have various biological interpretations (e.g. P-bodies, translation factories, stress granules).

In *bigfish* this labeling of detected RNAs and clusters can be achieved with a few lines:

```python
import bigfish.multistack as multistack

# discriminate foci and transcription sites
spots_no_ts, foci, ts = multistack.remove_transcription_site(
    rna=spots,
    clusters=clusters,
    nuc_mask=nuc_label,
    ndim=3)
```

The advantage of a coordinate representation

In Figure 5.1 we see a cropped smFISH image on the left, with cell and nuclear membranes in red and blue respectively. On the right, these membranes are illustrated in black and blue respectively. In addition, we see RNA spots (in red), RNA clusters (in orange, with the estimated number of RNA clustered) and transcription sites (in green).

With such *extraction* we lose pixel-wise information like intensity values or image texture. We also rely on detection and segmentation performances to return meaningful coordinates. Nonetheless, coordinate representation is a sparse and more natural representation for mRNA localization pattern classification: the coordinates - rather than the fluorescence images themselves - represent the actual measurement I want to make, when I perform smFISH experiments. Finally, in contrast to raw images, coordinates of detected objects allow a much more convincing data integration: the number of potential batch effects and biases is greatly reduced.

```python
import bigfish.multistack as multistack

# extract cell-level results
fov_results = multistack.extract_cell(
    cell_label=cell_label,
    ndim=3,
    nuc_label=nuc_label,
    rna_coord=spots_no_ts,
    others_coord={"foci": foci, "transcription_site": ts},
    image=image_contrasted,
    others_image={"dapi": nuc_mip, "smfish": smfish_mip})
```

Filtering

We can filter the data at the cellular and object level.

At the cellular level, we might want to ensure that only one nucleus is assigned to each cell, and we can remove cells at the border of the FoV. Their segmentation is incomplete and might bias final results. Third, extracted cells can be filtered out according to the number of detected objects (especially the number of RNAs). By censoring empty cells, we remove potential outliers, detection or segmentation failures and therefore help a subsequent statistical analysis.

We can also filter out individual objects according to their localization in the cell. A user might want to exclude a detected object if it locates within a segmented surface. For example, some studies require to remove transcription sites before further analysis [Chouaib *et al.*(2020)], or on the contrary focus on quantification of transcription sites.

5.1.2 Statistical description

At this stage, standard and useful statistics for every cell can already be computed. In particular, we can measure cell and nucleus areas, but also RNA distribution, inside and outside nucleus. From cluster coordinates, we can estimate cluster size, as well as proportion of clustered RNAs. RNA proportion in specific cellular compartments are also noteworthy. Such indicators are already relevant to quantify or validate meaningful biological properties. For example, a recent study [Cochard *et al.*(2022)] uses *bigfish* to estimate RNA recruitment in bioengineered condensates (segmented from a GFP channel).

```
import bigfish.multistack as multistack

# compute cell-level statistics
df = multistack.summarize_extraction_results(fov_results, ndim=3)
```

5.2 Hand-crafted localization features

Here I present hand-crafted features I have implemented for the analysis of RNA localization patterns. In section 5.2.1, I present related work. We will see that there is already a large number of features available in the literature. In section 5.2.2, I present the features I have chosen for our studies.

5.2.1 Related work

Feature engineering with bioimages

In bioimage analysis, in particular for High Content Screening, there is an abundant literature on feature families that have been used for cell and whole image classification. They can be roughly categorized into shape features and texture features. Examples include Haralick features [Haralick *et al.*(1973)], statistical geometric features [Walker et Jackway(1996)], local binary patterns [Ahonen *et al.*(2006)], moment based features [Reeve et Prokop(1992)] and morphological granulometries [Serra(1983)], to name a few.

These features have been used to classify cells according to their protein localization patterns [Boland *et al.*(1998),Glory et Murphy(2007)], or their phenotypes [Wang *et al.*(2008),Jones *et al.*(2009),Walter *et al.*(2010),Walter(2020)] or classify directly full images with entire cell populations [Uhlmann *et al.*(2016)].

Importantly, such feature families are readily available in standard open-source software for computational phenotyping, such as CellProfiler [Carpenter *et al.*(2006),Jones *et al.*(2008),McQuin *et al.*(2018)], CellCognition [Held *et al.*(2010)], Mahotas [mah(2013)] and ilastik [Berg *et al.*(2019)].

Most of these hand-crafted features come from the computer vision literature: they are pretty generic with limited biological interpretability. Furthermore, they are designed to quantitatively describe object shapes and textures, rather than directly representing spatial distributions of biomolecules. Of note, this also holds for protein localization screens [Glory et Murphy(2007), Ouyang *et al.*(2019b)]. This is due to the fact that with traditional microscopy, individual proteins cannot be resolved. In addition, proteins can be abundant and often concentrate on specific structures. As a consequence, the protein signal cannot be decomposed into single points, and it is therefore logical to describe their spatial distribution by texture features. A notable exception is provided by super-resolution microscopy, where protein distributions are indeed modeled as point clouds [Levet *et al.*(2019)].

In conclusion, there is a need for features describing spatial distributions of RNA molecules inside the cell, implemented in user-friendly open-source tools.

Feature design to describe RNA localization

Hand-crafted features to classify RNA localization patterns were already developed in previous studies [Battich *et al.*(2013), Samacoits *et al.*(2018)]. Generally such features are inspired by literature on spatial statistics [Ripley(2005)] and adapted for fluorescence microscopy images [Lagache *et al.*(2015), Stueland *et al.*(2019)]. Furthermore, several packages implement modules to perform smFISH analysis and compute these hand-crafted features [Mueller *et al.*(2013), Savulescu *et al.*(2019), Mah *et al.*(2022)].

In [Battich *et al.*(2013)], authors designed 18 features "that reflect the relative localization of each spot in a single cell, with respect to both the cell and other spots". The authors compute per-transcript features, then exploit their per-cell mean and standard deviation to identify subcellular localization patterns.

Two recent Python libraries propose algorithms for the quantification of FISH experiments. DypFISH [Savulescu *et al.*(2019)] was developed to analyze RNA and protein (co-)localization. It includes features to investigate clustering and Main Microtubule Organizing Center (MTOC)-related patterns. Across different imaging acquisitions of cells with a architecture constraints, they propose techniques to study mRNA-protein spatial distribution [Savulescu *et al.*(2021b)]. Bento [Mah *et al.*(2022)] proposes modules to analyze images generated with SeqFISH and MERFISH. They adapt feature extraction pipeline to highly multiplexed spatial transcriptomics data.

The features I present in this chapter extends previous work in our groups, implemented in MATLAB [Mueller *et al.*(2013), Samacoits *et al.*(2018)], where more than 20 features were proposed to analyze mRNA localization patterns. Compared to previous feature sets [Battich *et al.*(2013)], they led to better performance in RNA pattern classification, as shown by simulations. A first set of features includes distance features between RNA spots and cell or nucleus. A second set of features involved the Ripley K-function [Ripley(2005)]:

$$K(r) = \frac{1}{n} \sum_{i=1}^{n} \frac{N_i(r)}{\lambda} \tag{5.1}$$

With $N_i(r)$ the number of RNAs in a circle of radius r centered on the i^{th} RNA and λ the total density of RNAs in the cell. It quantifies the aggregation or dispersion of mRNAs. Several features are designed from these values, exploiting their maximum or their correlation with the radius r. However, Ripley features are sensitive to boundary effects (RNAs close to a membrane have a limited neighborhood) and require to be correctly normalized. A third set of features is based on assessing RNA density measured in subcellular regions, defined by morphological operators. Lastly, dispersion and polarization indices are implemented.

5.2.2 Hand-crafted features to describe RNA localization patterns

Here, I present the features that I have implemented in *bigfish.classification*. Many of these features were designed for our own studies [Chouaib *et al.*(2020), Safieddine *et al.*(2021), Pichon *et al.*(2021)], but are sufficiently generic to be of general utility. These features capture more specific information about RNA localization, beyond expression levels (*nb_rna*).

One of the important aspects is the normalization of these features. Indeed, many features are per definition highly dependent on the expression level or the cellular morphology. Here, the objective is to define features that characterize preferential localization of RNA, independently from expression levels.

Distance features

In order to capture preferential localization with respect to the cell membrane, we define *index_mean_distance_cell*:

$$\text{index\_mean\_distance\_cell} = \frac{\overline{d_{cell}(x_i, y_i)}}{\lambda_{cell}} \tag{5.2}$$

With $d_{cell}(x_i, y_i)$ the euclidean distance to the cell membrane for the RNA i and λ_{cell} the expected average distance under uniform RNA distribution. λ_{cell} is calculated efficiently as the average value of the 2D distance map from the cell membrane. The normalization with respect to λ_{cell} makes the feature robust with respect to different cell morphologies [Samacoits *et al.*(2018)]: we measure how much we deviate from a uniform distribution, given the cell morphology and the expression level.

Similarly, we compute the normalized average distance of RNAs to the nucleus: *index_mean_distance_nuc*. Alternative computation with the median function is available for these two features too.

In the current version, segmentations are provided in 2D (and consequently, the distances are also defined in 2D).

Compartment density features

The idea of this feature family is to partition the cellular region into several subregions and to evaluate the RNA density in these regions. This corresponds to spatial histograms of RNA distribution. These features come in two variants:

1. *propotion* is the proportion of RNAs inside the region with respect to the total number of RNAs in the cell.
2. *index* is the ratio between measured and expected RNAs inside the region.

The choice of the partition then conditions the kind of patterns that will can be differentiated with the feature family. First, as already indicated above, we defined as a coarse partition the nuclear and cytoplasmic region and calculate the proportion of RNAs within the nucleus (*proportion_rna_in_nuc*).

Second, we defined regions according to distance intervals to the nucleus and the cell membrane respectively. Region R_i is thus defined as:

$$R_i = \{x \mid d_{compartment}(x) \in [\tau_i, \tau_{i+1}]\} \tag{5.3}$$

where x is a point in the image plane, and *compartment* can mean the nucleus or cytoplasmic membrane. We count the number of RNA falling in R_i and calculate the proportion and the index.

In our studies, we defined regions with respect to their distance to the nucleus, where the interval boundaries were $\tau in \{-500, 500, 1000, 1500, 2000\}$ (in nm) and where

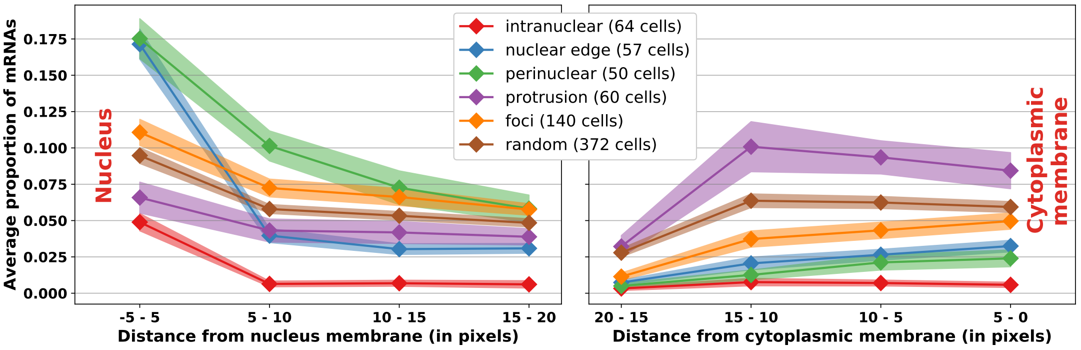


Figure 5.2: Proportion of mRNAs at various distances from cell and nuclear membranes. Dataset and annotated patterns from [Chouaib *et al.*(2020)]. *Colored areas* represent the 95% confidence interval

$d_{nucleus} < 0$ refers to points inside the nucleus. Furthermore, we defined regions with respect to their distance to the cytplasmic membranes, using the interval boundaries $\tau in \{0, 5000, 1000, 1500, 2000\}$ (in nm). In summary, we thus have:

- $proportion\_rna\_nuc\_radius\_\tau_i\_\tau_{i+1}$
- $index\_rna\_nuc\_radius\_\tau_i\_\tau_{i+1}$
- $proportion\_rna\_cell\_radius\_\tau_i\_\tau_{i+1}$
- $index\_rna\_cell\_radius\_\tau_i\_\tau_{i+1}$

In order to investigate the discriminative power of these features, we visualized the average RNA proportion for the different regions in Figure 5.2 and the corresponding 95% confidence interval for different patterns. These patterns will be introduced in detail in chapter 6. In brief, *intranuclear* refers to an accumulation of RNA inside the nucleus, *nuclear edge* to an accumulation in proximity of the nuclear envelope, *perinuclear* to a preferred localization close to the nucleus but less pronounced than *nuclear edge*, *protrusion* to an accumulation in cell extensions, *foci* to RNA clusters and *random* to the absence of all the other patterns. As expected, nuclear edge and perinuclear patterns present a higher proportion of RNAs along the nuclear membrane. On the opposite, cells with a protrusion pattern have a higher RNA density along the cell membrane.

Sometimes, it can also make sense to probe for the presence of RNAs in specific regions that are not necessarily present in all cells. In order to quantify the RNA presence in protrusions for instance, we build the partition by a morphological top-hat: $f - \gamma_B f$, where γ_B is the morphological opening with a structuring element B. This operation extracts the parts of the cytoplasm that cannot contain the structuring element. Here, we use a disk with radius $300nm$.

Dispersion features

I implemented three features described and tested in a recent paper [Stueland *et al.*(2019)] to quantify RNA polarization and dispersion within the cell.

Polarization index is computed by comparing RNA point cloud centroid and cell centroid:

$$\text{index\_polarization} = \frac{\sqrt{(x_{rna} - x_{cell})^2 + (y_{rna} - y_{cell})^2}}{Rg_{cell}} \tag{5.4}$$

With (x_{rna}, y_{rna}) the coordinates of the RNA centroid and (x_{cell}, y_{cell}) the coordinates of the cell centroid. The radius of gyration Rg_{cell} normalizes the index for different cell sizes. It is defined as the root-mean-squared distance between every cell pixel and the cell centroid. The higher, the more polarized RNAs are.

The dispersion index measures the dispersion of the RNA point cloud. In addition to the extracted coordinates, its computation also implies pixel intensities from the original smFISH image:

$$\text{index\_dispersion} = \frac{\frac{\sum_i d_i^2 I_i}{\sum_i I_i}}{\frac{\sum_j d_j^2 I_j}{\sum_j I_j}} \tag{5.5}$$

With d_i and d_j the euclidean distance of RNA i and cell pixel j to the RNA centroid respectively, I_i the pixel intensity of RNA i and I_j the pixel intensity of cell pixel j. Pixel intensity of transcripts distant from the RNA centroid are overweighted. As the index is normalized considering every pixel j from the cell mask, it tends to 1 when RNA point cloud is uniformly distributed. A diffuse point cloud has a value greater than 1. On the opposite, if RNAs are concentrated anywhere in the cell, index value is lower than 1.

The peripheral distribution index measures how close the RNAs localize to the cell periphery (*index_peripheral_distribution*). Its computation is similar to the dispersion index, but the RNA centroid is replaced by the nucleus centroid in the equation. A completely dispersed point cloud still has a value of 1, but it increases if RNAs move toward the cell periphery, with a concentrated or diffused pattern. Again, an aggregation of transcripts around the nucleus centroid (often close to the cell centroid too) decreases index value.

Centrosomal features

The nuclear and cytoplasmic membranes are major landmarks of the cell, but in more specific screens, it might be interesting to assess the distance distributions with respect to other compartments in the cell, if such compartments are labeled in the experiments. For instance, I have worked on a screen, where we wanted to study RNA localization related to the Microtubule Organizing Centers MTOC [Safieddine *et al.*(2021)], which I detail in chapter 6, where centrosomes were fluorescently labeled.

A first obvious feature is the average (or median) distance between RNAs and the closest detected centrosomes (up to two centrosomes can be detected in the cell): *index_mean_distance_centrosome*. Similarly to the distance features I compute with the cell or nuclear membranes, I compute the expected distance under uniform RNA distribution for normalization.

A second set of features consists in delimiting an area around each centrosome to be considered as centrosome's neighborhood. In my case I manually choose a radius of 2000nm around centrosomes to define such areas, as illustrated in Figure 5.3. I can then compute normalized RNA count (*index_rna_centrosome*) or RNA proportion (*proportion_rna_centrosome*) in these regions.

Lastly, I derived a feature from the dispersion index described above. I define a centrosomal dispersion index to quantify RNA dispersion around centrosomes: *index_centrosome_dispersion*. The feature is designed like the dispersion index, but instead of RNA centroid, I use the closest centrosome coordinates to compute the euclidean distance. The lower, the closer RNAs localized to the centrosomes.

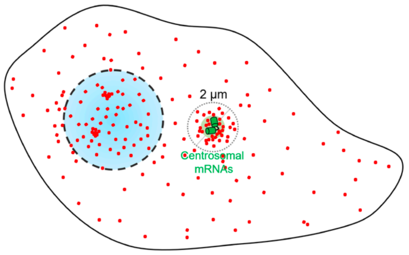


Figure 5.3: Centrosomal RNAs and its neighborhood from [Safieddine *et al.*(2021)]

Cluster features

In addition, to the distances to landmarks, it is often useful to also analyze inter-point distances. In particular, small clusters of RNAs are of biological interest. For this, we decided to rather represent these clusters by detection (see subsection 3.2.2) and not by generic inter-point features, such as those based on the Ripley's K-function. More specifically, number of detected clusters (*nb_foci*) or RNA proportion inside these clusters (*proportion_rna_in_foci*) are relevant features to identify transcripts with a tendency for clustering.

```python
import bigfish.classification as classification

# compute features
features, features_names = classification.compute_features(
    cell_mask=cell_mask,  # individual cell mask
    nuc_mask=nuc_mask,  # individual nucleus mask
    ndim=3,
    rna_coord=rna_coord,
    smfish=smfish,
    voxel_size_yx=103,  # in nanometer
    foci_coord=foci_coord,
    compute_distance=True,
    compute_intranuclear=True,
    compute_protrusion=True,
    compute_dispersion=True,
    compute_topography=True,
    compute_foci=True,
    compute_area=True,
    return_names=True)
```

5.3 Learned localization features

A second approach to analyze RNA localization patterns is to learn features. To this end I designed and trained a deep learning model, PointFISH, on a simulated pretext task [Imbert *et al.*(2022a)]. I then reused the internal representation learned by the model as a point cloud embedding to discriminate RNA localization patterns. The later was evaluated on an experimental dataset.

5.3.1 Related work

Learning features and embeddings

A neural network learns representations that can be used for transfer learning. One advantage is to pre-train relevant representations on a first task with a large and general annotated dataset, before addressing a more difficult or specific task with sometimes a limited dataset available. Such model can be used as a feature extractor by computing features from one of its intermediate layers. Computer vision community progressively replaces hand-crafted features [Lowe(1999), Bay *et al.*(2006)] by deep learning features to analyze images. The best convolutional neural networks pretrained on large and general classification challenges [He *et al.*(2016), Szegedy *et al.*(2016), Tan et Le(2019), Huang *et al.*(2017)] are used as backbone or feature extractor for more complex task like face recognition, detection or segmentation. NLP community follows this trend as well with a heavy use of word embeddings [Mikolov *et al.*(2013), Joulin *et al.*(2016)] or the more recent transformer models. As a last example, with graph computation, node2vec [Grover et Leskovec(2016)] learns ”task-independent representations” for nodes in networks.

Such embeddings have also the advantage to be a continuous and numerical representation. This is especially useful when dealing with a non-structured data like text or graph. With spage2vec [Partel et Wählby(2021)], authors learn a low dimensional embedding of local spatial gene expression (expressed as graphs). Eventually, they identify meaningful gene expression signatures by computing this embedding for tissue datasets.

Convolutional features

Because I analyze smFISH images, a first intuition would be to build a convolutional neural network to directly classify localization patterns from the fluorescent image. This approach is actually already in place for protein localization. Unlike RNA, proteins are usually studied with fluorescent protein, like GFP markers, and are in general impossible to resolve individually (except by using super-resolution microscopy, a technique that is not used in HCS today). In this case, they appear as a gradient of intensity in the fluorescent image and thus protein localization has often been approached like texture classification problem. First studies [Boland *et al.*(1998)] for example compute a set of feature from the microscopy image before training a classifier. With recent successes of deep learning, protein localization is now tackled with convolutional neural network, but still framed as a texture classification problem. After crowdsourcing annotations for the Human Protein Atlas dataset [Uhlén *et al.*(2015)] (through a video game), researchers trained a machine learning model (Loc-CAT) from hand-crafted features to predict subcellular localization patterns of proteins [Sullivan *et al.*(2018)]. In a second time, they organized an online challenge [Ouyang *et al.*(2019b)] where a majority of top-ranking solutions were based on convolutional neural networks. For protein localization the shift from hand-crafted features to convolutional features is significant and allows more accurate and robust pipelines.

A recent perspective paper [Savulescu *et al.*(2021a)] fosters the use of deep learning models for RNA localization analysis. Today, such analysis can be performed with fluorescent images or RNA sequencing. Authors emphasize the recent successes and flexibility of neural nets with both types of input, and therefore the possibility to design a multimodal pipeline. However, smFISH images have clear spots that can be individually resolved and easily detected. Therefore, a texture classification approach seems suboptimal to address RNA localization pattern recognition.

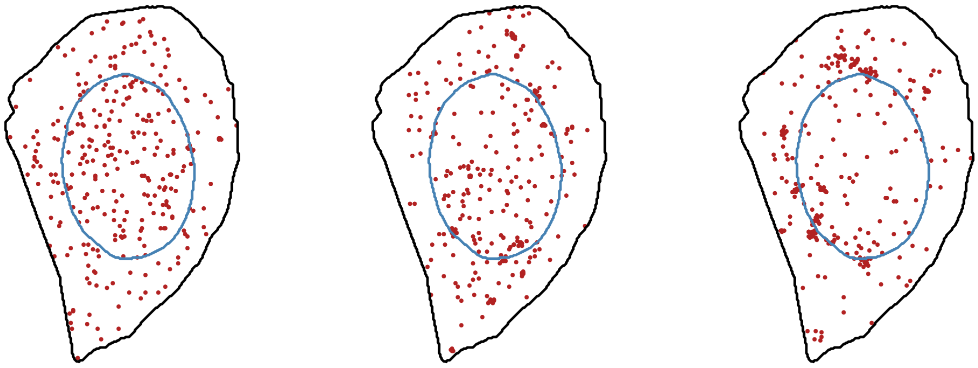
Point cloud models

I postulate that learning to classify RNA localization patterns directly from detected spots coordinates is the most efficient approach. A point cloud has an unordered and irregular structure. Projecting the coordinates into images or voxels [Maturana et Scherer(2015)] transforms the problem as an easier vision challenge, but it comes along with some input degradations. It dramatically increases the memory needed to process the sample and loses relevant spatial information. In case of RNA point cloud, I have contributed to a study, where we explored this approach by using convolutional neural network on a binary image of the point cloud [Dubois *et al.*(2019)]. Qualitative analysis of the clustering results indicated that the method does capture relevant signal, but it still cumbersome to first detect individual RNAs, to represent them in a binary image, which is then processed with a CNN [Dubois *et al.*(2019)] (a more detailed description of the method is available in appendix C).

A recent paper [Khater *et al.*(2019)] proposes to train a machine learning pipeline to discriminate caveolae clusters from scaffolds clusters. These cell structures can be recognized through the detection of caveolin-1 proteins and the shape of the resulting point clouds acquired from Single-molecule Localization Microscopy (SMLM) techniques. The authors compare three pipelines to address the problem: a random forest classifier trained on hand-crafted features, a convolutional neural network applied on the point cloud image and more importantly a PointNet fed directly with point cloud. Albeit legitimate, the PointNet method is less successful for this task than the two others pipelines.

PointNet [Qi *et al.*(2017a)] is a seminal work that leads the way for innovative models to address shape classification. It directly processes point clouds with shared MLPs and a max pooling layer, making the network invariant to input permutation. However, the pooling step is the only way for the model to share information between close points, which ultimately limits its performance. Yet, recent research dramatically improves point cloud modelling and especially the capture of local information.

PointNet++ [Qi *et al.*(2017b)] learns local geometric structures by recursively applying PointNet to different regions of the point cloud, in a hierarchical manner. This way, local information can be conveyed through the network more efficiently. DGCNN [Wang *et al.*(2019)] proposes a new EdgeConv layer where edge features are computed between a point and its neighbors. Some models propose to adapt convolutions to point cloud by designing new weighting functions or kernel operations like PointCNN [Li *et al.*(2018)], PointConv [Wu *et al.*(2019)] or KPConv [Thomas



(a) 10% clustered RNA (b) 50% clustered RNA (c) 90% clustered RNA

Figure 5.4: Foci pattern with increasing pattern strength simulated with *simfish*

et al.(2019)]. Another inspiration from the computer vision or NLP literature is the attention-based model. To this end, PointTransformer [Zhao *et al.*(2021)] proposes an attention layer to be applied to local regions within the point cloud. Last but not least, PointMLP [Ma *et al.*(2022)] proposes a simple but still efficient network with a pure deep hierarchical MLP architecture.

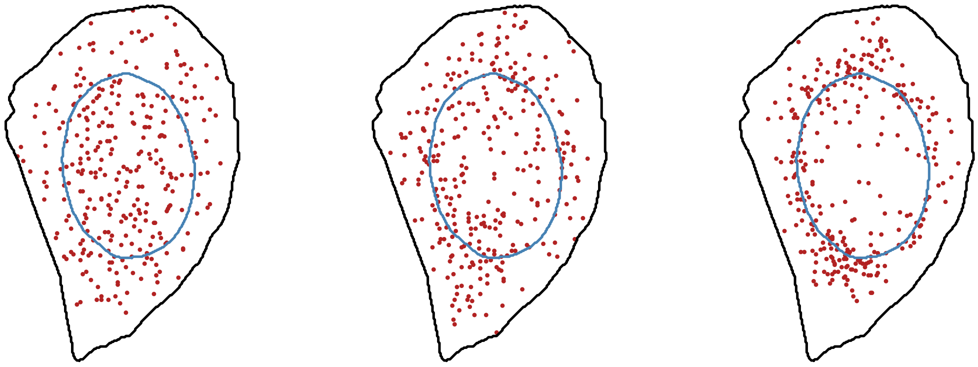
5.3.2 Problem statement

The objective is to train a model, that takes as input directly the point cloud coordinates as an input and compute a continuous vector representation.This representation can then be used for classification of different RNA localization patterns. Such a deep learning model might require a large volume of annotated data to reach satisfying performance. Manual annotation of RNA localization patterns is complex and time consuming. For rare patterns, it might even be impossible to find cells in sufficient numbers. For this reason, I turned to simulation to build a large training set, from which I can train a point cloud neural network, providing a feature vector that describes RNA spatial distributions. Finally, I evaluate these learned features on a experimental dataset.

Localization pattern simulations

To build my simulated dataset, I use methods implemented in *simfish*. My package exploits a template of 318 real cells to simulate realistic point clouds. They were originally extracted for FISH-quant v1 [Samacoits *et al.*(2018)] to simulate realistic cell and nucleus shapes. The 3D segmentation masks for the cell and the nucleus were extracted from smFISH experiments against GAPDH, an abundant house-keeping gene localizing in the cytoplasm. With respect to the previous version [Samacoits *et al.*(2018)], I made the following improvements:

- Migration to Python and removal of any proprietary dependencies.
- Significant acceleration of the simulation process.
- Extension of available localization patterns.



(a) 10% perinuclear RNA (b) 50% perinuclear RNA (c) 90% perinuclear RNA

Figure 5.5: Perinuclear pattern with increasing pattern strength simulated with *simfish*, from [Imbert *et al.*(2022a)]

- Redefinition of pattern strength to make simulations more consistent.

Simulation's outcome includes the cell mask and its membrane coordinates (in 2D), the nucleus mask and its membrane coordinates (in 2D) and the RNA coordinates (in 3D). To match with the rest of the *bigfish* pipeline, I voluntarily return 2D membrane coordinates.

For the simulation, we have to set 3 parameters: (1) the number n of RNAs, (2) the pattern strength p referring to the proportion of RNAs that localize according to the simulated pattern, where $(1 - p) \times n$ RNAs are placed uniformly across the cell or cytoplasm and (3) the localization pattern for the $n_{pattern} = p \times n$ RNAs.

The 9 patterns currently available are: random, foci, intranuclear, extranuclear, nuclear edge, perinuclear, cell edge, pericellular and protrusion. Random pattern is the default pattern where RNAs are simulated uniformly within the cell. The foci pattern consists in a random number of RNA clusters localizing outside the nucleus. I draw the expected number of RNAs λ per cluster from a uniform distribution between 5 and 21 RNAs. The number of clusters itself results from $n_{cluster} = \frac{n_{pattern}}{\lambda}$. For each cluster, a distinct number of RNAs is drawn again with a Poisson distribution of mean λ. Clusters are then localized outside the nucleus and remaining RNAs uniformly in the cell. In Figure 5.4 we can observe foci simulations with an increasing pattern strength (from 10% to 90% of clustered RNAs).

Other patterns are simulated with a similar scheme. In a first step I generate a probability map to bias the localization of $n_{pattern}$ RNAs. In a second step I complete the point cloud with random RNAs until I reach the expected number of transcripts. Intranuclear pattern has a uniform probability map inside the nucleus and zeros outside. Extranuclear pattern is the exact opposite, with nonzero probabilities outside the nucleus and zero inside. Nuclear edge and cell edge have nonzero probabilities along the nuclear and cell membranes. Similarly, perinuclear and pericellular are patterns where RNAs are polarized towards nuclear and cell membranes. Perinuclear probability map is build from the cell distance map. The euclidean distance is contrasted by computing its quadratic values. The same operation is performed to build the pericellular probability map, using the nucleus distance map. As a result, for pericellular pattern, RNAs have a higher probability to localize in regions distant from nucleus. The protrusion pattern has a uniform probability map within the annotated protrusion regions and zeros everywhere else. In Figure 5.5 different perinuclear simulations can be observed as an example. In addition, an overview of every simulated pattern is available in appendix A.2.

```
import simfish as sim

# load template dataset
path_template_directory = load_extract_template(path_output)

# localization pattern simulation
instance_coord = sim.simulate_localization_pattern(
    path_template_directory,
    n_spots=150,
    pattern="intranuclear",
    proportion_pattern=0.6)
```

Simulated dataset

With *simfish* I simulate a dataset with 8 different localization patterns: random, foci, intranuclear, extranuclear, nuclear edge, perinuclear, cell edge and pericellular. I choose these patterns since they represent a diverse panel of localization patterns in different subcellular regions. I simulate for each pattern 20,000 cells with 50 to 900 RNAs per cell, resulting in a full dataset of 160,000 simulated cells. Random pattern excepted, every simulated pattern has a proportion of RNAs with preferential localization ranging from 60% to 100%. With random simulations the pattern strength has no effect. I split my dataset between train, validation and test, with 60%, 20% and 20% respectively. In order to avoid data leakage, I make sure that simulations from the same cell template cannot be assigned to different splits. Finally, point clouds are augmented with random rotation along the up-axis, centered and normalized into the unit sphere. In order to test how the trained features generalize to unknown localization patterns, I did not simulate RNA localization in protrusions, while these localization class is present in the experimental dataset.

Pattern	# of cells
Random	372
Foci	198
Intranuclear	73
Nuclear edge	87
Perinuclear	64
Protrusion	83

Table 5.1: Annotated experimental dataset

Experimental dataset

I use the experimental dataset extracted from our study [Chouaib *et al.*(2020)] to validate the feature representation learned on simulated images. Images are obtained from a smFISH study in HeLa cells targeting 27 different genes, then processed with *bigfish*. After cleaning, it consists of 9710 individual cells, with cropped images and coordinates extracted. Cells have on average 346 RNAs in average and 90% of them have between 39 and 1307 transcripts. Furthermore, 810 cells have manually annotated

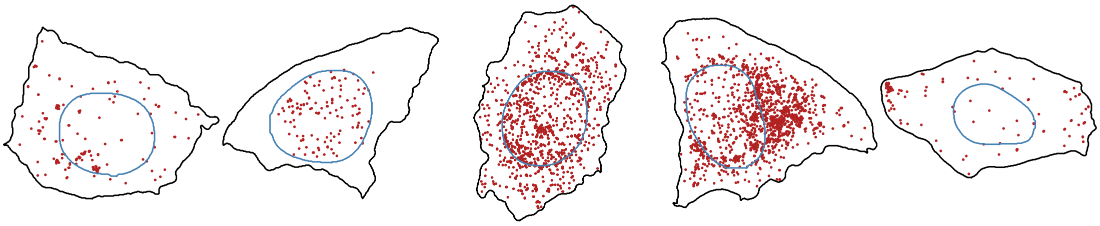

(a) Foci (b) Intranuclear (c) Nuclear edge (d) Perinuclear (e) Protrusion

Figure 5.6: RNA localization patterns from [Chouaib *et al.*(2020)]. Coordinate representations with RNA spots (*red*), cell membrane (*black*) and nuclear membrane (*blue*). Detection and segmentation results are extracted and visualized with *bigfish*

localization patterns, as detailed in Table 5.1 and illustrated in Figure 6.2. Importantly, these patterns are not mutually exclusive since cells can display several patterns at the same time (for example foci with a perinuclear distribution). I use these annotations as a ground truth for validation.

5.3.3 PointFISH

Input preparation

The original RNA point cloud $X \in \mathbb{R}^{N \times 3}$ can be extended in two ways. First, we can add 2D coordinates from the cell and the nucleus membranes. As they only exist in 2D, the z-coordinate is chosen arbitrarily to the average height of the RNA point cloud (0 if it is centered). Here, we sampled 300 and 100 points from the cytoplasmic and nuclear membrane, respectively. Also, we have to make sure that the network can distinguish between the different roles of spatial coordinates of the RNAs and the spatial coordinates of nuclear and cytoplasmic membrane. For this, we add two extra boolean variables for each position (RNA and membrane points), indicating their origin: RNA points are characterized by $(0, 0)$, nuclear membrane points by $(1, 0)$ and cytoplasmic membrane by $(0, 1)$. Second, I can compute the distance from cell and nucleus for every RNA node. Third, I can leverage the cluster detection algorithm from *bigfish* in order to label each RNA node as clustered or not, represented by an additional boolean variable. If morphological, distance and clustering information are used, our input matrix is thus $\tilde{X} \in \mathbb{R}^{N \times d}$, where $\tilde{N} = N + 300 + 100$ and $d = 8$ (3D spatial coordinates, 2 dimensions to indicate the spot type, 2 dimensions to indicate distance to nuclear and cytoplasmic membrane and 1 dimension to indicate whether the spot belongs to an RNA cluster).

Model architecture

I adopt the generic architecture introduced by PointNet [Qi *et al.*(2017a)]: successive point-wise representations with increasing depth followed by a max pooling operation to keep the network invariant by input permutation. I incorporate state-of-the-art modules to learn efficient local structures within the point cloud. As illustrated in Figure 5.7, I also adapt the network to the specificity of RNA point clouds.

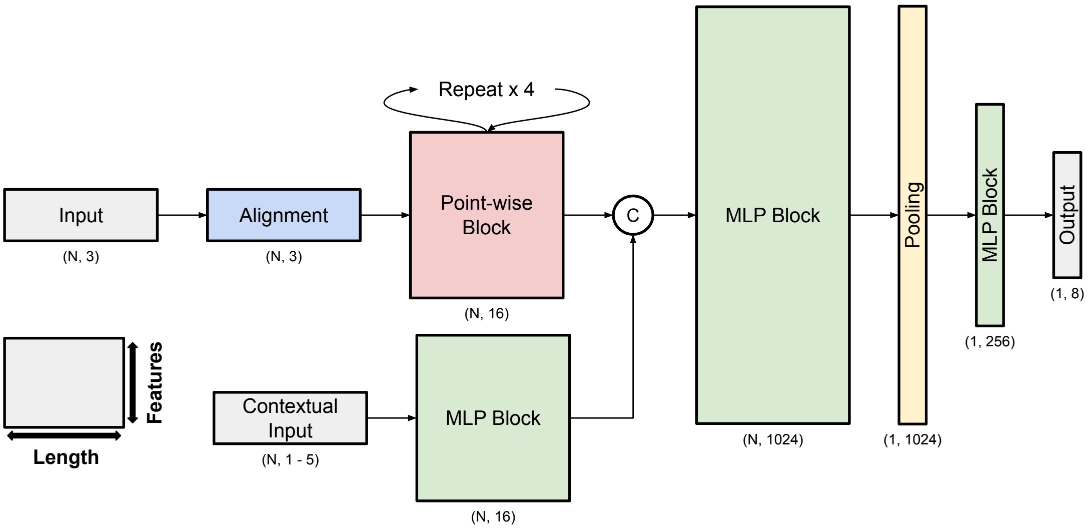


Figure 5.7: PointFISH architecture, from [Imbert *et al.*(2022a)]. Width and height of *boxes* represent output length and dimension, respectively. *Tuples* represent output shapes

Point-wise Block Instead of shared MLPs like PointNet, I implement a multi-head attention layer based on point transformer layer [Zhao *et al.*(2021)]. First, I assign to each datapoint x_i its 20 nearest neighbors $X(i) \subset X$, based on the euclidean distance in the features space. I also compute a position encoding $\delta_{ij} = \theta(x_i - x_j)$ for every pair within these neighborhoods, with θ a MLP. Three sets of point-wise features are computed for each datapoint, with shared linear projections ϕ, ψ and α. Relative weights between datapoints $\gamma(\phi(x_i) - \psi(x_j))$ are computed with the subtraction relation (instead of dot product as in the seminal attention paper [Vaswani *et al.*(2017)]) and a MLP γ. These attention weights are then normalized by softmax operation ρ. Eventually, datapoint's feature y_i is computed as weighted sum of neighbors value $\alpha(x_j)$, weighted by attention. With the position encoding added to both the attention weights and the feature value, the entire layer can be summarized as:

$$y_i = \sum_{x_j \in X(i)} \rho(\gamma(\phi(x_i) - \psi(x_j) + \delta_{ij})) \odot (\alpha(x_j) + \delta_{ij}) \tag{5.6}$$

For a multi-head attention layer, process is repeated in parallel with independent layers, before a last linear projection merges multi-head outputs. A shortcut connection and a layer normalization [Ba *et al.*(2016)] define the final output of my multi-head attention layer.

Alignment Module Albeit optional, this module is critical. Some papers stress the necessity to preprocess the input point cloud by learning a projection to *align* the input coordinates in the right space [Qi *et al.*(2017a), Wang *et al.*(2019)]. In addition, density heterogeneity across the point cloud and irregular local geometric structures might require local normalization. To this end, I reuse the geometric affine module described in PointMLP [Ma *et al.*(2022)] which transforms local datapoints to a normal distribution. With $\{x_{i,j}\}_{j=1,...,20} \in \mathbb{R}^{20\times 3}$, the neighborhood's features of x_i, I compute:

$$\{x_{i,j}\} = \alpha \odot \frac{\{x_{i,j}\} - x_i}{\sigma + \varepsilon} + \beta \tag{5.7}$$

Where $\alpha \in \mathbb{R}^3$ and $\beta \in \mathbb{R}^3$ are learnable parameters, σ is the feature deviation across all local neighborhoods and ε is a small number for numerical stability.

Contextual Inputs My RNA point cloud does not include all the necessary information for a localization pattern classification. Especially, I missed morphological information. To this end, deep learning architectures allow flexible insertions. Several contextual inputs $\tilde{X}$ can feed the network through a parallel branch, before concatenating RNA and contextual point-wise features. My best model exploits cluster and distance information in addition to RNA coordinates.

5.3.4 Experiment

Training and evaluation on simulated patterns

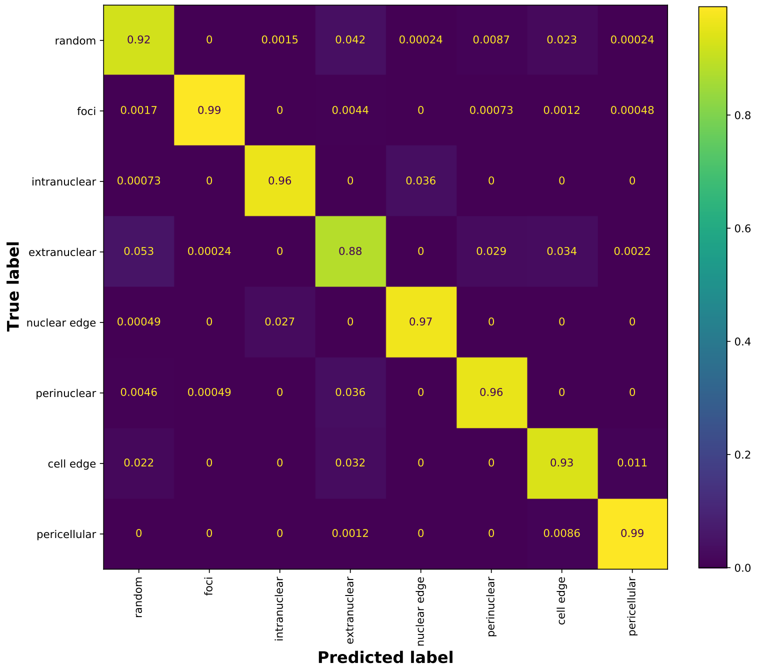
PointFISH is trained on the simulated dataset. The implementation is based on TensorFlow [Abadi *et al.*(2015)]. I use Adam optimizer [Kingma et Ba(2015)] with a learning rate from 0.001 to 0.00001 and an exponential decay (decay rate of 0.5 every 20,000 steps). Model is trained for a maximum of 150 epochs, with a batch size of 32, but early stopping criterion is implemented if validation loss does not decrease after 10 consecutive epochs. Usually model converges after 50 epochs. I apply a 10% dropout for the last layer and classifications are evaluated with a categorical cross entropy loss. Even if localization patterns are not necessarily mutually exclusive, for the simulations I trained the model to predict only one pattern per cell. For this reason, I did not simulate mixed patterns and assume it could help the model to learn disentangled representations. Training takes 6 to 8 hours to converge with a Tesla P100 GPU.

A first evaluation can be performed on the simulated test dataset. Because each pattern is equally generated, a simple accuracy metric is enough. For the experimental dataset, imbalanced between localization patterns implies a more robust metric like F1-score. With my best PointFISH models, I obtain a general F1-score of 95% over the different patterns (see confusion matrix 5.8).

Embedding extraction

From a trained PointFISH model I can remove the output layer to get a feature extractor that computes a 256-long embedding from a RNA point cloud.

Learned embedding I compute the embedding for the entire cell population studied in [Chouaib *et al.*(2020)]. All the 9170 cells can be visualized in 2D using a UMAP projection [McInnes *et al.*(2018)]. In Figure 5.9 every point represents a cell. Among the 810 annotated cells, those with a unique pattern are colored according to the localization pattern observed in their RNA point cloud. The rest of the dataset remains gray. Overall, PointFISH embedding discriminates well the different localization patterns. Intranuclear, nuclear edge and perinuclear cells form distinct clusters, despite their spatial overlap, as well as protrusions. Cells with foci can be found in a separated



True label \ Predicted label	random	foci	intranuclear	extranuclear	nuclear edge	perinuclear	cell edge	pericellular
random	0.92	0	0.0015	0.042	0.00024	0.0087	0.023	0.00024
foci	0.0017	0.99	0	0.0044	0	0.00073	0.0012	0.00048
intranuclear	0.00073	0	0.96	0	0.036	0	0	0
extranuclear	0.053	0.00024	0	0.88	0	0.029	0.034	0.0022
nuclear edge	0.00049	0	0.027	0	0.97	0	0	0
perinuclear	0.0046	0.00049	0	0.036	0	0.96	0	0
cell edge	0.022	0	0	0.032	0	0	0.93	0.011
pericellular	0	0	0	0.0012	0	0	0.0086	0.99

Figure 5.8: Confusion matrix with simulated test patterns (normalized over the *rows*)

cluster, but also mix with nuclear and perinuclear patterns. This confusion is not surprising as a large number of cells in the dataset present a nuclear-related foci pattern: cells have RNAs clustered in foci, which in turn are close to the nuclear envelope, in which case the cell would be labeled with both patterns.

Supervised classification Because PointFISH already return meaningful embeddings, I can apply a simple classifier on top of these features to learn localization patterns. I use the 810 manually annotated cells from the experimental dataset. I compare the 15 hand-crafted features selected in [Chouaib *et al.*(2020)] with my learned embedding. Every set of features is rescaled before feeding a classifier. Expert features include:

- The number of foci and the proportion of clustered RNAs.
- The average foci distance from nucleus and cell (normalized by the expected distance with a random foci distribution).
- The proportion or RNAs inside nucleus.
- The average RNA distance from nucleus and cell (normalized by the expected distance with a random RNA distribution).
- The number of RNAs detected in cell extensions (normalized by the expected number with a random RNA distribution) and the peripheral dispersion index [Stueland *et al.*(2019)].

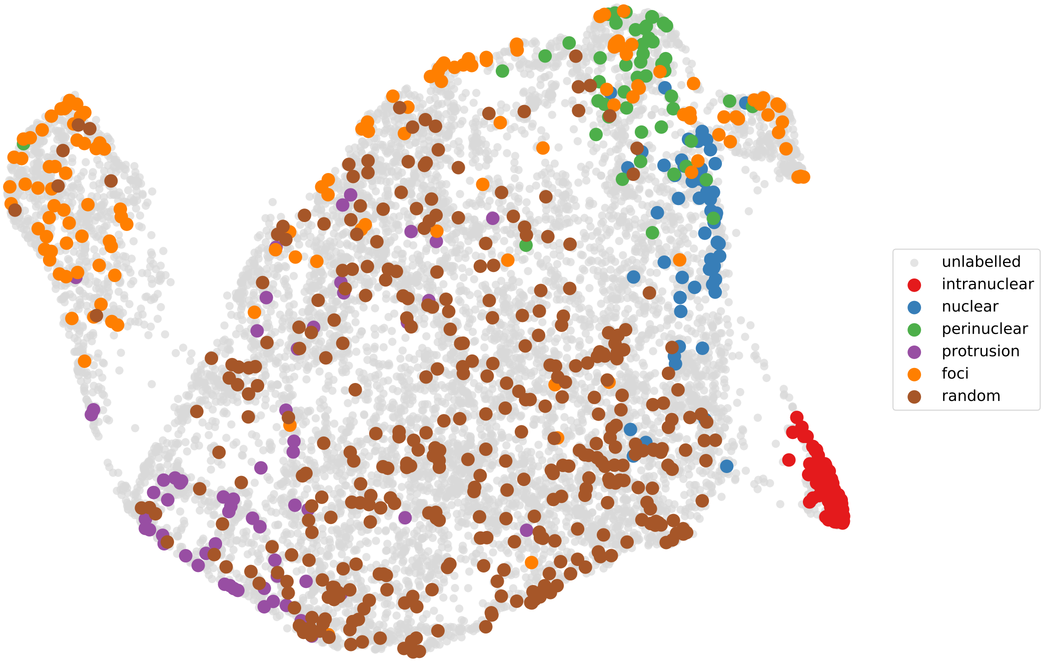


Figure 5.9: UMAP embedding with learned features, from [Imbert *et al.*(2022a)]. Each point is a cell from experimental dataset. Manually annotated cells are colored according to their localization pattern

- The number of RNAs within different relevant subcellular regions (normalized by the expected number with a random RNA distribution).

I design 5 binary classification tasks, one per localized pattern (random pattern is omitted). The classifier is a SVC model [Chang et Lin(2011)]. For evaluation purpose, I apply a nested cross-validation scheme. First a test set is sampled from the dataset (20%), then the remaining cells are used with a gridsearch to fit an optimal SVC model (with a 20% validation split). Parameters grid includes the choice between a linear or a RBF kernel and the strength of the regularization. The entire process is repeated 50 times, with different test split, and F1-score for each classification task is returned. This full evaluation pipeline is implemented with *scikit-learn* [Pedregosa *et al.*(2011)]. F1-score's distribution over 50 splits are summarized in Figure 5.10. Learned features match performances of hand-crafted features selected for the tasks. While the recognition of localization in protrusions is slightly worse, it is important to point out that I did not include simulations of this patterns in the training dataset.

Ablation study

I perform several ablation studies to evaluate the impact of different components in PointFISH model. In the point cloud literature, papers often propose new modules to improve network's performances, but they implement them in slightly different architectures. Heterogeneity in terms of normalization, latent dimensions or layer size complicate comparisons between techniques. In order to isolate the importance of each

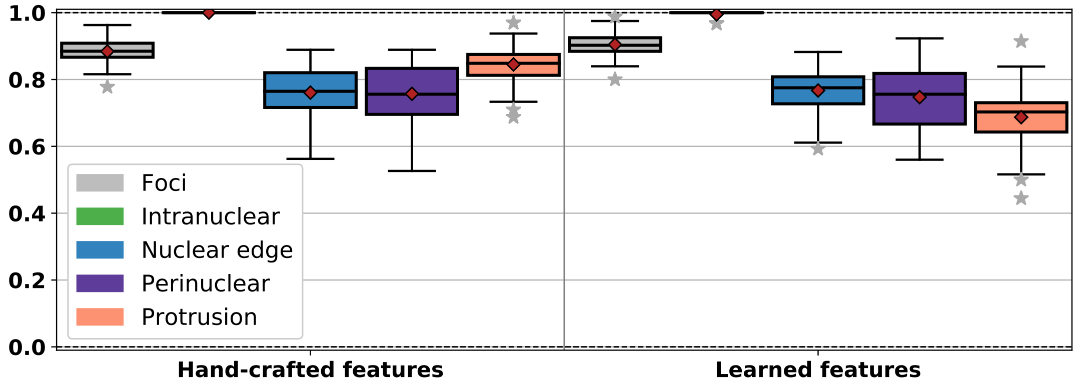


Figure 5.10: F1-score distribution of localization pattern classification (SVC model), from [Imbert *et al.*(2022a)]

element, I use a template architecture as illustrated in Figure 5.7. Instead of comparing PointFISH with DGCNN, I compare PointFISH with an equivalent network where attention layers are replaced by EdgeConv [Wang *et al.*(2019)]. The rest of the network remains strictly identical.

Distance	Cluster	Morphology	F1-score
✗	✗	✗	0.42
✓	✗	✗	0.74
✗	✓	✗	0.45
✓	✓	✗	0.81*
✓	✓	✓	**0.82**

Table 5.2: Impact of contextual inputs, from [Imbert *et al.*(2022a)]. F1-score is averaged over 4 trainings with different random seeds. Best model is in bold. Reference model is labelled with $*$

Additional input I compare the use of RNA point cloud only as input or the inclusion of contextual inputs through a parallel branch. RNA coordinates do not have any morphological information about the cell. In Table 5.2, this design logically returns the lowest F1-score. Three additional inputs are available: RNA distance from cell and nucleus (*distance*), RNA clustering flag (*cluster*) and the integration of cell and nucleus membrane coordinates (*morphology*). Best performances are obtained with, at least, distance and cluster information. Cell and nucleus coordinates do not increase significantly the classification performance. The reason is probably that the distance feature already represents the relevant information and adding membrane coordinates does thus not improve performance anymore. However, inclusion of membrane points dramatically increases the computation time of the model (as a larger point cloud has to be processed). Cluster information greatly improves foci pattern recognition while distances boost others localization patterns; both observations are expected.

Alignment module and Point-wise block To measure the impact of the geometric affine module [Ma *et al.*(2022)] I compare it with the TNet module implemented in PointNet [Qi *et al.*(2017a)]. I also design a variant TNetEdge where MLP layers extracting point-wise independent features are replaced with EdgeConv layers. Results are reported in Table 5.3. An alignment block seems critical at the beginning of the network. However, the geometric affine module is both more efficient (F1-score of 0.81) and much lighter than TNet and TNetEdge.

From the PointNet and DGCNN seminal articles, I also compare the use of their respective point-wise blocks against my multi-head attention layer. As expected, EdgeConv blocks convey a better information than PointNet by exploiting local neighborhood within point cloud (F1-score of 0.78 and 0.75 respectively). Yet, they do not match the performance of multi-head attention layer.

Concerning these layers, I evaluate how the number of parallel heads can influence the performance of PointFISH. By default, I use 3 parallel attention layers to let the model specialize its attentions vectors, but I also test 1, 6 and 9 parallel heads. In Table 5.3 we only observe a slight benefit between the original point transformer layer [Zhao *et al.*(2021)] (without one attention layer) and its augmented implementation.

Latent dimensions The second part of PointFISH architecture is standardized: a first MLP block, a max pooling operation, a second MLP block and the output layer. I quantify the impact of additional MLP layers within these blocks. My reference model returns an embedding with 256 dimensions (before the output layer). In a MLP block, I use ReLU activation and layer normalization, but also increase or decrease the depth by a factor 2 between layers. Before the pooling layer, the first MLP block includes 4 layers with an increasing depth (128, 256, 512 and 1024). After the pooling layer, the second MLP block includes 2 layers with a decreasing depth (512 and 256). Similarly, to return 128, 64 or 32 long embeddings, I implement 6 (128, 256, 512, pooling, 256 and 128), 5 (128, 256, pooling, 128 and 64) or 4 final layers (128, pooling, 64 and 32). We observe in Table 5.3 a drop in performance for the lowest dimensional embedding (64 and 32). This hyperparameter is also critical to design lighter models, with a division by 4 in terms of trainable parameters between a 256 and a 128 long embedding.

5.3.5 Discussion

Being able to directly process list of points provides the community with a tool to integrate large datasets obtained with different techniques on different model systems. While the actual image data might look strikingly different between such projects, they can all be summarized by segmentation masks of nuclei and cytoplasm, and a list of coordinates of RNA locations. Having methods that act directly on point clouds is therefore a strategic advantage for data integration.

The idea of training on simulated data provides us the opportunity to query datasets with respect to new localization patterns that have not yet been observed, and for which we do not have real examples so far. In addition, this strategy allows us to control for potential confounders, such as cell morphology, or number of RNAs, because we control the composition of the training data and can therefore remove statistical biases a priori.

Finally, I provide a generic method that can leverage these simulations to find a suitable representation, without the tedious process of handcrafting new features. It is not necessary that the simulated patterns are optimized as to resemble real data: they

Alignment	Point-wise block	# heads	# dimensions	# parameters	F1-score
-	Attention layer	3	256	1,372,608	0.73
TNet	Attention layer	3	256	1,712,521	0.74
TNetEdge	Attention layer	3	256	1,589,321	0.74
Affine	MLP	-	256	1,374,526	0.75
Affine	EdgeConv	-	256	1,387,006	0.78
Affine	Attention layer	9	256	1,403,334	**0.82**
Affine	Attention layer	6	256	1,387,974	**0.82**
Affine	Attention layer	3	256	1,372,614	0.81*
Affine	Attention layer	1	256	1,362,374	0.81
Affine	Attention layer	3	128	352,966	0.81
Affine	Attention layer	3	64	97,094	0.77
Affine	Attention layer	3	32	32,646	0.75

Table 5.3: Ablation studies on experimental dataset, from [Imbert *et al.*(2022a)]. F1-score is averaged over 4 trainings with different random seeds. Best models are bold. Reference model is labelled with $*$

rather serve as a pretext task. If a network is capable of distinguishing the simulated patterns, chances are high that the corresponding representation is also informative for slightly or entirely different patterns. Likewise, representations trained on ImageNet can be used for tumor detection in pathology images. I show this by omitting the protrusion pattern from the simulation. Indeed, in Figure 5.9, the protrusion patterns live in a particular region of the feature space, without specific training. There is however a drop in classification accuracy for the omitted class. We therefore see that if a class was not specifically simulated, the representation is less optimized with respect to the recognition of that class. On the other hand, the classification accuracy remains high even for the omitted class.

As a future work, a first valuable task would be the generation of datasets for validation across cell lines and imaging conditions in order to validate the robustness of a point cloud model. An interesting starting point might be the dataset released in [Savulescu *et al.*(2021b), Mah *et al.*(2022)].

A second interesting perspective would be to explicitly address the domain shift between simulated and real data. Indeed, training a model on simulated data might seem counter-intuitive, as we know that the simulated data follows a different distribution as compared to real data. Here we argue that the simulations do not necessarily have to reflect the reality in order to lead to powerful simulations; similar to lessons learned from self-supervised learning, where perturbations also do not need to represent real-world alterations. Nevertheless it would be interesting to see whether explicit domain adaptation is capable of improving the downstream classification performance.

Another potential improvement concerns the segmentation. I exploit so far a 2D segmentation because it is the more frequent use-case and the task for which I have a large number of trained models available. Yet, a 3D segmentation mask would allow a more comprehensive 3D point cloud mixing of RNA and cell datapoints. This could also dramatically improve pattern representation. Last but not least, training a point cloud model with a self-supervised training procedure would result in task-independent representations and could therefore be an interesting alternative to our simulations.

5.4 Conclusion

I first present the coordinate representation of the cell, from which I base my feature engineering and my statistical analysis. On the top of existing solutions to extract RNA spots and cell morphology coordinates, I then list the different hand-crafted features I designed to quantify and classify RNA localization patterns. These features are implemented in *bigfish.classification*. Lastly, in the third section, I propose to directly process the extracted point clouds, without the need to design hand-crafted features. For this, I leverage coordinates of simulated localization patterns to train a specifically designed neural network taking as input a list of points and associated features that greatly enhance generalization capabilities. Recent advances in point cloud analysis through deep learning models allows us to build a flexible and scalable pipeline that fits with RNA point cloud specificity. I show that this method is on par with carefully designed, hand-crafted feature sets.

Part III

Applications

6

RNA localization by the numbers

Abstract:

FISH-quant was applied in three high content screening studies to extract quantitative evidence from smFISH images. In the first application, a classification pipeline is developed to identify five different patterns of RNA localization: intranuclear, nuclear edge, perinuclear, foci, and protrusion. In addition, a novel mechanism of spatial control of gene expression, called translation factories, is characterized. The second application focuses on the centrosomal pattern. A modified analysis pipeline is used to detect centrosomes and study the spatial and temporal dynamics of several genes. The third application analyzes the protrusion pattern, implementing dedicated features to study the role of the motor protein KIF1C in the transport of some APC-dependent transcripts to cell extensions. In total, these studies involve several dozen transcripts, analyzed through 100,000 individual cells.

Résumé:

FISH-quant a été appliqué dans trois études de criblage à haut débit pour obtenir des analyses quantitatives à partir d'images smFISH. Dans la première étude, un pipeline de classification supervisée est développé pour identifier cinq schéma de localisation de l'ARN : intranucléaire, membrane nucléaire, périnucléaire, foci et protrusion. En outre, un nouveau mécanisme de contrôle spatial de l'expression des gènes, appelé usines de traduction, est caractérisé. La deuxième étude se concentre sur le schéma centrosomal. Un pipeline d'analyse modifié est utilisé pour détecter les centrosomes et étudier la dynamique spatiale et temporelle de plusieurs gènes. La troisième étude analyse le schéma de protrusion, en développant des indicateur dédiées pour étudier le rôle de la protéine motrice KIF1C dans le transport de certains ARNs vers les protrusions cellulaires. Au total, ces études portent sur plusieurs dizaines de gènes, analysés sur 100000 cellules individuelles.

Contents

6.1 A systemic quantification of RNA localization 97
6.1.1 Introduction 97
6.1.2 Materials and methods 98
6.1.3 Results 102
6.2 Local translation and translation factories 107
6.2.1 Introduction 107
6.2.2 Materials and methods 108
6.2.3 Results 109
6.3 A translation and cell cycle dependent centrosomal pattern . . 111
6.3.1 Introduction 111
6.3.2 Materials and methods 111
6.3.3 Results 114
6.4 The key role of KIF1C in protrusion pattern 116
6.4.1 KIF1C and protrusion mRNAs 116
6.4.2 Quantification of peripheral mRNAs 116
6.5 Conclusion 117

In this chapter, I present several applications of the pipeline described in the previous chapters. More specifically, I analyze experimental datasets in order to explore, quantify and validate biological insights about mRNA localization. Results from this chapter are computed from three different high content screening studies, totaling tens of genes observed through thousands of bioimages.

In the first section, I describe the development of a general classification pipeline to identify generic mRNA localization patterns from smFISH images [Chouaib *et al.*(2020)]. In the second section, exploiting the same dataset, I focus on a specific and novel clustering pattern: translation factories. These first two sections describe the work published in:

R. Chouaib, A. Safieddine, et al. (2020), *A dual protein-mRNA localization screen reveals compartmentalized translation and widespread co-translational RNA targeting*, Developmental Cell 54 (6), 773.

In the third section, we used a GFP channel to visualize and detect centrosomes. I observed a centrosomal localization patterns for different transcripts and mitosis phases [Safieddine *et al.*(2021)]. This work was published in:

A. Safieddine, E. Coleno, et al. (2021), *A choreography of centrosomal mRNAs reveals a conserved localization mechanism involving active polysome transport*, Nature Communications 12 (1), 1352.

In the fourth section, I detail several transcripts with a localization to cellular protrusions [Pichon *et al.*(2021)]. This last section mainly describes results presented in:

X. Pichon, K. Moissoglu, et al. (2021), *The kinesin KIF1C transports APC-dependent mRNAs to cell protrusions*, RNA 27 (12), 1528.

6.1 A systemic quantification of RNA localization

In [Chouaib *et al.*(2020)], I implemented a quantitative pipeline to analyze the localizations of mRNA molecules. This work lays the foundation of the Python packages developed in FISH-quant v2 [Imbert *et al.*(2022b)]. My contribution consists in classifying individual cells with one or several RNA localization patterns.

6.1.1 Introduction

Most of mRNAs have a random distribution throughout the cytoplasm, but some localize in specific subcellular regions [Blower(2013), Jung *et al.*(2014), Eliscovich et Singer(2017), Bovaird *et al.*(2018)].

Such localization can be related to either RNA metabolism, for example with untranslated mRNAs stored and repressed in processing bodies (P-bodies) (but not degraded [Hubstenberger *et al.*(2017)]), or protein metabolism with locally translated proteins. This local synthesis concerns both mature proteins and nascent peptides.

Different cellular processes imply the delivery of mature proteins in specific subcellular regions and a local regulation of the proteome. Local translation can contributes to cell fate determination during metazoan development, as observed in [Melton(1987)] with a clear modification of the Vg1 RNA spatial distribution between mature and immature Xenopus oocytes. In mammalian cells, RNA localization influences cell polarization and motility, usually through actin localization at the cell edge [Lawrence et Singer(1986)]. Locally translated proteins are also known to be involved in axonal growth and so neural plasticity [Van Driesche et Martin(2018)]. Finally, by precisely controlling the mRNA localization and the subsequent translation process, cells can avoid to release proteins at inappropriate places [Müller *et al.*(2013)] and help the synthesis of protein complexes [Pichon *et al.*(2016)].

Several mechanisms drive the localization of mRNAs. Sometimes the nascent peptide can serve as a targeting signal, but most of the time the RNA molecule itself will initiate its localization. The molecule often include a zip-code sequence that is read by a RBP and starts the assembly of a transport complex comprising different organelles or motor proteins. Such complex can then directly transport the RNA along the cytoskeleton [Blower(2013)]. Coupled with an anchoring mechanism at the targeted destination, it provides a better stability of the RNA localization. Following a random diffusion across the cell, a transcript can also be trapped in specific subcellular compartments. Likewise, a local protection from degradation will bias the RNA spatial distribution.

In a pioneering work on Drosophila embryogenesis [Lécuyer *et al.*(2007)], authors analyze 3,370 genes and find that 71% of them encode a transcript with a non-uniform localization pattern. In human cell lines recent studies exploit and improve smFISH techniques to image thousands of mRNAs in the perinuclear region, the mitochondria and the cell membrane [Battich *et al.*(2013), Chen *et al.*(2015), Eng *et al.*(2019), Xia *et al.*(2019)]. In [Chouaib *et al.*(2020)], we explored more systematically the link between non-random RNA localization and local translation in HeLa cells. We performed a quantitative analysis leveraging supervised and unsupervised methods in order to recognize up to 6 different localization patterns. The current section emphasizes this quantitative pipeline that contributes to make our analysis scalable and more robust.

6.1.2 Materials and methods

The quantitative pipeline used in this study and the existing work in FISH-quant v1 are the building blocks of FISH-quant v2. Therefore, the various methods presented here were not yet packaged in *bigfish*, but Python libraries that formed its foundation were already in used. This work was developed in Python and the code repository is public[1].

Experimental data

The study is a dual-color screen, where all genes were GFP tagged, to measure the protein, and the same oligos, targeting the GFP sequence, were used for the smFISH. Our collaborators first manually inspected 500 genes to identify potential transcripts with

---

[1] `https://github.com/Henley13/paper_translation_factories_2020`

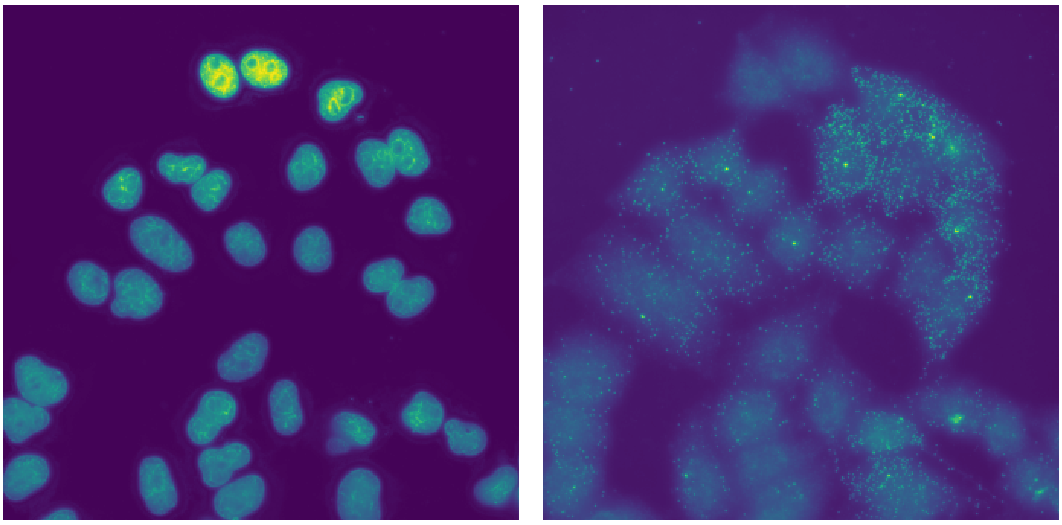


Figure 6.1: Contrasted image with Dapi (*left*) and smFISH (*right*) channels. Images are projected in 2D. Targeted transcript is DYNLL2. Plot built with *bigfish*

non random localization patterns. To obtain a more quantitative description of these data, we also developed a pipeline to systematically classify RNA localization patterns. To this end, I exploited a dataset of 526 FoVs, consisting of 3D stacks (z-spacing of 0.3μm) with DAPI and smFISH as illustrated in Figure 6.1. Acquisitions were performed with a Zeiss Axioimager Z1 widefield microscope or a Nikon Ti fluorescence microscope. In total, this dataset was built from 57 independent experiments gathering observations from 27 different mRNAs under different experimental conditions.

Semi-automated RNA detection

RNA detection was performed with a Python implementation of FISH-quant v1 [Mueller *et al.*(2013)]. In short, I applied a LoG filter on the 3D smFISH images, then a local maximum detection algorithm to localize individual RNA molecules. This detection requires a manually set intensity threshold for every experiment to discriminate the actual RNAs from the noisy background. Large agglomeration of spots were decomposed with a Gaussian mixture model, based on previous work [Samacoits *et al.*(2018)]. Ultimately, RNA foci were detected with the DBSCAN algorithm [Ester *et al.*(1996)] applied on the detected spot positions: a foci is then defined as a set of at least 5 RNAs with a maximum distance of 350 nanometers between RNAs belonging to the same foci. Foci overlapping the nuclear area in the projected 2D images were considered as a transcription site and removed from the analysis. Percentages of RNA in foci were then calculated as number of RNA inside the foci divided by the number of cytoplasmic RNAs.

I would like to highlight two main differences compared to the current detection methods implemented in FISH-quant v2 (presented in Chapter 3), both of which I improved: a detection threshold has to be set manually and the decomposition of agglomerated spots has since been simplified.

Cell and nucleus segmentation

Nuclear segmentation was performed from the DAPI channel and cellular segmentation from the cell autofluorescence in the smFISH channel. Segmentation was performed in 2D for both nuclei and cells, thus 3D images are projected in two dimensions using their maximal local focus values [Tsanov *et al.*(2016)].

Nuclei were segmented with NucleAIzer [Hollandi *et al.*(2020)], a deep neural network pipeline trained with the annotations from the Data Science Bowl 2018 challenge[2]. This pipeline is based on a Mask R-CNN architecture [He *et al.*(2017)], with optionally a fine-tuning of the segmented boundaries with a U-Net model [Ronneberger *et al.*(2015)]. After a first round of segmentation, some nuclei were missing, so I removed the segmented nuclei from the DAPI channel and re-analyzed the image with the remaining nuclei again with NucleAIzer. Removing the segmented nuclei from the original image implied the morphological reconstruction technique described in Chapter 4. This technique is now implemented in FISH-quant.

Cells were then segmented with a watershed algorithm using the nuclei masks as seeds. A threshold value was set manually for every experiment to discriminate the cell surface from the background. In case of poor results, some segmentation masks were corrected manually. This was especially the case for transcripts that tend to localize in cell protrusions. Further, accurate cellular segmentation is especially important for this localization pattern (also because only RNAs detections overlapping segmented cells are considered in our analysis).

Localization feature engineering

Based on the segmentation masks and the coordinates of the detected spots, I identified 9,710 individual cells with an average of 346 RNAs per cell. For each cell, I collected the cell and nucleus masks in 2D, the RNA and the potential cluster coordinates in 3D. In addition I saved an image of the cell, for visualization purposes, and different information about the experiment (the presence of a treatment, the targeted gene, etc.). Cropped cells on the edge of the image and cells with less than 30 detected RNA inside were removed. At this point, I had a coordinate representation of every cell, as presented in Chapter 5.

I designed and computed a set of 15 localization features to describe the spatial distribution of RNAs inside the cell:

- The number of foci.
- The proportion of RNAs inside foci.
- The proportion of RNAs inside the nucleus.
- The average RNA distance to the cell membrane, normalized by the value obtained under a uniform RNA distribution.
- The average RNA distance to the nucleus membrane, normalized by the value obtained under a uniform RNA distribution.

---

[2] `https://www.kaggle.com/c/data-science-bowl-2018`

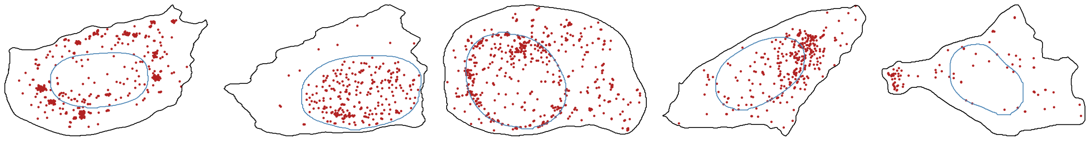

(a) Foci (b) Intranuclear (c) Nuclear edge (d) Perinuclear (e) Protrusion

Figure 6.2: RNA localization patterns from [Chouaib *et al.*(2020)]. Coordinate representations with RNA spots (*red*), cell membrane (*black*) and nuclear membrane (*blue*). Detection and segmentation results are extracted and visualized with *bigfish*

- The average foci distance to the cell membrane, normalized by the value obtained under a uniform foci distribution.
- The average foci distance to the nucleus membrane, normalized by the value obtained under a uniform foci distribution.
- The proportion of RNAs inside cell protrusions. A protrusion region is defined by calculating the difference between the segmented cellular region and its morphological opening with a large window.
- The peripheral dispersion index, defined as the squared point distance to the centroid of the cell and normalized by the value obtained under a uniform RNA distribution.
- The number of RNAs within 515 nm from the nucleus membrane, normalized by the value obtained under a uniform RNA distribution.
- The number of RNAs between 515 nm and 1030 nm from the nucleus membrane, normalized by the value obtained under a uniform RNA distribution.
- The number of RNAs between 1030 nm and 1545 nm from the nucleus membrane, normalized by the value obtained under a uniform RNA distribution.
- The number of RNAs between 0 nm and 515 nm from the cell membrane, normalized by the value obtained under a uniform RNA distribution.
- The number of RNAs between 515 nm and 1030 nm from the cell membrane, normalized by the value obtained under a uniform RNA distribution.
- The number of RNAs between 1030 nm and 1545 nm from the cell membrane, normalized by the value obtained under a uniform RNA distribution.

The size of the concentric regions for the last 6 features were determined to be 5 pixels based on visual inspection of the samples. A conversion to nanometer (with a pixel-size of 103nm) results in the reported ranges.

Binary classification models

I used these hand-crafted features to train several binary classifiers, one for each localization pattern we wanted to recognize: foci, intranuclear, nuclear edge, perinuclear, protrusion. Random localization was considered by default. Figure 6.2 illustrates an example of these patterns with a coordinate representation.

To train the classifiers, manual annotations were needed as a ground truth. I generated panels with the cropped original image of the individual cells and their coordinate representations. These panels were then manually tagged with the appropriate localization pattern by several experienced microscopists. This resulted in a dataset of 810 annotated cells (Table 6.1).

Pattern	# of cells
Random	372
Foci	198
Intranuclear	73
Nuclear edge	87
Perinuclear	64
Protrusion	83

Table 6.1: Annotated cells

A first evaluation of how well our localization features describe the different localization classes was the visualization of the feature space. To do so, I reduced the dimensionality with a t-distributed stochastic neighbour embedding (t-SNE) transformation [van der Maaten et Hinton(2008)] in order to visualize the features point cloud. I initialized the t-SNE with a PCA transformation and used a perplexity value of 30. From the original 15-dimensional feature space, I thus obtained a 2D vector representation that could be easily inspected.

I also defined a supervised learning problem with the training of 5 independent binary Random Forest classifiers [Breiman(2001)]. Such a random forest classifier is an ensemble model of tree classifiers. Each tree is trained on a subsample of the observations and a subset of features. This ensembling framework makes random forest quite robust to overfitting. The choice to design the problem as several binary classifiers instead of one multi-class problem allowed me to define the localization patterns as non mutually exclusives. Indeed, an individual cell can display several patterns at the same time, like RNA clusters (foci) localizing around the nuclear membrane (nuclear edge). For each model, I then built a training set including all the cells of one class and a subsampling of cells from others classes, such that the imbalance is 1:4 for the positive class. This is a so called ”one vs. all” training strategy. For every sample, an Out-of-bag (OOB) prediction can be computed using only the trees fitted without the sample. In such manner, the model can ”be fit in one sequence, with cross-validation being performed along the way.” [Hastie *et al.*(2009)]. I initialized the random forests with 100 trees, a maximal depth of 3 and a minimum number of samples per splitter node of 2. During the training, for each split, I considered a subset of 10 features and entropy criterion. In addition, input dataset was rescaled to have zero mean and unit variance.

6.1.3 Results

It is widely known that RNAs levels of a given gene can vary substantially from one cell to another. Thanks to their design and/or their normalization, our spatial features are mostly invariant to RNA concentration. They enable the use of unsupervised or supervised methods to classify cells among several localization patterns without the compounding impact of gene expression variability.

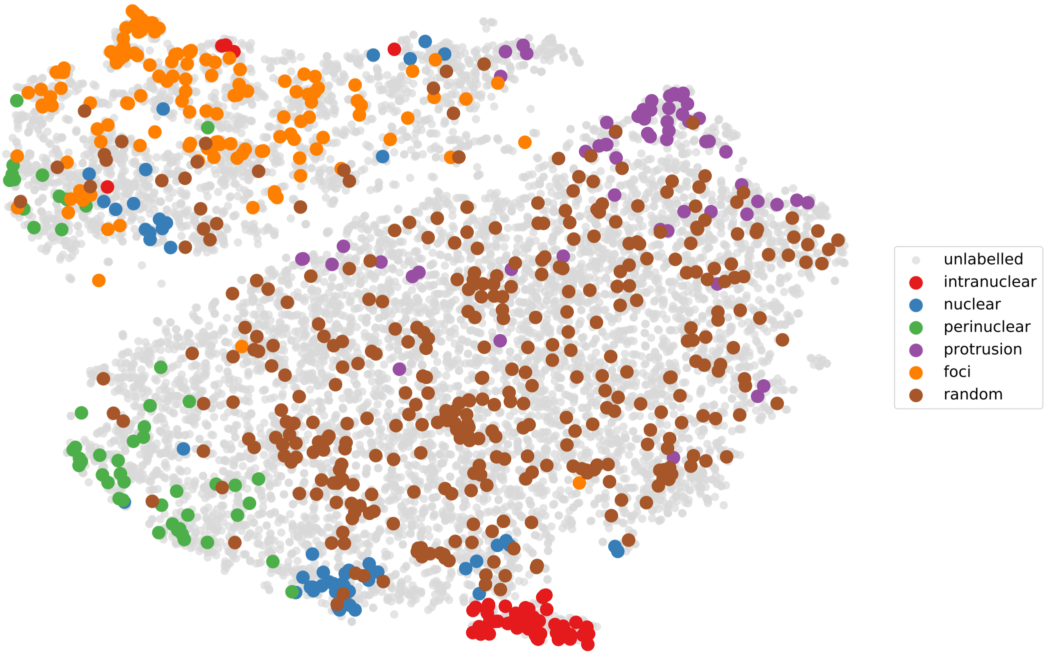


Figure 6.3: t-SNE embedding from [Chouaib *et al.*(2020)]. Each point is a cell in the feature space after a t-SNE transformation. Manually annotated cells are colored according to their localization pattern. Cells that were not annotated or that had multiple patterns are colored in gray

Unsupervised visualization

The resulting embedding from the t-SNE algorithm can be observed in Figure 6.3. Cells with different patterns are localized in different regions of the embedded feature space. Moreover, cells with the same annotations, cluster in the same regions independently of their targeted gene. This suggests that the hand-crafted features capture key information of the localization patterns.

Initializing the t-SNE algorithm with a PCA transformation results in a better preservation of the global data structure, thus enabling more relevant global interpretation. In particular, we can discriminate at the top a large cluster of cells with a foci pattern from the rest of the cell population. The resulting embedding also seems to polarize between nucleus-related patterns (bottom left) and the cell-related patterns (top right). This polarization remains even among the subpopulation with potential foci pattern.

Such conclusion validates the design to let a cell being classified with several non-exclusive localization patterns. Indeed, a foci can be localized in one of several relevant subcellular compartments.

Supervised classification

The result presented above showed that the feature space discriminates well between localization classes. Next, I wanted to test how well these features perform when used in a random forest classifier.

For this, I computed 5 binary predictions, one per pattern, for each cell. A pattern was then assigned to a cell if the probability given by the random forest classifier for that pattern was higher than 0.5. If no pattern was detected, the cell was classified with a random localization pattern. In total, I analyzed 27 different genes in a total of 9,710 cells.

The OOB accuracy score obtained for the different patterns was between 0.93 and 0.99, while a dummy classifier only returned a 0.8 accuracy score (the dataset was built with 20% positive samples). Among all patterns, intranuclear localization was the easiest to detect. On the other hand, the nuclear edge and perinuclear patterns were the most difficult to classify due to possible confusion with a random pattern or the lack of annotated cells.

The total number of cells per gene varied, thus predictions were aggregated for each gene. This aggregation also facilitated data inspection and a first comparison among genes. For every gene, I computed the proportion of cells displaying each indicated localization pattern. Results were eventually reported in a heat map 6.4, for every gene and pattern. Importantly, row values did not sum to one because classifiers (and so columns predictions) were independent.

We manually identified a group of random genes that could be used as a control group: KIF20B, MYO18A, SYNE2 and PLEC. This control group permitted to perform statistical testing over the aggregated predictions. A Fisher's exact test can measure if the proportion of cells observed with a pattern is significantly greater than the proportion observed in the control group (with a p-value $< 10^{-3}$). Genes whose transcripts presented a significant localization preference are reported in Figure 6.4.

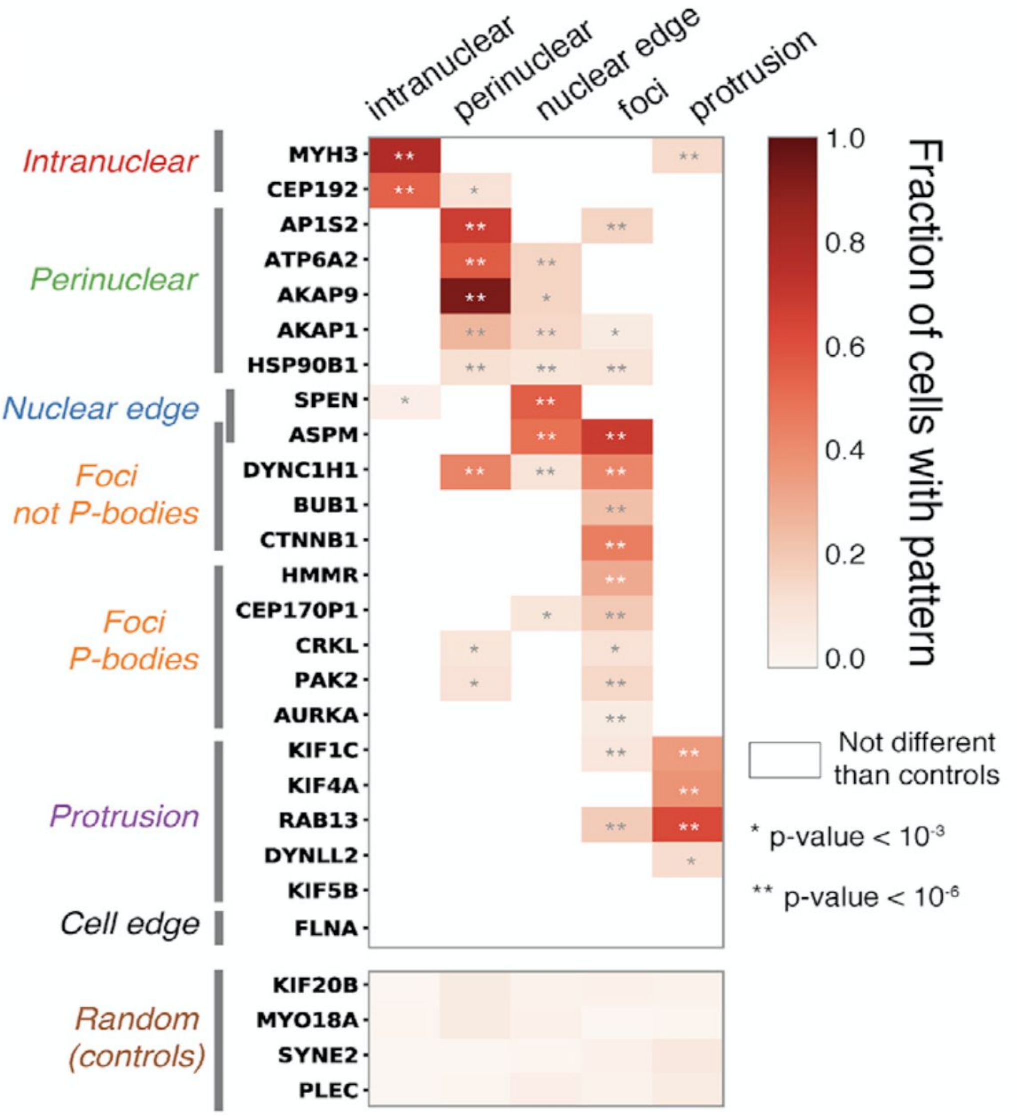


Figure 6.4: Heat maps from [Chouaib *et al.*(2020)] with the fraction of cells classified in the indicated pattern. (*Top heat map*) Genes whose RNAs shows a localization preference. Only values significantly different from negative controls are colored (p-value computed with Fisher's exact test). (*Bottom heat map*) Negative control with genes whose RNAs localize randomly. (*Left bars*) Manual annotations of the genes are indicated with the same color scheme than Figure 6.3

Pattern	Accuracy score
Random	-
Foci	0.95
Intranuclear	0.99
Nuclear edge	0.93
Perinuclear	0.93
Protrusion	0.94
Dummy classifier	*0.80*

Table 6.2: Random forest accuracy (OOB)

Overall, we observed consistent results between the automated analysis and the manual annotations performed by the biologists (the colored annotations in the t-SNE plot 6.3 and the vertical colored bars next to the heat map 6.4). The agreement between automated and manual classification held with a high degree of statistical significance, except for FLNA and KIF5B. More specifically, FLNA transcripts were manually annotated with a potential cell edge pattern. However, this localization pattern was difficult to recognize with the quantitative pipeline, mostly because the pattern is only visible in 3D while the cell segmentation, and the spatial features resulting from it, was in 2D. For these reasons we did not study in depth this localization pattern.

Recognition of five localization patterns

Apart from KIF5B, others transcripts localizing in cellular protrusions were correctly identified: KIF1C, KIF4A, RAB13 and DYNLL2. The frequency of this pattern varied substantially, between 14% and 62% of cells classified, respectively for DYNLL2 and RAB13. Interestingly, this pattern concerns four RNAs encoding motor proteins: the three kinesins KIF1C, KIF4A, KIF5B and DYNLL2.

MYH3, another transcript encoding a motor protein, localized partially in protrusions, although it is not its most frequent localization pattern. Most of the time, MYH3 transcripts remained in the nucleus, exhibiting an intranuclear pattern. This is the most easily recognizable pattern: more than 55% of cells spotting MYH3 and CEP192 RNAs showed an accumulation of transcripts inside nucleus.

The two genes observed with a nuclear edge pattern, ASPM and SPEN, had 50% and 55% of cells identified with this localization, respectively.

Several genes were identified with RNAs in the perinuclear area: AKAP1, AKAP9, AP1S2, ATP6A2, and HSP90B1. The number of cells displaying this pattern ranged from 13% (HSP90B1) to 93% (AKAP9). Even with a small number of cells, HSP90B1 showed a significant perinuclear pattern compared to the control group. In our study, we could then further show that HSP90B1 localizes to the the endoplasmic reticulum, and that this localization is translation dependent.

A large number of transcripts were found to localize in foci. We classified them in two groups, with a more detailed analysis in the Section 6.2. The first group contains transcripts accumulating in P-bodies: AURKA, HMMR, CEP170P1, CRKL and PAK2. The second group includes ASPM, DYNC1H1, BUB1 and CTNNB1 transcripts, accumulating in non-P-bodies foci we call translation factories. It is noteworthy to mention that a number of genes exhibited foci with another localization preference in parallel like ASPM (nuclear edge), DYNC1H1, AP1S2, AKAP1, HSP90B1 (perinuclear), KIF1C and RAB13 (protrusion). In general, non-P-bodies genes had a higher proportion of cells classified with a foci pattern: between 24% (BUB1) and 69% (ASPM) cells while P-bodies genes have between 7% (AURKA) and 31% (HMMR) cells. Likewise, in term of RNA content, non-P-bodies genes had a higher proportion of mRNA in foci, varying from 9% (BUB1) to 28% (CTNNB1), while P-bodies accumulated between 2% to 15% of mRNAs. This foci quantification is in line with the previously reported estimations in the literature of 10% to 20% mRNAs clustered in foci [Pillai *et al.*(2005), Hubstenberger *et al.*(2017)].

Cell-to-cell variability

We then focused on individual cells instead of the aggregated information per gene. This showed a remarkable heterogeneity of RNA localization. For example, for CTNNB1, we observed a binary behavior, with cells either having no foci at all or alternatively more than 60% of mRNAs clustered in foci.

For a more detailed quantification of this intercellular variability, I would like to refer the reader to Appendix D. Cells of specific genes are plotted on t-SNE D.1 and single cell results are summarized in heat maps D.2. Pattern probabilities returned by classification models varied from cell to cell. For a given gene, cells were dispersed on the t-SNE plot, although a majority of them still accumulated in the expected area. Finally, some cells had simultaneously several localization patterns. As mentioned above, this is frequently the case with cells having RNA foci, where foci can be either found in protrusion or close to the nucleus. Likewise, MYH3 cells frequently showed an intranuclear and protrusion pattern.

In conclusion, my quantitative pipeline confirmed the manual observations and measured a high degree of heterogeneity of RNA localization across individual cells. This variability is a general phenomenon, at least for HeLa cells, since it is seen with nearly all the genes analyzed in [Chouaib *et al.*(2020)].

6.2 Local translation and translation factories

Beyond the generic pattern recognition pipeline I have implemented, another important quantitative result is related to RNA and their implication in translation, a phenomenon we termed *translation factory.* This analysis is part of a broader investigation about the co-localization of mRNAs and their encoded proteins, a major contribution of [Chouaib *et al.*(2020)]

6.2.1 Introduction

In contrast to previous studies on RNA localization [Lécuyer *et al.*(2007), Battich *et al.*(2013), Chen *et al.*(2015), Eng *et al.*(2019), Xia *et al.*(2019)], our experimental design also permitted to detect translated proteins. In brief, we used a library of HeLa cell lines, were in each cell line a Bacterial artificial chromosome (BAC) including a GFP-tagged gene [Poser *et al.*(2008)] was introduced. This allows not only to visualize the RNA with FISH probes again the GFP signal, as exploited above in the study of RNA localization, but also to visualize the translated protein in the GFP channel. In [Chouaib *et al.*(2020)], we performed a high-content screen in HeLa cells and visualize simultaneously the mRNA and its encoded protein.

Along with RNA localization patterns, we observe occurrences of local translation in several subcellular areas. For example, the phenomenon is observed in the nuclear membrane with the ASPM and SPEN transcripts. Some local translations are expected, like HSP90B1 (an endoplasmic reticulum protein), ATP6A2 (in endolysosomes) or AKAP1 (an RBP at the surface of mitochondria). Others are discoveries like AKAP9 or AP1S2, whose mRNAs are known to localized respectively in the Golgi or on endosomes.

Our collaborators further used the SunTag system, which allows to visualize the presence of nascence peptide chains, and hence ongoing translation [Pichon *et al.*(2016), Pichon *et al.*(2018),Wu *et al.*(2016)]. While this required genetic engineering, to include the SunTag sequence into the gene of interest, it provides important information about co-translational targeting. Here, this technique allows us to confirm that both ASPM mRNAs and nascent proteins localized in nuclear membrane, thus suggesting a local translation.

The remainder of this section will detail our analysis of the translation-dependent foci pattern. When investigating the RNAs that form foci, we found that most colocalize with P-body markers. Such localization in P-bodies has been linked to RNA storage or degradation. In contrast, we found four transcripts that accumulate in foci, but do not colocalized with P-body markers: BUB1, DYNC1H1, CTNNB1 and ASPM. Interestingly, we found that these foci disappear when cells were treated with a translation inhibitor. Further analysis revealed a co-localization of RNA foci and nascent proteins 6.5. Taken together, this suggests that these foci bring together RNAs, which are locally translated. We hence named these complexes *translation factories*, where specific mRNAs accumulate to be translated, and not repressed. My second contribution to [Chouaib *et al.*(2020)] was the quantification of this unexpected phenomenon, exploiting my cluster detection methods and experiments with a translation inhibition protocol.

6.2.2 Materials and methods

In the continuation of the work presented in Section 6.1, the quantification was developed in Python and available online[3].

Translation inhibition

To analyze the role of translation in mRNA localization, we performed several experiments where translation is inhibited by either Puromycin or Cycloheximide. Coupled with my computational pipeline, we could then measure the impact of a translation inhibition.

While both drugs inhibit translation, they differ in the their mode of action: Puromycin inhibits translation by releasing the nascent peptide chain from the ribosomes, while Cycloheximide freezes the ribosomes on the mRNAs. Importantly, the latter results in the nascent peptide chain still being present at the translated RNA. If RNA localization patterns are impacted by these drug treatments, this localization is likely driven by translation. Furthermore, these two treatments allow to clarify if a translation-dependent localization requires the nascent peptide chain. In the case of ASPM gene, we observed for example that the localization patterns (nuclear edge and foci) disappeared with Puromycin, but not with Cycloheximide. It suggests that the mRNA localization requires the nascent protein.

---

[3] `https://github.com/Henley13/paper_translation_factories_2020`

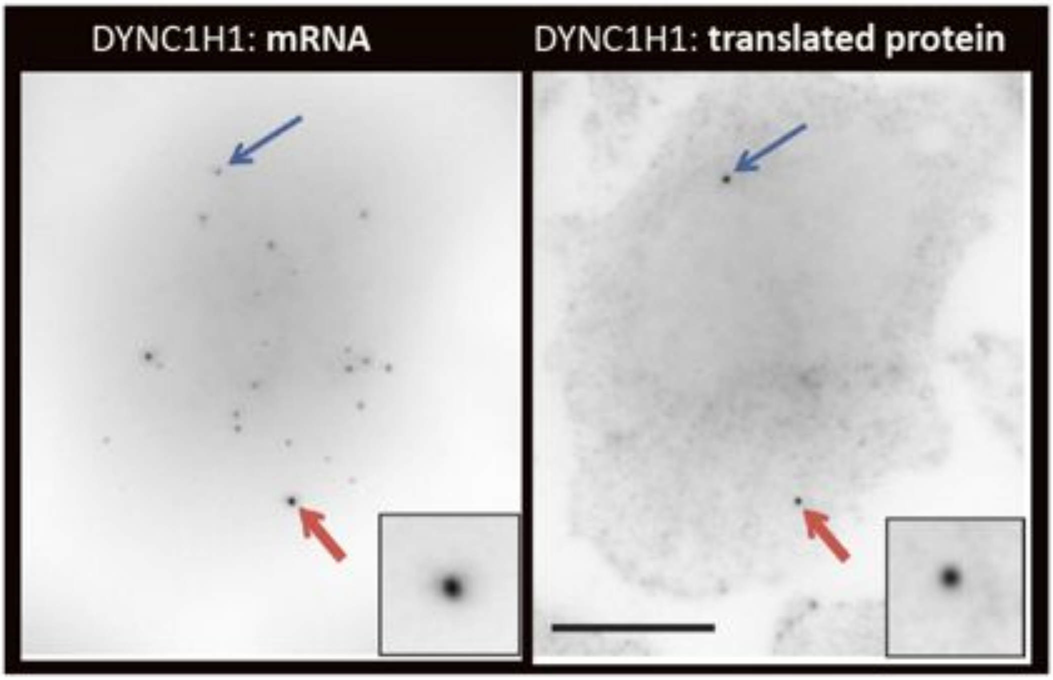


Figure 6.5: Colocalization between the DYNC1H1 transcript and its nascent protein, from [Pichon *et al.*(2016)]. Transcripts and proteins are targeted with smFISH probes (*Left*) and SunTag (*Right*) respectively. *Scale bar* is 10μm

Foci detection

Both on the treated and untreated cells, I applied my spot detection algorithms, the decomposition algorithm for RNA aggregates, and finally the RNA cluster detection (see Subsection 6.1.2). It is important to note that the cluster detection is distinct from the decomposition of dense areas. Such areas can indeed be processed in a way that no cluster is detected afterward.

6.2.3 Results

Translation-dependent localization

An obvious indicator to assess if a foci pattern is disrupted is the number of RNA foci detected per cell. When I compared treated and untreated cells, the difference is clear for the four genes BUB1, DYNC1H1, CTNNB1 and ASPM. Their transcripts usually accumulate in foci, but with the presence of Puromycin, mRNAs are dispersed within the cell and the number of detected foci drops. An analysis with a finer granularity is detailed in Appendix D and especially in Figure D.1, where changes on the single cell level are reported. In addition, a dual-color smiFISH allowed us to visualize two different transcripts at the same time and showed that the foci detected for these four genes are not overlapping. These are hence distinct RNA complexes.

As a comparison, in Figure 6.6, I performed the same analysis with HMMR, a gene found in P-bodies. The difference is striking, as the foci pattern seems unchanged when translation is inhibited.

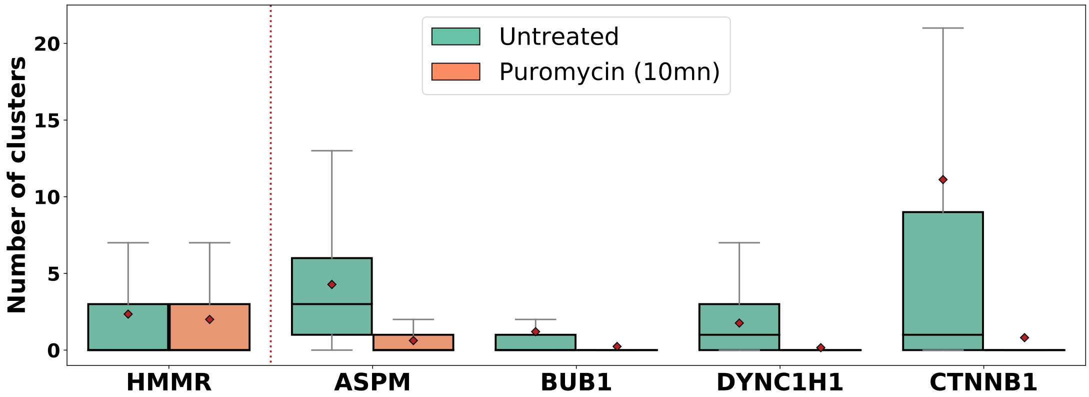


Figure 6.6: Box plot with the number of detected RNA foci for different genes and treatments. Red diamonds are the mean and the *whiskers* equal 1.5 the interquartile range

Taken together, this leads us to the conclusion that for these four genes, the RNA foci are acting as translation factories.

CTNNB1: translation and degradation

Among the four genes with translation factories, cytoplasmic foci of CTNNB1 mRNA present a surprising regulation mechanism. This mRNA encodes the β-catenin protein, which is the main transcription factor of the Wnt signaling pathway [Grainger et Willert(2018)]. The later is known to play a role in carcinogenesis and embryogenesis. The Wnt signal allows the β-catenin protein to translocate and accumulate in the nucleus in order to activate targeted transcriptions.

The existence of translation factories offers a new insight on the dynamic at stakes in the presence of Wnt. β-catenin is degraded in a ”destruction complex” that involves APC, AXIN, in addition to the kinases CK1α and GSK3 [Stamos et Weis(2013)]. When the Wnt pathway is activated, these molecules are recruited to the cell membrane and can no longer interact with β-catenin, thus stopping its degradation. As a consequence, we observed that β-catenin is highly expressed and accumulates in the nucleus. Interestingly, in the presence of Wnt, we also observed that the CTNNB1 mRNA foci disappear. This phenomenon suggests a relation between the degradation of β-catenin and foci formation. In [Chouaib *et al.*(2020)] we demonstrated that CTNNB1 translation factories rely on the nascent peptide chain, but it seems they also require the ”destruction complex”. When the different components of this complex are disrupted, the mRNA foci tend to disappear. Moreover, these components accumulate in the foci. CTNNB1 translation factories are thus sites of both β-catenin synthesis and degradation. As [Chin et Lécuyer(2020)] summarizes it, these factories are ”sites of co-translational protein degradation, where CTNNB1 protein rapidly comes into contact with the destruction complex, offering an elegant mechanism to tightly control the cellular levels of potent signaling pathway effectors”.

6.3 A translation and cell cycle dependent centrosomal pattern

For the analysis in [Safieddine *et al.*(2021)] I could further validate and refine my analysis pipeline, and already automated workflows permitted a substantially faster analysis. Specifically, it benefits from the development of FISH-quant v2 and the improvements published in the literature about cell segmentation [Imbert *et al.*(2022b), Stringer *et al.*(2021)]. My contribution to this study was the quantification of the centrosomal localization pattern observed for some mRNAs.

6.3.1 Introduction

This study is a follow-up of the above described screen, with the goal to link RNA localization and translation at centrosomes.

Centrosomes have an important role in cell division, in addition to regulate cell motility and polarity [Wu et Akhmanova(2017)]. Before cell division, the centrosome duplicates. As soon as division begins, the two centrosomes move to opposite ends of the cell. Microtubules then assemble into a spindle between the two centrosomes, which then helps to separate the replicated chromosomes into the two daughter cells.

First observations of a mRNA accumulation at the centrosomes were made on Xenopus early embryos [Groisman *et al.*(2000)], with cyclin B1 mRNAs concentrating at the mitotic apparatus. Subsequent approaches of microscopy with FISH techniques allow to image mRNA localization in Drosophila [Lécuyer *et al.*(2007), Wilk *et al.*(2016)] and human cell lines [Sepulveda *et al.*(2018), Chouaib *et al.*(2020)]. As these mRNAs encode for known centrosomal proteins, these studies suggest a phenomenon of local translation.

In [Safieddine *et al.*(2021)], we identified 8 mRNAs with a centrosomal pattern. Remarkably, these mRNAs displayed a RNA localization pattern with a dependence on the cell cycle. Their localization appear to be translation dependent, through the synthesis of a nascent protein, and cell cycle dependent. Moreover, these mRNAs conserved their properties in both human and drosophila cell lines.

6.3.2 Materials and methods

The quantitative analysis presented here exploits an improved version of *bigfish*, especially for the spot detection. This work was developed in Python and the code repository is public[4].

Experimental data

Image acquisition was performed on an Opera Phenix High-Content Screening System (PerkinElmer). We acquire 3D images with around 35 z-slices and a z-spacing of 0.3μm. In total, and after curation, we collected 3,678 FoVs of HeLa cell line for 12 different transcripts, with control experiments based on treatment with translation inhibitors Puromycin or Cycloheximide. In addition to the smFISH channel, each image stack

---

[4] `https://github.com/Henley13/paper_centrosome_2020`

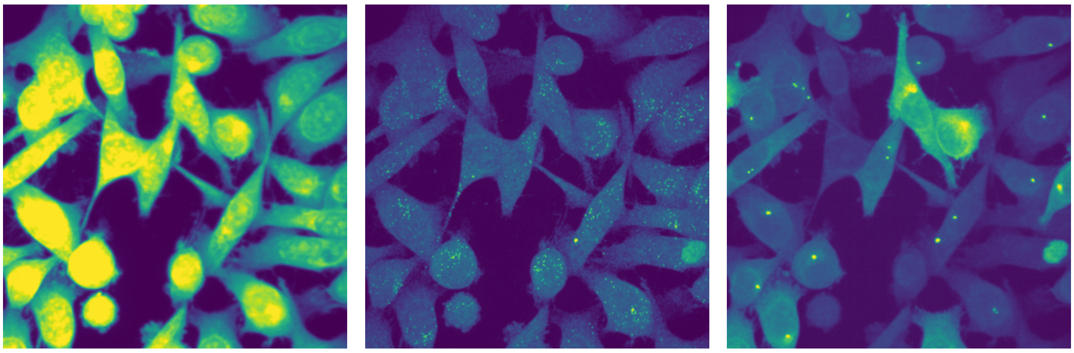


Figure 6.7: Contrasted image with CellMask™ (*left*), smFISH (*center*) and GFP (*right*) channels. Targeted transcript is BICD2. Images are projected in 2D. Plot built with *bigfish*

includes three fluorescent labels to visualize the nucleus (DAPI), the cell (CellMask™) and the centrosomes (Centrin1-GFP) as illustrated in Figure 6.7.

RNA and centrosome detection

In [Safieddine *et al.*(2021)], and for the first time, I used the threshold-free spot detection described in Chapter 3. This heuristic approach permitted me to automatically detect RNA in thousands of images. RNA detection was followed by the decomposition of dense areas and the detection of RNA foci, everything being processed from the smFISH channel projected in 2D.

The novelty in this paper is the detection of centrosomes from a GFP channel (projected in 2D). Since the centrosome appears as a much larger spot than the individual RNA molecule, I applied my RNA cluster detection method with fine-tuned parameters. This worked robustly, and allowed for an automated centrosome detection (Figure 6.8).

Cell and nucleus segmentation

Nucleus and cell segmentation was performed with a pretrained Cellpose model [Stringer *et al.*(2021)] from the 2D DAPI and CellMask™ channels, respectively . CellPose is based on a modified U-Net architecture [Ronneberger *et al.*(2015)] and trained on a large and diverse dataset. The performance of a deep learning model like Cellpose makes the segmentation step scalable with little or no manual intervention. I found that Cellpose is quite sensitive to the set average size of the segmented structure. For our data, I used an average diameter of 14.4μm for nuclei, 22.7μm for cells. In post-processing step, I matched these segmentation results, to guarantee that properly segmented cells contain one nucleus. Isolated cells or nuclei were removed in this step. Lastly, I only kept cells with 1 or 2 centrosomes. This resulted in 54,263 individual cells which were used for further analysis.

Centrosomal features and cell cycle classification

I adapted the spatial features developed for [Chouaib *et al.*(2020)] to characterize a centrosomal RNA localization at the single-cell level. In particular, I defined the cen-

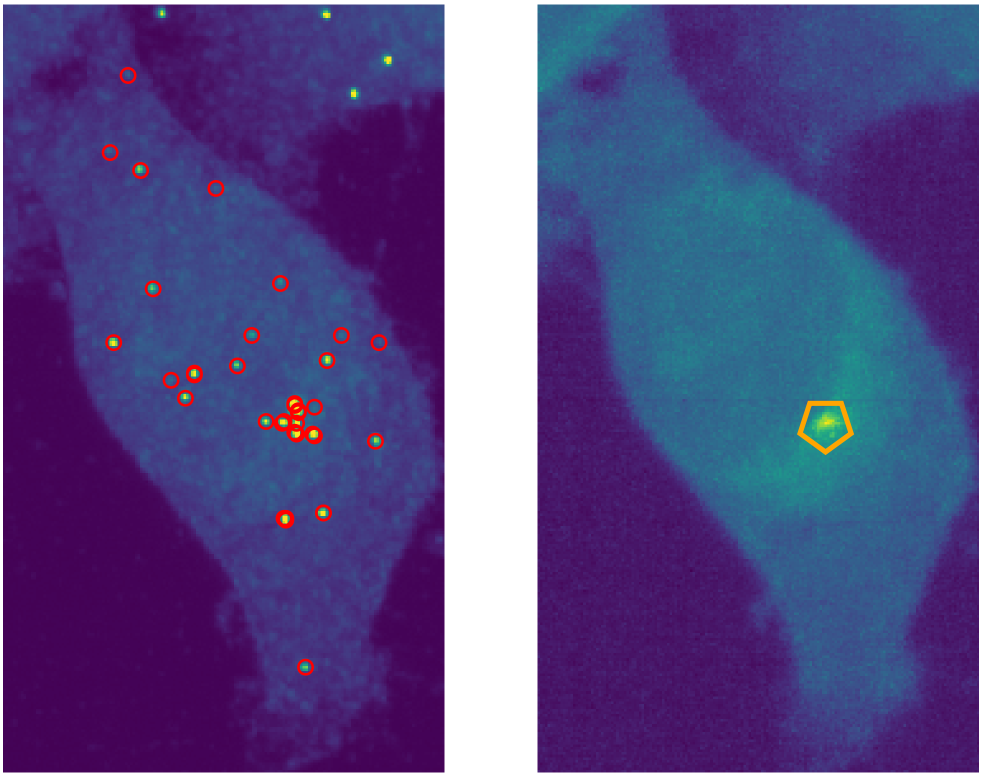


Figure 6.8: Contrasted image with detected RNAs on a smFISH channel (*left*) and detected centrosome on a GFP channel (*right*). Targeted transcript is BICD2. Plot built with *bigfish*

trosomal region as a disk with an empirically set radius of 2000nm around the detected centrosomes. From there, I computed any potential accumulation of RNAs within centrosome's neighborhood, normalized by the expression level observed for each cell. An example of such accumulation can be seen in Figure 6.8. A complete list of the different implemented features is described in Section 5.2.2 and illustrated in Figure 5.3.

We speculated that RNA localization of some genes might be strongly cell-cycle dependent. In order to classify the cell cycle, we computed nucleus morphological features from the DAPI channel and the segmentation results using CellCognition [Held *et al.*(2010)]. Completed with manual annotations it allowed us to discriminate interphase (DNA replication), early mitosis (preparation for chromosome division) and late mitosis stage (cell division).

6.3.3 Results

Centrosomal mRNAs

My contribution in [Safieddine *et al.*(2021)] was to quantify centrosomal mRNA localization. Among the centrosomal protein-coding genes analyzed, we found 8 mRNAs accumulating in centrosome's neighborhood: BICD2, NIN, CCDC88C, PCNT, CEP350, HMMR, NUMA1 and ASPM. I designed a quantitative pipeline to process a high-content screening dataset and compute the proportion of mRNAs detected around a centrosome. The results are reported in Figure 6.9. In addition to the centrosomal transcripts, I also processed several control genes without the studied pattern. While TRIM59 and TTBK2 present no specific localization pattern, KIF1C and DYNC1H1 are respectively and frequently identified with protrusion and foci patterns (see Section 6.1).

Two main observations can be drawn from this plot. First, most of the centrosomal mRNAs display a higher proportion of transcripts around the centrosomes: from 4.4% (ASPM) to 19.7% (BICD2), against 4.0% for the control genes, in average. These results already illustrate a limitation of my quantification, since the gene ASPM displayed a lower proportion of centrosomal mRNAs than our control DYNC1H1 (5.8% in average). DYNC1H1 transcripts accumulate in foci close to the nucleus can thus can be easily confused with a centrosomal pattern.

Importantly, we can observe in Figure 6.9, that a while a Puromycin treatment resulted in a drop of centrosomal mRNAs localization, Cycloheximide treatment did not. Similarly to the translation factories, it suggests that the centrosomal pattern is driven by the presence of nascent peptide chains.

A cell cycle regulated localization

In order to get a more detailed view of this RNA localization, we implemented an image analysis approach to obtain the cell-cycle of each cell (interphase and early/late mitosis). Five mRNAs exhibited a centrosomal localization pattern during interphase and early mitosis: HMMR, BICD2, CEP350, PCNT and NIN. In addition, HMMR was the only gene whose mRNAs and proteins co-localized at cytokinetic bridge in telophase. Lastly, ASPM and NUMA1 localized only during mitosis and CCDC88C in interphase. However, while NUMA1 mRNAs localized during early mitosis (prophase

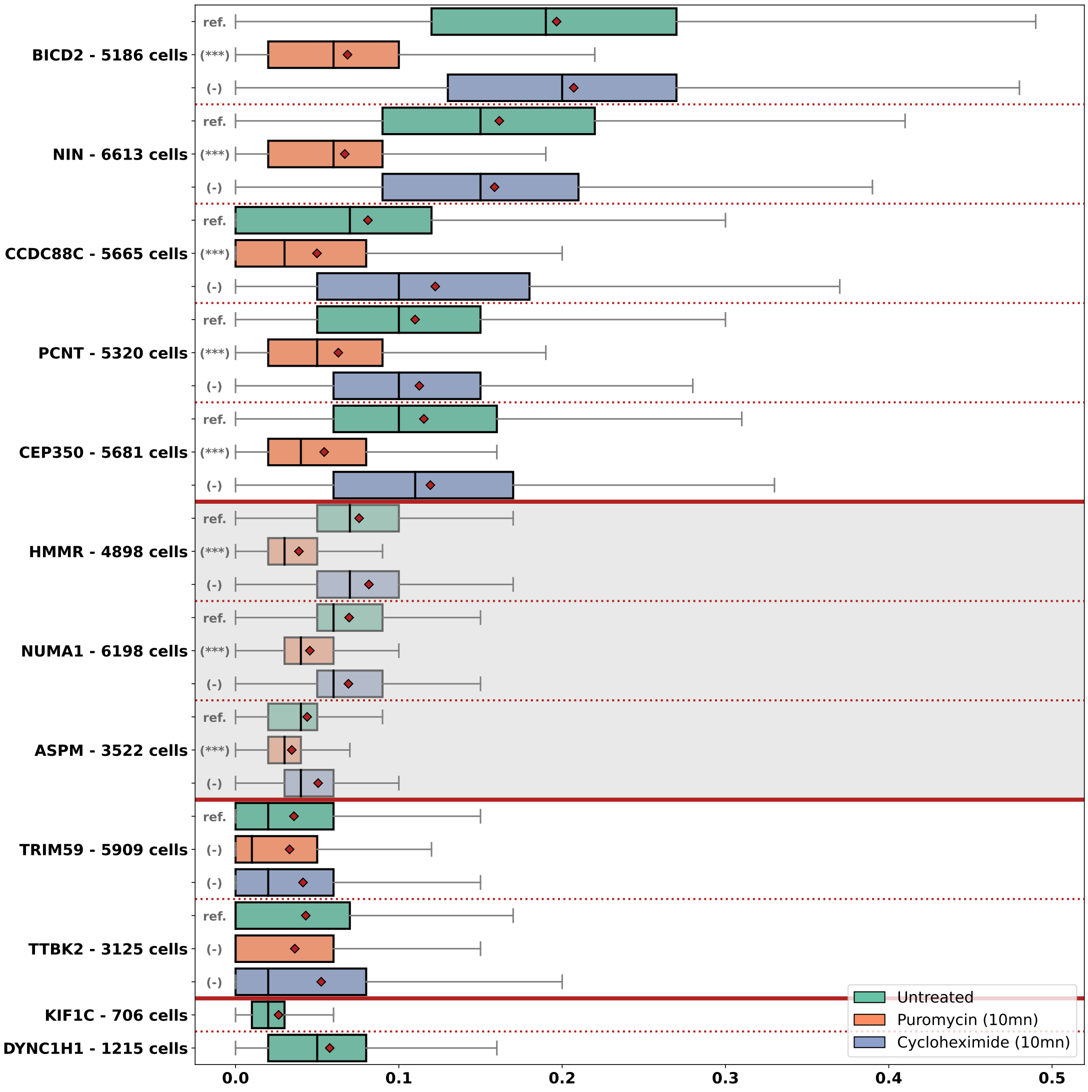


Figure 6.9: Box plot with the proportion of centrosomal mRNAs in cells (interphase and mitotic phase), for different genes and treatments. Red diamonds are the mean and the *whiskers* equal 1.5 the interquartile range. HMMR, NUMA1 and ASPM (in *gray*) are endogenous mRNAs, the rest are BAC-transcribed mRNAs. TRIM59, TTBK2, KIF1C and DYNC1H1 are used as control mRNAs. A one-sided Welchs t-test is used to evaluate significance (***: p-value < 0.001)

and prometaphase), a centrosomal accumulation of ASPM mRNAs can be observed for every mitotic phase. All in all, these results reveal an elegant *choreography* of mRNA localization, cell cycle regulated and translation driven.

6.4 The key role of KIF1C in protrusion pattern

In [Pichon *et al.*(2021)], I could take full benefit from the developed analysis pipeline to extract quantitative results from a smFISH high content screening study. My contribution for this study, was the quantification of a protrusion pattern manifested by the accumulation of mRNAs in cell extensions.

6.4.1 KIF1C and protrusion mRNAs

Three RNA transport mechanisms are reported in the literature [Medioni *et al.*(2012), Bovaird *et al.*(2018)]:

1. a localized protection from RNA degradation
2. a random diffusion coupled with a targeted entrapment
3. an active RNA transport along the cytoskeleton, with specific motor proteins

In this section, we focus on the active transport mechanism. As an example, in vertebrates, the β-actin mRNA has a *zipcode* sequence that the RBP ZBP1 will recognize, hence allowing the transport of the mRNA molecule along microtubules and actin filaments [Oleynikov et Singer(2003)].

More specifically, we investigated the role of the kinesin KIF1C that accumulates in cell protrusion (see Figure 6.4) and encodes a microtubule motor protein. This protein interacts with a specific group of mRNAs defined as APC-dependent (they require the APC protein to localize [Wang *et al.*(2017)]). KIF1C protein binds a subset of these mRNAs and transports them to cell extensions where they localize in RNA foci: NET1, TRAK2 and RAB13 . Interestingly, this kinesin protein also actively transports its own transcripts to protrusions.

Finally, we demonstrated that KIF1C is needed to transport these APC-dependent mRNAs to protrusions along microtubules and to form their foci. Indeed, we did not observe these foci in cell extensions when the KIF1C protein is absent. These results suggest a dual function of KIF1C as microtubule motor and ”mRNA anchoring module promoting clustering” [Pichon *et al.*(2021)].

6.4.2 Quantification of peripheral mRNAs

This study was an opportunity to exploit a proven FISH-quant v2 and to scale a quantitative analysis that characterizes protrusion RNAs from HeLa cells. I identified 27,644 individual cells, detecting mRNAs from 40 different genes, including KIF1C, APC-dependent mRNAs and control transcripts. For every cell, I performed an automated spot detection, nucleus and cell segmentation, from 2D projected images. I compute RDI Calculator features from [Stueland *et al.*(2019)], especially the peripheral distribution index that measures how close the RNAs localize to the cell periphery (see

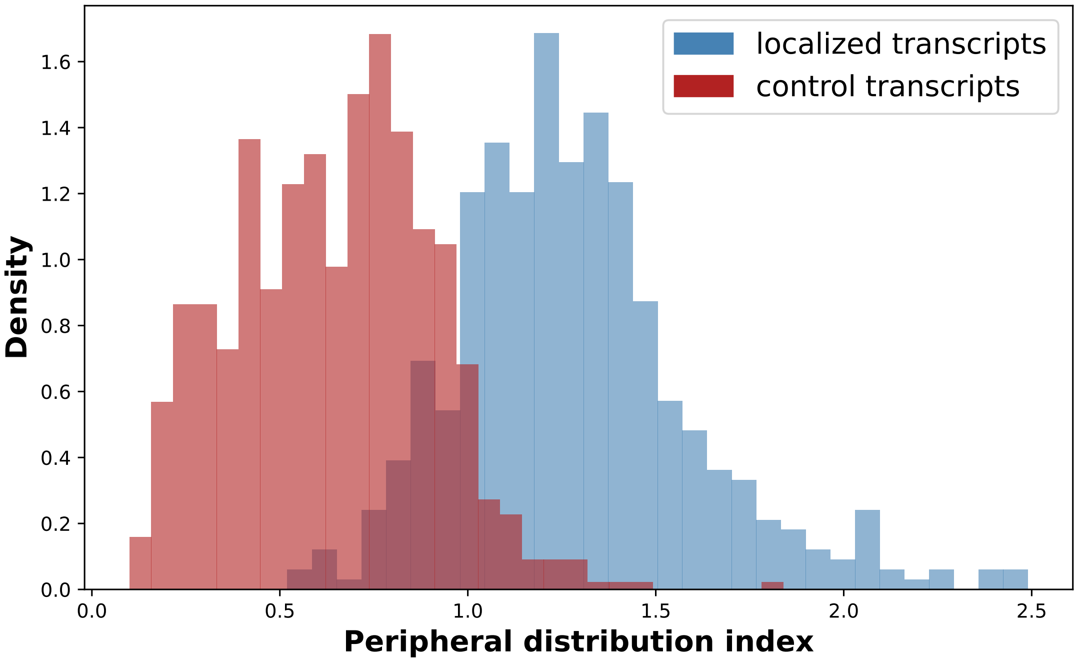


Figure 6.10: Histogram of the peripheral distribution index for two group of transcripts. Localized transcripts include: KIF1C, TRAK2 and NET1. Control transcripts include NEK9, NIN, NPM1, OBSL1, OLA1, PAX2, PKHD1, RGS14, SMAD7 and SPAST

Section 5.2.2). The greater this index, the more concentrated in the protrusion are the transcripts.

In Figure 6.10, I compared two groups of transcripts: KIF1C, TRAK2 and NET1, three mRNAs that bind to KIF1C protein for active transport to the protrusions, and a control group unrelated to KIF1C. For the peripheral distribution index, a completely random distribution has a value of 1. Here, as expected, the protrusion mRNAs exhibited a biased spatial distribution toward the cell extensions.

6.5 Conclusion

In this chapter, I present several applications of FISH-quant v2. Based on several high content screens in HeLa cell, I performed large scale quantitative analyses to answer question centered around RNA localization and local translation. In total, several dozens of transcripts were analyzed across 100,000 individual cells and hundreds of fluorescent bioimages.

In [Chouaib *et al.*(2020)], I provided quantifications highlighting the high level of heterogeneity in mRNAs localization, both on the population as well as the single-cell level. The developed classification pipeline enabled the identification of complex RNA localization patterns: intranuclear, nuclear edge, perinuclear, foci and protrusion. We showed that the foci pattern is caused by two distinct biological processes: first, RNA storage in P-bodies and second, translation factories, where RNAs are efficiently translated By extrapolating to the 20,000 human genes, we could expect to find a few hundred of translation factory structures in a cell. Remarkably, such phenomenon of locally targeted translation is observed for other localization patterns. For some cases, this localization even seems to be translation dependent and tends to disappear if the nascent protein synthesis is inhibited.

In [Safieddine *et al.*(2021)], a second set of localized transcripts was analyzed with spatial and temporal dynamics. I characterized mRNAs with a centrosomal localization. This pattern presents a temporal dynamic in addition to be translation dependent. Cell cycle regulates the choreography and the localization of different mRNAs around the centrosomes.

In [Pichon *et al.*(2021)], we confirmed the importance of the KIF1C motor protein in the transport of some APC-dependent transcripts to cell extensions. Surprisingly, this proteins also bind its own mRNAs along microtubules which suggests the existence of a positive feedback loop to locally enrich cell protrusion with the needed mRNAs.

Taken together, our data suggest that even in a simple system such as HeLa cells, translation might be compartmentalized to a much higher degree and with a finer granularity than anticipated. Our analysis reveals spatial and temporal dynamics as well as complex mechanisms to regulate the metabolism of nascent proteins. Robust quantitative and numerical methods are needed more than ever to address these complexity and volume of interactions.

Part IV

Conclusion

7

Conclusion and perspectives

Contents

7.1 Results of the thesis 122
7.2 Perspectives on the future of smFISH analysis 123
7.3 Publications 125

The objective of my PhD thesis was to design methods and software for the large-scale analysis and quantification of RNA localization patterns from smFISH images and to apply these methods to large-scale screens. This provides new opportunities to explore more advanced algorithms, incorporate new input modalities, or answer more challenging questions.

7.1 Results of the thesis

In **Chapter 2**, I present FISH-quant v2, the general framework implemented and released during my PhD [Imbert *et al.*(2022b)]. It is a user-friendly, open-source and Python-based tool that builds on and extends the initial MATLAB version of FISH-quant providing modules for the most common operations needed for a smFISH analysis. FISH-quant v2 features a fully automatic spot detection and pre-trained deep learning models for cell and nuclei segmentation and a set of hand-crafted features designed to quantitatively describe the spatial distribution of RNA inside cells. The software architecture is highly modular and flexible, allowing the design of customized workflows. FISH-quant v2 includes *bigfish*, a general Python package for the analysis, a GUI, the module *simfish*, a Python package for simulations, as well as documentation and interactive examples. In addition, the protocol and the analysis pipeline we have performed for different applications are described in a recent publication [Safieddine *et al.*(2022)].

In **Chapter 3**, I present detection methods integrated in FISH-quant v2. An important improvement concerns the spot detection itself with the implementation of a heuristic to threshold the detected spots without any manual intervention. This enables a large scale application of our methods. Furthermore, I describe solutions to decompose regions with cluttered spots, detect clusters or analyze spot colocalization. Finally, I explain how spot images are simulated in FISH-quant v2. Simulation plays a key role for the performance assessment of spot detection under different noise conditions.

In **Chapter 4**, I present methods integrated in FISH-quant v2 to segment nuclei and cells. It includes in-house deep learning models trained on various fluorescence microscopy images. Most importantly, the ability to use external frameworks and models is facilitated with postprocessing methods to format and refine segmentation masks. In addition, I describe contributions to other projects aiming at overcoming the need for massive image annotation for image segmentation, for instance by the use of in silico labeling for pre-training [Bonte *et al.*(2022)].

In **Chapter 5**, I present my different contributions to analyze RNA distribution. First, previous detection and segmentation results are summarized in a homogeneous coordinate representation of each identified cell. Second, I present a set of features describing the spatial distribution of spots, which I implemented in FISH-quant v2, based on the extracted coordinates. This feature set is building on and extending features that have been previously used in our research group. Third, I present alternative approaches to classify RNA localization patterns, either with CNN [Dubois *et al.*(2019)] or point cloud models [Imbert *et al.*(2022a)]. These alternatives also leverage a simulation framework that generates localization patterns at will and available in FISH-quant v2.

In **Chapter 6**, I present three (published) applications of my quantitative methods.

In the first study [Chouaib *et al.*(2020)], I design a classification pipeline to discriminate several generic RNA localization patterns: intranuclear, nuclear edge, perinuclear, foci and protrusion. I also detail additional quantifications performed on the impact of translation inhibitors for a newly characterized pattern, the *translation factories*. In the second study [Safieddine *et al.*(2021)], I focus on the centrosomal pattern and adapt my analysis pipeline to detect centrosomes at scale. This enables a statistical description of several genes of interest. In the third study [Pichon *et al.*(2021)], I analyze the protrusion pattern and implement dedicated features recently described in the literature. All in all, these studies include several dozens of transcripts, analyzed through 100,000 individual cells. They reveal regulation mechanisms more complex than expected with various spatial and temporal dynamics.

7.2 Perspectives on the future of smFISH analysis

Since the first design of smFISH in the late 1990s, fluorescent microscopy techniques have dramatically improved. Fluorophores are cheaper or brighter, or both. The hybridization protocols are standardized and thus more reproducible. The fluorescent images have a higher SNR and this trend is likely to continue. Therefore, in future experimental studies, the detection and segmentation steps could be facilitated.

Improving localization features In chapter 5, I have presented PointFISH, a neural network architecture that can directly process point clouds. This network was trained on simulated data. There is a number of possible improvements: first, we could improve the simulations themselves, for instance by investigating a larger diversity of patterns. An in-depth study of the relation between pattern strength and generalization capacity of the trained network would also be an interesting perspective. Furthermore, training on simulated data always represents the problem that the data distribution we have trained on does usually not match the distribution of real data. For this reason, the use of domain adaptation might be a promising strategy to investigate. Finally, the use of self-supervised learning is also an interesting alternative to training on simulations.

Reference markers The analysis of RNA localization requires the segmentation of the cell boundary and ideally of structures inside the cell. The insights that can be gained from smFISH studies very much depend on the choice of the marked structures, as we have seen in chapter 6. An important extension is therefore the use of in silico labeling techniques. This would allow for instance to segment certain structures without the use of fluorescent labels, which in turn frees these channels to mark other structures. Such a strategy thus has the potential to enrich the description of the spatial RNA distribution by adding more reference structures.

Analysis of localization heterogeneity Another question that has not yet been addressed in detail is the explanation of localization heterogeneity. Indeed, we have seen in chapter 6, that for a given RNA not all cells show the same localization pattern, and that only a fraction of RNA molecules has preferential localization. It is not well understood where these differences come from. One possible explanation might be the cell cycle phase. It will therefore be interesting to assess the cell cycle phase, ideally by in silico labeling from phase contrast images or dedicated fluorescent markers, in order to stratify cells and to investigate which of the patterns are cell cycle dependent. On a larger perspective, other cell properties could be correlated to the observed patterns. For this reason, one important aspect to be investigated in the future will be to relate the localization information to other cellular properties.

Analyzing transcriptomic profiles The most important trend today is the advent of experimental techniques such as MERFISH and seqFISH, that allow the imaging of tens or hundreds of RNAs in the same cells. From a computational point of view, the algorithmic steps (cell and nucleus segmentation, spot detection, etc.) still remain the same, but they have to be complemented by the joint analysis of different RNA channels. In most studies, these data are used to map each cell to a cell type or cell state, that is defined by the transcriptomic profile. If used in cell culture, this means that each cell is then equipped with an expression vector of multiple RNAs, the resulting cell type or state, the localization features for each of these RNAs and the morphological properties of the cell. The biggest challenge in these data is to deal with these high dimensional input data, to identify relationships between the input variables and to synthesize this extremely rich information to representations that can be interpreted biologically. FISH-quant v2 offers solutions to some of these tasks, by providing tools for the basic image analysis and the analysis of localization features, but it is clear that we will need important extensions to tackle these additional challenges.

From cell culture to tissue While there are studies employing these transcriptomic techniques in cell culture, the most widely used application scenario concerns the study of tissue architecture. While in principle, the overall algorithmic steps remain similar, segmentation can be a bottleneck, as it is much more difficult in tissue than in cell culture. Moreover, the degree of difficulty is very variable depending on the tissue type. Also spot detection can be complicated by the presence of auto-fluorescence, possibly requiring more refinement steps. Finally, the main task of such data is also different, because the RNA channels are mostly used for cell type calling. An important challenge is to describe the spatial distribution of cells in the tissue. For this, there might be some methodological overlap with the analysis of subcellular localization patterns, but there are important differences too. Most importantly, spatial transcriptomics at cellular resolution is an expensive technique, and purely data-driven approaches might therefore be limited by the low number of data points.

Data repositories and challenges On a more general perspective, bioimage analysis will continue to be driven by the generation of massive annotated datasets, and challenges. In the past, such challenges played an important role for the design and benchmarking of segmentation networks, such as Stardist, NucleAIzer and Cellpose. For protein localization patterns, the availability of large scale annotated data has also been instrumental. For these reasons, it would be very interesting to organize a challenge in the field of RNA detection and classification of localization patterns. Our simulation framework could play an important role for this: computational challenges often contain one sub-task related to simulated image data.

Altogether, new experimental methods, accompanied by computational methods, will enable a more efficient exploration of the transcriptome, a better understanding of the mechanism and function of RNA localization and the spatial regulation of gene expression. Obviously, there is still a lot of work to achieve for future researchers and I hope that my contribution with this thesis will be useful to the community.

7.3 Publications

First author or co-first author

- Racha Chouaib[1], Adham Safieddine[1], Xavier Pichon[1], **Arthur Imbert**[1], Oh Sung Kwon, Aubin Samacoits, Abdel-Meneem Traboulsi, Marie-Cécile Robert, Nikolay Tsanov, Emeline Coleno, Ina Poser, Christophe Zimmer, Anthony Hyman, Hervé Le Hir, Kazem Zibara, Marion Peter, Florian Mueller, Thomas Walter, Edouard Bertrand (2020),*A dual protein-mRNA localization screen reveals compartmentalized translation and widespread co-translational RNA targeting*, Developmental Cell 54 (6), 773.

- **Arthur Imbert**, Wei Ouyang, Adham Safieddine, Emeline Coleno, Christophe Zimmer, Edouard Bertrand, Thomas Walter, Florian Mueller (2022), *FISH-quant v2: a scalable and modular tool for smFISH image analysis*, RNA, pp. 786–795, iSSN: 1355–8382, 1469–9001.

- **Arthur Imbert**, Florian Mueller, Thomas Walter (2022), *PointFISH: learning point cloud representations for RNA localization patterns*, in 2022 European Conference on Computer Vision (ECCV 2022) Workshop on BioImage Computing *(to be published).*

Co-author

- Rémy Dubois, **Arthur Imbert**, Aubin Samacoïts, Marion Peter, Edouard Bertrand, Florian Mueller, Thomas Walter (2019), *A Deep Learning Approach To Identify mRNA Localization Patterns*, in 2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019), pp. 1386-1390, iSSN: 1945-8452.

- Adham Safieddine, Emeline Coleno, Soha Salloum[1], **Arthur Imbert**[1], Abdel-Meneem Traboulsi, Oh Sung Kwon, Frederic Lionneton, Virginie Georget, Marie-Cécile Robert, Thierry Gostan, Charles-Henri Lecellier, Racha Chouaib, Xavier Pichon, Hervé Le Hir, Kazem Zibara, Florian Mueller, Thomas Walter, Marion Peter, Edouard Bertrand (2021), *A choreography of centrosomal mRNAs reveals a conserved localization mechanism involving active polysome transport*, Nature Communications 12 (1), 1352.

- Xavier Pichon[1], Konstadinos Moissoglu[1], Emeline Coleno, Tianhong Wang, **Arthur Imbert**, Marie-Cecile Robert, Marion Peter, Racha Chouaib, Thomas Walter, Florian Mueller, Kazem Zibara, Edouard Bertrand, Stavroula Mili (2021), *The*

---

[1] Equal contribution

kinesin KIF1C transports APC-dependent mRNAs to cell protrusions, RNA 27 (12), 1528.

- Adham Safieddine, Emeline Coleno, Frederic Lionneton, Abdel-Meneem Traboulsi, Soha Salloum, Charles-Henri Lecellier, Thierry Gostan, Virginie Georget, Cédric Hassen-Khodja, **Arthur Imbert**, Florian Mueller, Thomas Walter, Marion Peter, Edouard Bertrand (2022), *HT-smFISH: a cost-effective and flexible workflow for high-throughput single-molecule RNA imaging*, Nature Protocol.

- Thomas Bonte, Maxence Philbert, Emeline Coleno, Edouard Bertrand, **Arthur Imbert**, Thomas Walter (2022), *Learning with minimal effort: leveraging in silico labeling for cell and nucleus segmentation*, in 2022 European Conference on Computer Vision (ECCV 2022) Workshop on BioImage Computing *(to be published).*

Part V

Appendices

A

Simulations

A.1 Spot and cluster simulations

Figure A.1 is a sample of simulated images of spots. We modulate the number of spots and the intensity of the noise for each image.

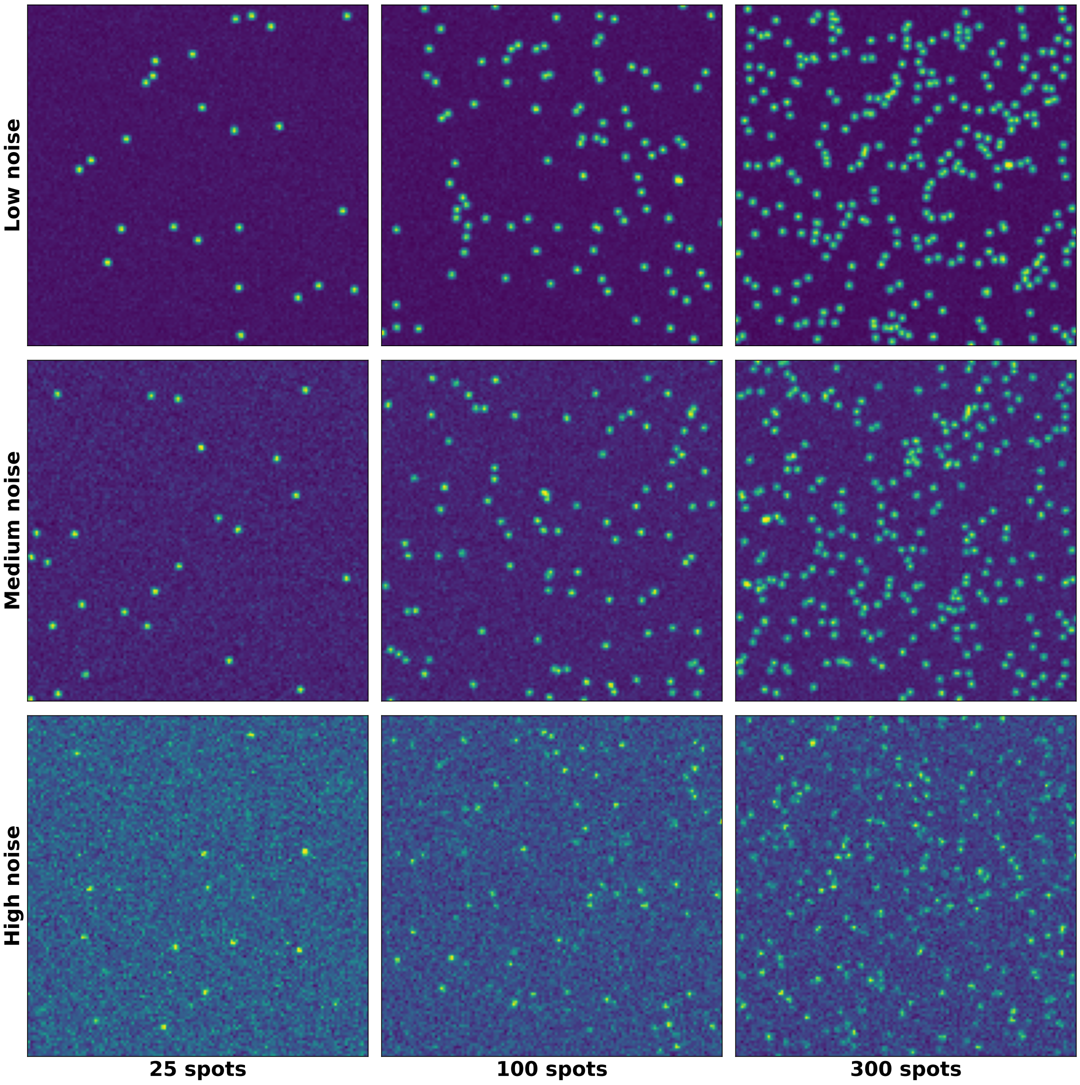


Figure A.1: Simulations of spots under different noise regimes

Figure A.2 use the same logic but with a unique cluster of spots simulated in the center. This time, we modulate the number of spots inside the cluster.

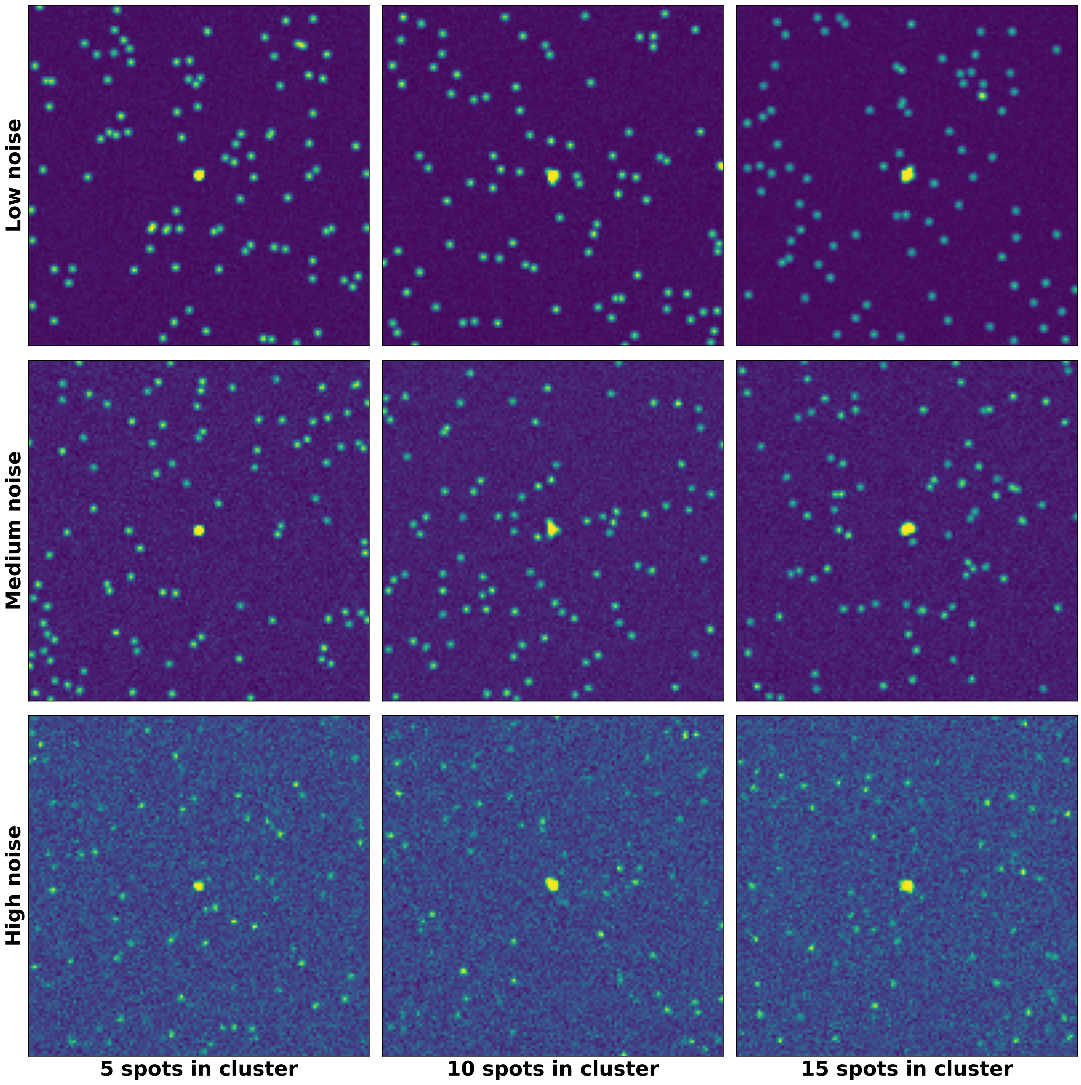


Figure A.2: Simulations of cluster under different noise regimes

A.2 Localization pattern simulations

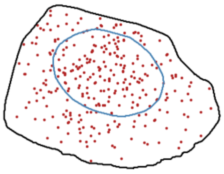


Figure A.3: Simulations of random spots

Figure A.3 is an example of random pattern with 300 simulated spots. On the contrary, in figures A.4 and A.5 we can observe localized patterns, with 300 simulated spots as well, but different pattern strengths (10%, 50% and 90% of localized spots).

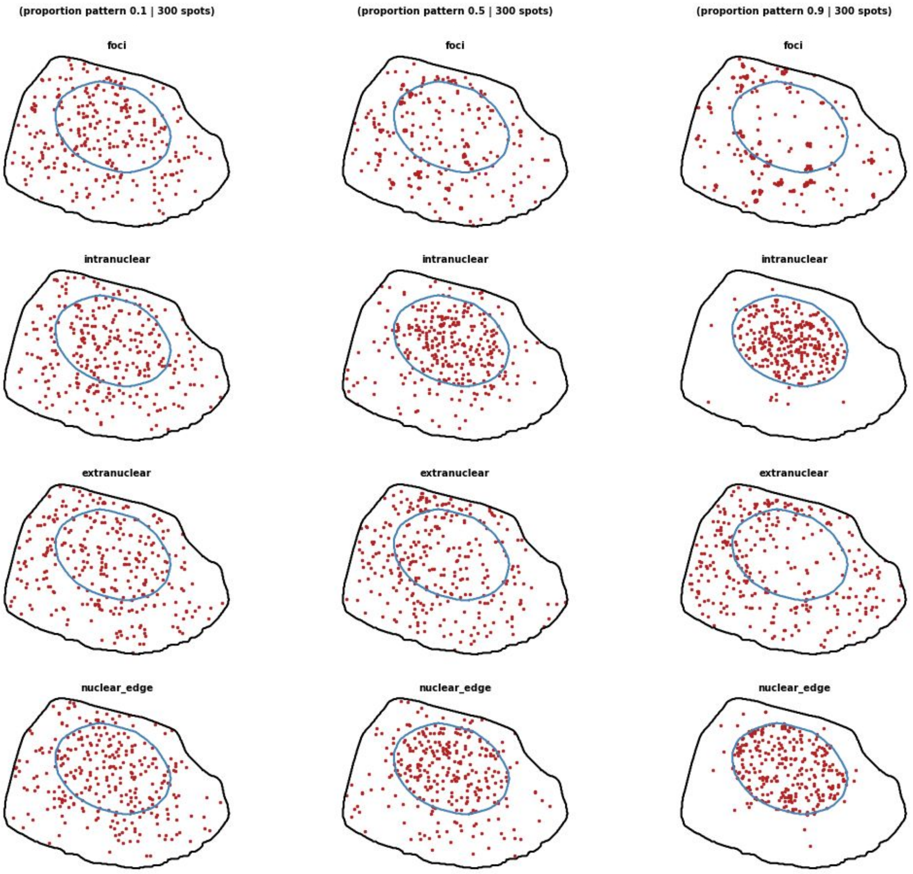

(proportion pattern 0.1 | 300 spots)
(proportion pattern 0.5 | 300 spots)
(proportion pattern 0.9 | 300 spots)
foci
foci
foci
intranuclear
intranuclear
intranuclear
extranuclear
extranuclear
extranuclear
nuclear_edge
nuclear_edge
nuclear_edge

Figure A.4: Simulations of localization patterns (first part)

Because we visualize plots in 2D, some of these localization patterns (simulated in 3D) can be misleading. In particular, visualized in 2D, cell edge pattern can appear similar to random pattern, and nuclear edge pattern looks like intranuclear one. Same difficulties can arise when one manually annotate real cell images.

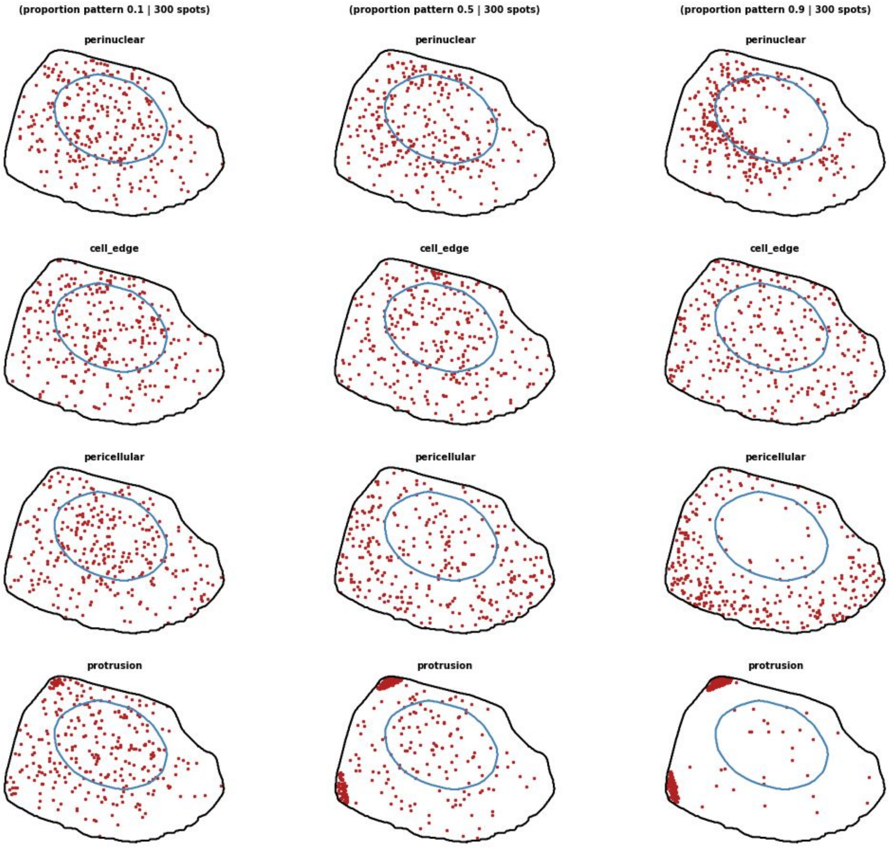


Figure A.5: Simulations of localization patterns (second part)

B

Detection and noise

B.1 Detection with different noise intensities

In figure B.1 we present elbow curves for three levels of noise. We observe the number of detected spots as a function of intensity thresholds. The optimal threshold (in red) is selected based on these curves. When an image has a high SNR, the difference of regime between the noisy background blobs and the actual spots is clearly distinct in the elbow curve.

The result in term of detection can be observed in figure B.2. For each image, 100 spots are simulated. Detected positions are in red and simulated ground truth positions in white.

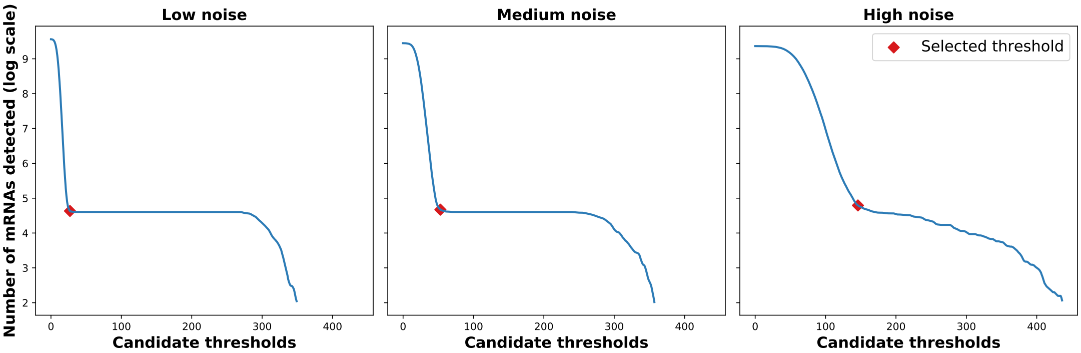


Figure B.1: Elbow curves with different noise levels

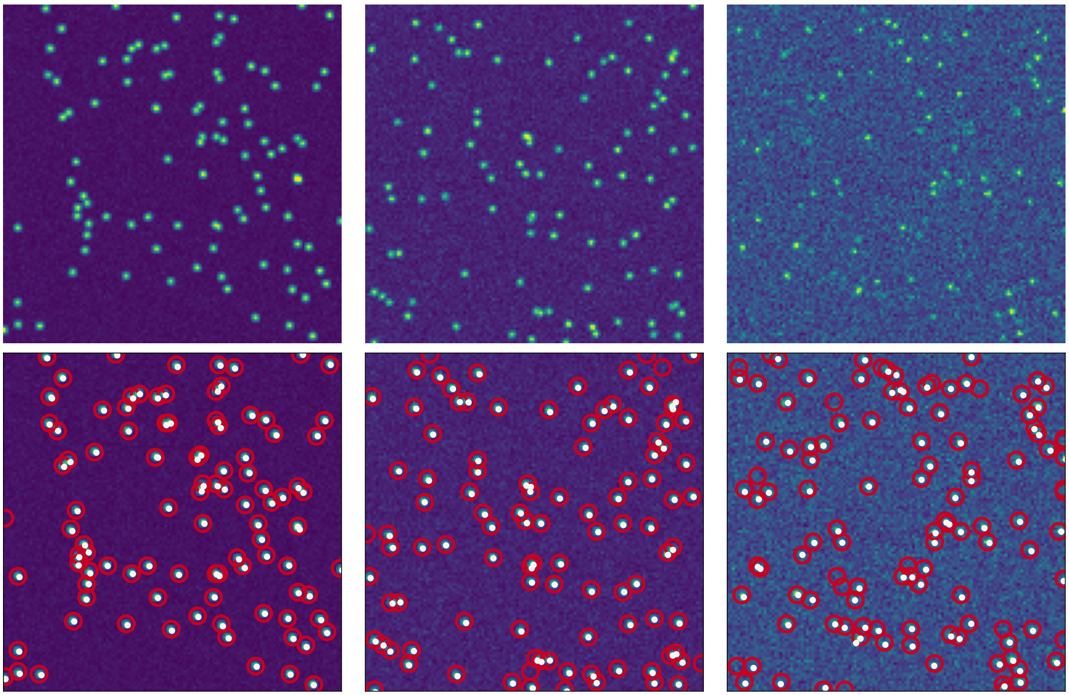

Low noise **Medium noise** **High noise**

Figure B.2: Detection of 100 spots with different noise levels

C

Convolutional features

A paper [Dubois *et al.*(2019)] published at my arrival in the CBIO team proposes to tackle the RNA localization problem by learning intermediate features through a convolutional network:

R. Dubois, A. Imbert, et al. (2019), *A Deep Learning Approach To Identify mRNA Localization Patterns*, in 2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019), pp. 1386-1390, iSSN: 1945-8452.

I have participated to this paper by running some experiments to affine the final results, but also by exploring some variants. Here we present the improvements we obtained with convolutional features and the last results we reached after the paper publication.

C.1 Localization features learned with convolutional neural network

A deep learning framework presents the advantage to simplify our feature engineering pipeline. To this end, we directly train a convolutional neural network to classify different localization patterns from a 3-channel image. Instead of an usual RGB image, we build an input image from the coordinate representation of the cell. More specifically, we use the RNA point cloud (projected in 2D), the 2D cell boundary and the 2D nucleus boundary. The original paper stacks three layers: one image that assigns to each pixel the number of RNAs detected in the pixel, one image with the nucleus boundary and one with the cell boundary. We also have tested different input designs, replacing the last two boundary images by the binary masks (see figure C.1) or the distance map. Eventually, best results were obtained with binary masks.

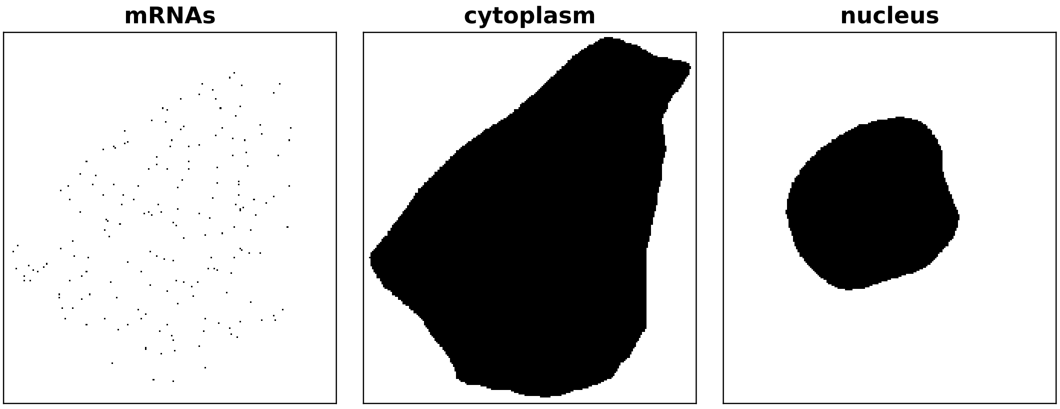


Figure C.1: Input images for the convolutional model

Usually, a neural network requires a large annotated dataset to reach good performance and generalized well. We exploit the MATLAB simulation framework previously released by our group [Samacoits *et al.*(2018)]. These simulations differ from *simfish*, even if the original 318 cell templates from which we simulate are the same. In particular, localization patterns and rules to modulate pattern strength are different. They are greatly simplified in *simfish*. We simulated 200,000 cells to train, validate and test our model (with a stratified split of 60%, 20% and 20% respectively). We also use a real dataset with 10 different transcripts visualized across 2791 cells. Unlike the original paper, we only simulate 5 different patterns: polarization patterns and cell edge were removed because too rare in the real dataset.

We use SqueezeNet [Iandola *et al.*(2016)] for the image classification task. This model integrates compression techniques to keep a reasonable accuracy, but drastically reduces the number of parameters to train. These techniques include squeeze operations to decrease the feature dimension of the layers or a more efficient mix of different convolution kernel sizes.

C.2 Results

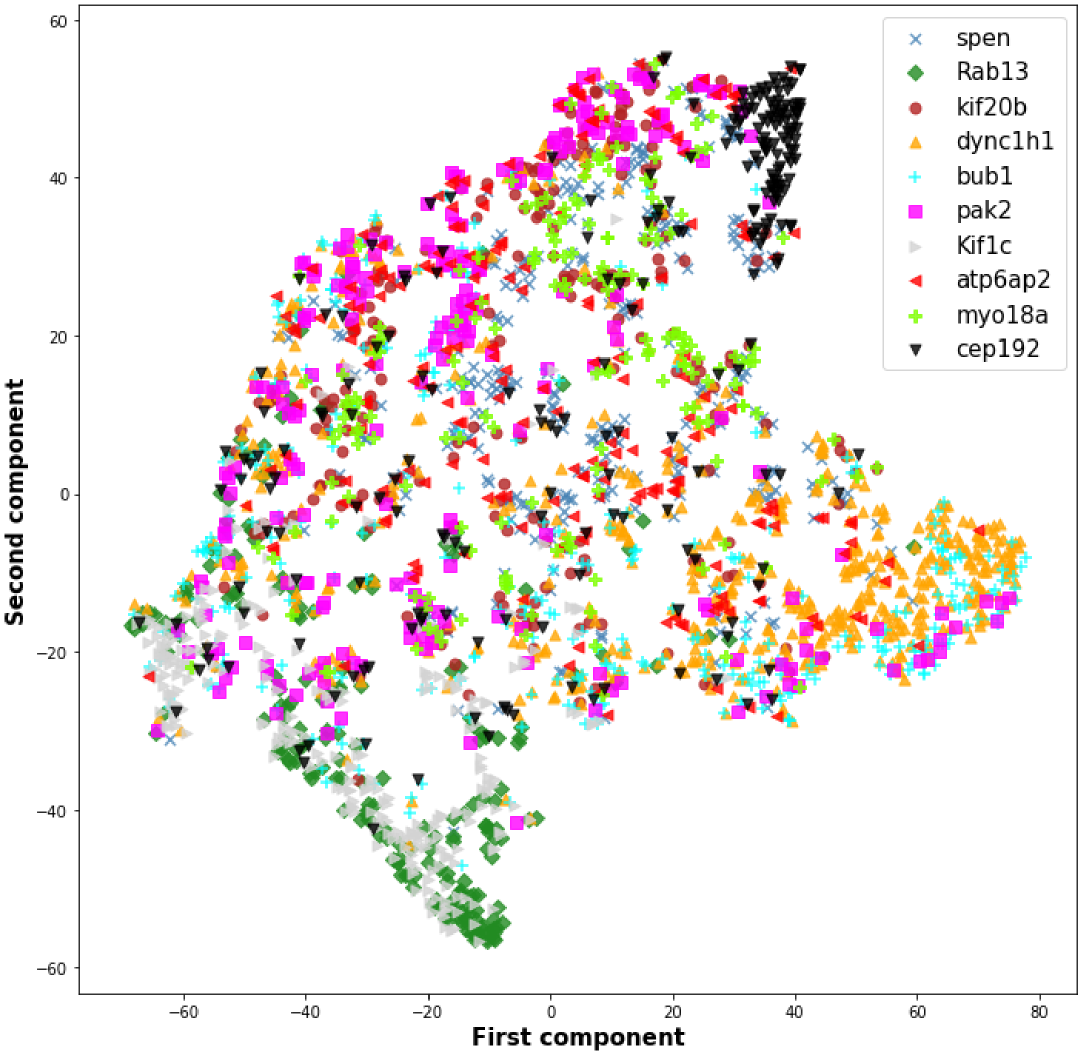
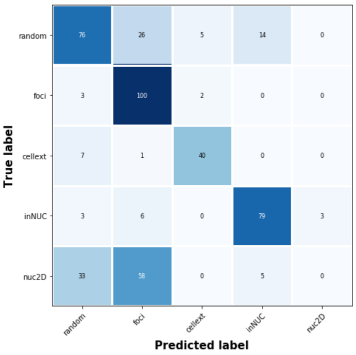


Figure C.2: t-SNE projection of the learned features

We can evaluate the classification predictions for the simulated test dataset. We observe on the confusion matrix C.3 that the model performs well with the foci pattern, the protrusion pattern (*cellext*) and the intranuclear pattern (*inNUC*). On the opposite, we fail to predict nuclear edge pattern (*nuc2D*) and random patterns are often confused with foci or intranuclear. Globally, results are not as good as expected and do not generalize to all the frequent localization patterns observed in the transcriptome.

With the real dataset, we do not have annotations for each cell, but we can still observe more frequent patterns for some genes. Indeed, the 10 genes selected present a diversity of localization patterns. We collect the last network layer (before the classification head) and vizualize our cell population with a 2D t-SNE embedding [van der Maaten et Hinton(2008), Wattenberg *et al.*(2016)]. A first striking observation in figure C.2 is the high level of heterogeneity among the cells (even among the ones labelled with the same FISH). However, distinct clusters can be observed, relevant with the frequent pattern observed with the genes like the intranuclear pattern frequently observed with the CEP192, or the protrusion pattern frequent with KIF1C and RAB13.



	random	foci	cellext	inNUC	nuc2D
random	76	26	5	14	0
foci	3	100	2	0	0
cellext	7	1	40	0	0
inNUC	3	6	0	79	3
nuc2D	33	58	0	5	0

Figure C.3: Confusion matrix

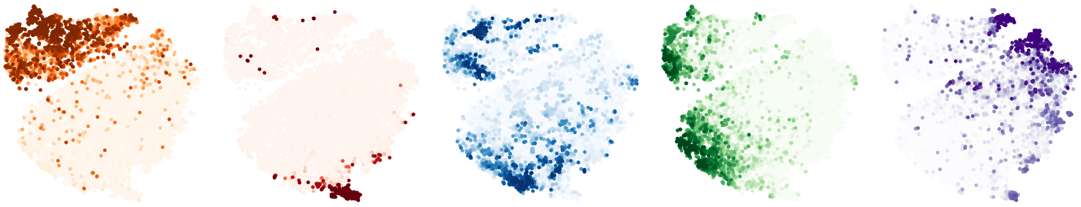
This study reveals two potential difficulties when we train neural network to learn intermediate features, instead of designing manual ones. First, when the model makes a prediction, it is much more complicated to understand why it favours a pattern over another. The internal representation built by the network are not easily accessible or easy to interpret. There is no direct method to visualize these representations, to quantify the importance of a subregion from the input image in the prediction or to measure the confidence given to a prediction. However, this is an active field of research and papers [Gal et Ghahramani(2016), Olah *et al.*(2017), Olah *et al.*(2018)] are investigating it. Second, by exploiting images of coordinates, we lose a lot of information. We project a potential 3D RNA point cloud in 2D, use an inefficient dense representation (an image) of a sparse input (the RNA point cloud) and convolutions might not be the right choice to deals with coordinates inputs [Liu *et al.*(2018)].

D

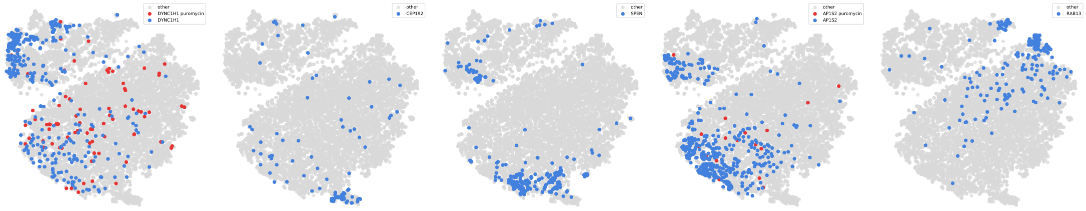
Pattern classification

D.1 Supervised and unsupervised analyses

To investigate the agreement between supervised and unsupervised analysis I try to interpret the different regions of the t-SNE plot from [Chouaib *et al.*(2020)]. With the trained classifiers, I compute the probability for each cell to be in a particular pattern and plot these probabilities on top of the t-SNE. In Figure D.1 we can observe that cells with high probability of a given pattern concentrated in the same area of the t-SNE plot. Therefore, the embedding appears consistent and a direct interpretation of the point cloud regions seems possible. In addition, we can notice that the assignments of the classifiers are consistent with the manual annotations illustrated in Figure 6.3.



(a) Foci (c) Intranuclear (e) Nuclear edge (g) Perinuclear (i) Protrusion



(b) DYNC1H1 (d) CEP192 (f) SPEN (h) AP1S2 (j) RAB13

Figure D.1: (*Top*) Visualization of random forest classification probabilities in the t-SNE, from [Chouaib *et al.*(2020)]. Color indicates the probabilities of the cell to be classified in the indicated localization pattern. The darker, the higher the probability is. (*Bottom*) Visualization of cells for some specific genes with the indicated localization pattern above. Cells can be untreated (*blue*) or treated with puromycin (*red*)

On the bottom row we can observed how the cells for 5 different genes localize in the t-SNE plot. From the results presented in Figure 6.4, we also identify dominant localization patterns for these transcripts:

- DYNC1H1 with a foci and perinuclear patterns.
- CEP192 with an intranuclear pattern.
- SPEN with a nuclear edge pattern.
- AP1S2 with a perinuclear and a small foci patterns.
- RAB13 with a protrusion and a small foci patterns.

These localizations observed at the gene level are consistent with the manual annotations in Figure 6.3 and t-SNE regions interpreted from the top row of Figure D.1.

A last observation that can be made from Figure D.1 is the apparent diversity of patterns observed at the gene level. For example, cells where we spot DYNC1H1 transcripts are polarized toward the nucleus-related regions of the point cloud. Moreover, a majority of untreated cells are in the foci-related region, while those treated with puromycin present a more uniform distribution across the t-SNE. Such results are consistent with the observed impact of the puromycin for genes like DYNC1H1 presenting a transcription factory pattern.

D.2 Cell-wise pattern heterogeneity

Another way to visualize the heterogeneity in terms of localization patterns is to plot the predicted probabilities for every cell in a heat map. In Figure D.2, I adapt the results aggregated by genes from Figure 6.4. It clearly appears that, for a given gene, the observed localization patterns are noisy. Even if SPEN transcripts localize most of the time along the nucleus membrane, they also can be observed forming a cluster of RNAs. For some cells we do not even observe a specific pattern. Such heterogeneity justifies the use of a quantitative pipeline to produce statistical results over a large population of cells.

These heat maps also confirm that different localization patterns can coexist for the same transcript but not necessarily within the same cells at the same time. For example with ASPM transcripts, cells are identified with a foci pattern, a nuclear edge pattern or both at the same time.

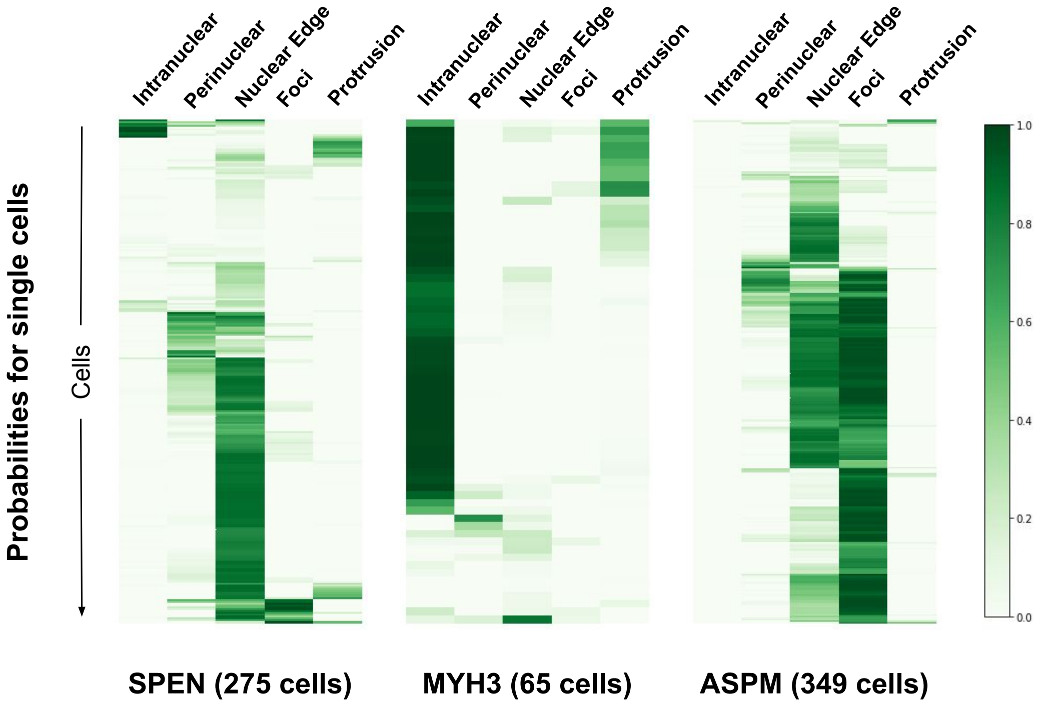


Figure D.2: Three heat maps from [Chouaib *et al.*(2020)] with the probability of single cells to have each patterns. Only cells where we visualized SPEN, MYH3 and ASPM transcripts are represented. Each row corresponds to a cell and the color indicates the probabilities of the cell to be classified in the indicated localization pattern

Bibliography

[mah(2013)] *Mahotas: Open source software for scriptable computer vision*Journal of Open Research Software **1** (1), e3 (2013).

[Abadi *et al.*(2015)] M. Abadi, A. Agarwal, et al. (2015), *TensorFlow: Large-scale machine learning on heterogeneous systems*.

[Ahonen *et al.*(2006)] T. Ahonen, A. Hadid, et M. Pietikainen, *Face description with local binary patterns: Application to face recognition*, IEEE Transactions on Pattern Analysis and Machine Intelligence **28** (12), 2037 (2006).

[Allalou et Wählby(2009)] A. Allalou et C. Wählby, *Blobfinder, a tool for fluorescence microscopy image cytometry*, Computer Methods and Programs in Biomedicine **94** (1), 58 (2009).

[Ba *et al.*(2016)] J. L. Ba, J. R. Kiros, et G. E. Hinton (2016), *Layer normalization*.

[Bahry *et al.*(2021)] E. Bahry, L. Breimann, et al., *RS-FISH: Precise, interactive, fast, and scalable FISH spot detection*, bioRxiv 10.1101/2021.03.09.434205 (2021).

[Bai et Urtasun(2017)] M. Bai et R. Urtasun (2017), *Deep watershed transform for instance segmentation*, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Battich *et al.*(2013)] N. Battich, T. Stoeger, et L. Pelkmans, *Image-based transcriptomics in thousands of single human cells at single-molecule resolution*, Nature Methods **10** (11), 1127 (2013).

[Battich *et al.*(2015)] N. Battich, T. Stoeger, et L. Pelkmans, *Control of Transcript Variability in Single Mammalian Cells*, Cell **163** (7), 1596 (2015).

[Bay *et al.*(2006)] H. Bay, T. Tuytelaars, et L. Van Gool (2006), *Surf: Speeded up robust features*, in *Computer Vision – ECCV 2006* (Springer Berlin Heidelberg, Berlin, Heidelberg) pp. 404–417.

[Berg *et al.*(2019)] S. Berg, D. Kutra, et al., *ilastik: interactive machine learning for (bio)image analysis*, Nature Methods **16** (12), 1226 (2019).

[Beucher et Lantuéjoul(1979)] S. Beucher et C. Lantuéjoul (1979), *Use of watershed in contour detection*, in *International Workshop on image processing: real-time Edge and Motion detection/estimation*.

[Bishop(2006)] C. M. Bishop (2006), *Pattern Recognition and Machine Learning (Information Science and Statistics)* (Springer-Verlag, Berlin, Heidelberg).

[Blower(2013)] M. D. Blower (2013) (Academic Press) pp. 1–39.

[Boland *et al.*(1998)] M. V. Boland, M. K. Markey, et R. F. Murphy, *Automated recognition of patterns characteristic of subcellular structures in fluorescence microscopy images*, Cytometry **33** (3), 366 (1998).

[Bonte *et al.*(2022)] T. Bonte, M. Philbert, et al. (2022), *Learning with minimal effort: leveraging in silico labeling for segmentation*, in *2022 European Conference on Computer Vision (ECCV 2022) Workshop on BioImage Computing.*

[Bouilhol *et al.*(2022)] E. Bouilhol, A. F. Savulescu, et al., *DeepSpot: a deep neural network for RNA spot enhancement in smFISH microscopy images*, Biological Imaging , 1 (2022).

[Bovaird *et al.*(2018)] S. Bovaird, D. Patel, et al., *Biological functions, regulatory mechanisms, and disease relevance of rna localization pathways*, FEBS Letters **592** (17), 2948 (2018).

[Bradley *et al.*(2020)] L. Bradley, B. Sipőcz, et al. (2020), *astropy/photutils: 1.0.0*.

[Breiman(2001)] L. Breiman, *Random Forests*, Machine Learning **45** (1), 5 (2001).

[Bustin(2000)] S. A. Bustin, *Absolute quantification of mRNA using real-time reverse transcription polymerase chain reaction assays*, Journal of Molecular Endocrinology **25** (2), 169 (2000).

[Carpenter *et al.*(2006)] A. E. Carpenter, T. R. Jones, et al., *CellProfiler: image analysis software for identifying and quantifying cell phenotypes.*, Genome biology **7** (10), R100 (2006).

[Chang et Lin(2011)] C.-C. Chang et C.-J. Lin, *Libsvm: A library for support vector machines*, ACM transactions on intelligent systems and technology (TIST) **2** (3), 1 (2011).

[de Chaumont *et al.*(2012)] F. de Chaumont, S. Dallongeville, et al., *Icy: an open bioimage informatics platform for extended reproducible research*, Nature Methods **9** (7), 690 (2012).

[Chen *et al.*(2015)] K. H. Chen, A. N. Boettiger, et al., *Spatially resolved, highly multiplexed rna profiling in single cells*, Science **348** (6233), aaa6090 (2015).

[Chin et Lécuyer(2017)] A. Chin et E. Lécuyer, *Rna localization: Making its way to the center stage*, Biochimica et Biophysica Acta (BBA) - General Subjects **1861** (11, Part B), 2956 (2017).

[Chin et Lécuyer(2020)] A. Chin et E. Lécuyer, *Translating messages in different neighborhoods*, Developmental Cell **54** (6), 691 (2020).

[Chouaib *et al.*(2020)] R. Chouaib, A. Safieddine, et al., *A dual protein-mrna localization screen reveals compartmentalized translation and widespread co-translational rna targeting*, Developmental Cell **54** (6), 773 (2020).

[Christiansen *et al.*(2018)] E. M. Christiansen, S. J. Yang, et al., *In Silico Labeling: Predicting Fluorescent Labels in Unlabeled Images*, Cell **173** (3), 792 (2018).

[Cochard *et al.*(2022)] A. Cochard, M. G.-J. Navarro, et al., *RNA at the surface of phase-separated condensates impacts their size and number*, Biophysical Journal **121** (9), 1675 (2022).

[Cornes *et al.*(2022)] E. Cornes, L. Bourdon, et al., *pirnas initiate transcriptional silencing of spermatogenic genes during c.ăelegans germline development*, Developmental Cell **57** (2), 180 (2022).

[Crouse(2016)] D. F. Crouse, *On implementing 2d rectangular assignment algorithms*, IEEE Transactions on Aerospace and Electronic Systems **52** (4), 1679 (2016).

[Cutler *et al.*(2022)] K. J. Cutler, C. Stringer, et al., *Omnipose: a high-precision morphology-independent solution for bacterial cell segmentation*, Nature Methods , 1 (2022).

[Das *et al.*(2021)] S. Das, M. Vera, et al., *Intracellular mRNA transport and localized translation*, Nature Reviews Molecular Cell Biology **22** (7), 505 (2021).

[Deng *et al.*(2009)] J. Deng, W. Dong, et al. (2009), *Imagenet: A large-scale hierarchical image database*, in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255.

[Dubois *et al.*(2019)] R. Dubois, A. Imbert, et al. (2019), *A Deep Learning Approach To Identify mRNA Localization Patterns*, in *2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019)*, pp. 1386–1390.

[Dufour *et al.*(2005)] A. Dufour, V. Shinin, et al., *Segmenting and tracking fluorescent cells in dynamic 3-d microscopy with coupled active surfaces*, IEEE Transactions on Image Processing **14** (9), 1396 (2005).

[Eichenberger *et al.*(2021)] B. T. Eichenberger, Y. Zhan, et al., *deepBlink: threshold-independent detection and localization of diffraction-limited spots*, Nucleic Acids Research **49** (13), 7292 (2021).

[Eliscovich et Singer(2017)] C. Eliscovich et R. H. Singer, *Rnp transport in cell biology: the long and winding road*, Current Opinion in Cell Biology **45**, 38 (2017).

[Eng *et al.*(2019)] C.-H. L. Eng, M. Lawson, et al., *Transcriptome-scale super-resolved imaging in tissues by RNA seqFISH+*, Nature **568** (7751), 235 (2019).

[Ershov *et al.*(2022)] D. Ershov, M.-S. Phan, et al., *TrackMate 7: integrating state-of-the-art segmentation algorithms into tracking pipelines*, Nature Methods , 1 (2022).

[Ester *et al.*(1996)] M. Ester, H.-P. Kriegel, et al. (1996), *A density-based algorithm for discovering clusters in large spatial databases with noise*, in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96 (AAAI Press, Portland, Oregon) pp. 226–231.

[Fazal *et al.*(2019)] F. M. Fazal, S. Han, et al., *Atlas of Subcellular RNA Localization Revealed by APEX-Seq*, Cell **178** (2), 473 (2019).

[Femino *et al.*(1998)] A. M. Femino, F. S. Fay, et al., *Visualization of single rna transcripts in situ*, Science **280** (5363), 585 (1998).

[Gal et Ghahramani(2016)] Y. Gal et Z. Ghahramani (2016), *Dropout as a bayesian approximation: Representing model uncertainty in deep learning*, in *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16 (JMLR.org) p. 10501059.

[Glory et Murphy(2007)] E. Glory et R. F. Murphy, *Automated subcellular location determination and high-throughput microscopy.*, Developmental cell **12** (1), 7 (2007).

[Godinez *et al.*(2017)] W. J. Godinez, I. Hossain, et al., *A multi-scale convolutional neural network for phenotyping high-content cellular images*, Bioinformatics **33** (13), 2010 (2017).

[Goodfellow *et al.*(2016)] I. Goodfellow, Y. Bengio, et A. Courville (2016), *Deep Learning* (MIT Press).

[Grainger et Willert(2018)] S. Grainger et K. Willert, *Mechanisms of wnt signaling and control*, Wiley Interdisciplinary Reviews: Systems Biology and Medicine **10** (2018).

[Greenwald *et al.*(2022)] N. F. Greenwald, G. Miller, et al., *Whole-cell segmentation of tissue images with human-level performance using large-scale data annotation and deep learning*, Nature Biotechnology **40** (4), 555 (2022).

[Groisman *et al.*(2000)] I. Groisman, Y.-S. Huang, et al., *Cpeb, maskin, and cyclin b1 mrna at the mitotic apparatus: Implications for local translational control of cell division*, Cell **103** (3), 435 (2000).

[Grover et Leskovec(2016)] A. Grover et J. Leskovec (2016), *Node2vec: Scalable feature learning for networks*, in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16 (Association for Computing Machinery, New York, NY, USA) p. 855864.

[Hamaguchi *et al.*(2018)] R. Hamaguchi, A. Fujita, et al. (2018), *Effective use of dilated convolutions for segmenting small object instances in remote sensing imagery*, in *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 1442–1450.

[Haralick *et al.*(1973)] R. M. Haralick, Dinstein, et K. Shanmugam, *Textural features for image classification*, IEEE Transactions on Systems, Man, and Cybernetics **SMC-3**, 610 (1973).

[Harris *et al.*(2020)] C. R. Harris, K. J. Millman, et al., *Array programming with NumPy*, Nature **585**, 357362 (2020).

[Hastie *et al.*(2009)] T. Hastie, R. Tibshirani, et J. H. Friedman (2009), *The Elements of Statistical Learning: Data Mining, Inference, and Prediction* (Springer).

[He *et al.*(2017)] K. He, G. Gkioxari, et al. (2017), *Mask r-cnn*, in *Proceedings of the IEEE International Conference on Computer Vision (ICCV).*

[He *et al.*(2016)] K. He, X. Zhang, et al. (2016), *Deep residual learning for image recognition*, in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778.

[Hedlund et Deng(2018)] E. Hedlund et Q. Deng, *Single-cell rna sequencing: Technical advancements and biological applications*, Molecular Aspects of Medicine **59**, 36 (2018).

[Held *et al.*(2010)] M. Held, M. H. A. Schmitz, et al., *CellCognition: time-resolved phenotype annotation in high-throughput live cell imaging*, Nature Methods **7** (9), 747 (2010).

[Hervé *et al.*(2004)] C. Hervé, I. Mickleburgh, et J. Hesketh, *Zipcodes and postage stamps: mRNA localisation signals and their trans-acting binding proteins*, Briefings in Functional Genomics **3** (3), 240 (2004).

[Hollandi *et al.*(2020)] R. Hollandi, A. Szkalisity, et al., *nucleAIzer: A Parameter-free Deep Learning Framework for Nucleus Segmentation Using Image Style Transfer*, Cell Systems **10** (5), 453 (2020).

[Huang *et al.*(2017)] G. Huang, Z. Liu, et al. (2017), *Densely connected convolutional networks*, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

[Hubstenberger *et al.*(2017)] A. Hubstenberger, M. Courel, et al., *P-body purification reveals the condensation of repressed mrna regulons*, Molecular Cell **68** (1), 144 (2017).

[Hunter(2007)] J. D. Hunter, *Matplotlib: A 2D Graphics Environment*, Computing in Science and Engineering **9** (3), 90 (2007).

[Iandola *et al.*(2016)] F. N. Iandola, S. Han, et al. (2016), *Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <0.5mb model size*.

[Imbert *et al.*(2022a)] A. Imbert, F. Mueller, et T. Walter (2022a), *Pointfish: learning point cloud representations for rna localization patterns*, in *2022 European Conference on Computer Vision (ECCV 2022) Workshop on BioImage Computing.*

[Imbert *et al.*(2022b)] A. Imbert, W. Ouyang, et al., *Fish-quant v2: a scalable and modular tool for smfish image analysis*, RNA 10.1261/rna.079073.121 (2022b).

[Jones *et al.*(2009)] T. R. Jones, A. E. Carpenter, et al., *Scoring diverse cellular morphologies in image-based screens with iterative feedback and machine learning.*, Proceedings of the National Academy of Sciences of the United States of America **106** (6), 1826 (2009).

[Jones *et al.*(2008)] T. R. Jones, I. H. Kang, et al., *CellProfiler Analyst: data exploration and analysis software for complex image-based screens.*, BMC bioinformatics **9**, 482 (2008).

[Joulin *et al.*(2016)] A. Joulin, E. Grave, et al. (2016), *Bag of tricks for efficient text classification*.

[Jumper *et al.*(2021)] J. Jumper, R. Evans, et al., *Highly accurate protein structure prediction with AlphaFold*, Nature **596** (7873), 583 (2021).

[Jung *et al.*(2014)] H. Jung, C. Gkogkas, et al., *Remote control of gene function by local translation*, Cell **157** (1), 26 (2014).

[Jung *et al.*(2012)] H. Jung, B. C. Yoon, et C. E. Holt, *Axonal mRNA localization and local protein synthesis in nervous system assembly, maintenance and repair*, Nature Reviews Neuroscience **13** (5), 308 (2012).

[Jupyter *et al.*(2018)] Jupyter, M. Bussonnier, et al. (2018), *Binder 2.0 - reproducible, interactive, sharable environments for science at scale.*

[Kass *et al.*(1988)] M. Kass, A. Witkin, et D. Terzopoulos, *Snakes: Active contour models*, International Journal of Computer Vision **1** (4), 321 (1988).

[Khater *et al.*(2019)] I. M. Khater, S. T. Aroca-Ouellette, et al., *Caveolae and scaffold detection from single molecule localization microscopy data using deep learning*, PLOS ONE **14** (8), e0211659 (2019).

[Kingma et Ba(2015)] D. P. Kingma et J. Ba, *Adam: A method for stochastic optimization*, CoRR **abs/1412.6980** (2015).

[Krizhevsky *et al.*(2012)] A. Krizhevsky, I. Sutskever, et G. E. Hinton (2012), *Imagenet classification with deep convolutional neural networks*, in *Advances in Neural Information Processing Systems*, Vol. 25 (Curran Associates, Inc.).

[Lagache *et al.*(2015)] T. Lagache, N. Sauvonnet, et al., *Statistical analysis of molecule colocalization in bioimaging*, Cytometry Part A **87** (6), 568 (2015).

[Lalit *et al.*(2021)] M. Lalit, P. Tomancak, et F. Jug (2021), *Embedding-based instance segmentation in microscopy*, in *Proceedings of the Fourth Conference on Medical Imaging with Deep Learning*, Proceedings of Machine Learning Research, Vol. 143 (PMLR) pp. 399–415.

[Lawrence et Singer(1986)] J. B. Lawrence et R. H. Singer, *Intracellular localization of messenger rnas for cytoskeletal proteins*, Cell **45** (3), 407 (1986).

[LeCun *et al.*(2015)] Y. LeCun, Y. Bengio, et G. Hinton, *Deep learning*, Nature **521** (7553), 436 (2015).

[LeCun *et al.*(1989)] Y. LeCun, B. Boser, et al., *Backpropagation applied to handwritten zip code recognition*, Neural Computation **1** (4), 541 (1989).

[Lee *et al.*(2014)] J. H. Lee, E. R. Daugharthy, et al., *Highly multiplexed subcellular rna sequencing in situ*, Science **343** (6177), 1360 (2014).

[Levet *et al.*(2019)] F. Levet, G. Julien, et al., *A tessellation-based colocalization analysis approach for single-molecule localization microscopy*, Nature Communications **10** (1), 2379 (2019).

[Li *et al.*(2018)] Y. Li, R. Bu, et al. (2018), *Pointcnn: Convolution on x-transformed points*, in *Advances in Neural Information Processing Systems*, Vol. 31 (Curran Associates, Inc.).

[Lin *et al.*(2014)] T.-Y. Lin, M. Maire, et al. (2014), *Microsoft coco: Common objects in context*, in *Computer Vision – ECCV 2014* (Springer International Publishing, Cham) pp. 740–755.

[Lindeberg(2015)] T. Lindeberg, *Image Matching Using Generalized Scale-Space Interest Points*, Journal of Mathematical Imaging and Vision **52** (1), 3 (2015).

[Liu *et al.*(2018)] R. Liu, J. Lehman, et al. (2018), *An intriguing failing of convolutional neural networks and the coordconv solution*, in *Advances in Neural Information Processing Systems*, Vol. 31 (Curran Associates, Inc.).

[Liu *et al.*(2021)] Z. Liu, J. H. Liew, et al. (2021), *Dance : A deep attentive contour model for efficient instance segmentation*, in *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 345–354.

[Lowe(1999)] D. Lowe (1999), *Object recognition from local scale-invariant features*, in *Proceedings of the Seventh IEEE International Conference on Computer Vision*, Vol. 2, pp. 1150–1157 vol.2.

[Lubeck *et al.*(2014)] E. Lubeck, A. F. Coskun, et al., *Single-cell in situ RNA profiling by sequential hybridization*, Nature Methods **11** (4), 360 (2014).

[Lécuyer *et al.*(2007)] E. Lécuyer, H. Yoshida, et al., *Global analysis of mRNA localization reveals a prominent role in organizing cellular architecture and function*, Cell **131** (1), 174 (2007).

[Ma *et al.*(2022)] X. Ma, C. Qin, et al. (2022), *Rethinking network design and local geometry in point cloud: A simple residual MLP framework*, in *International Conference on Learning Representations*.

[van der Maaten et Hinton(2008)] L. van der Maaten et G. Hinton, *Visualizing data using t-sne*, Journal of Machine Learning Research **9** (86), 2579 (2008).

[Machairas *et al.*(2014)] V. Machairas, E. Decencière, et T. Walter (2014), *Waterpixels: Superpixels based on the watershed transformation*, in *2014 IEEE International Conference on Image Processing (ICIP)*, pp. 4343–4347.

[Mah *et al.*(2022)] C. K. Mah, N. Ahmed, et al. (2022), *Bento: A toolkit for subcellular analysis of spatial transcriptomics data.*

[Maturana et Scherer(2015)] D. Maturana et S. Scherer (2015), *Voxnet: A 3d convolutional neural network for real-time object recognition*, in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 922–928.

[McInnes *et al.*(2018)] L. McInnes, J. Healy, et al., *Umap: Uniform manifold approximation and projection*, Journal of Open Source Software **3** (29), 861 (2018).

[Wes McKinney(2010)] Wes McKinney (2010), *Data Structures for Statistical Computing in Python*, in *Proceedings of the 9th Python in Science Conference*, pp. 56 – 61.

[McQuin *et al.*(2018)] C. McQuin, A. Goodman, et al., *CellProfiler 3.0: Next-generation image processing for biology*, PLOS Biology **16** (7), e2005970 (2018).

[Medioni *et al.*(2012)] C. Medioni, K. Mowry, et F. Besse, *Principles and roles of mRNA localization in animal development*, Development **139** (18), 3263 (2012).

[Melton(1987)] D. A. Melton, *Translocation of a localized maternal mRNA to the vegetal pole of Xenopus oocytes*, Nature **328** (6125), 80 (1987).

[Mikolov *et al.*(2013)] T. Mikolov, K. Chen, et al. (2013), *Efficient estimation of word representations in vector space*.

[Mueller *et al.*(2013)] F. Mueller, A. Senecal, et al., *FISH-quant: automatic counting of transcripts in 3D FISH images*, Nature Methods **10** (4), 277 (2013).

[Müller *et al.*(2013)] C. Müller, N. Bauer, et al., *Making myelin basic protein -from mrna transport to localized translation*, Frontiers in Cellular Neuroscience **7**, 10.3389/fncel.2013.00169 (2013).

[Naylor *et al.*(2019)] P. Naylor, M. Laé, et al., *Segmentation of nuclei in histopathology images by deep regression of the distance map*, IEEE Transactions on Medical Imaging **38** (2), 448 (2019).

[Neven *et al.*(2019)] D. Neven, B. D. Brabandere, et al. (2019), *Instance segmentation by jointly optimizing spatial embeddings and clustering bandwidth*, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[O'Connor et Adams(2010)] C. O'Connor et J. U. Adams (2010), *Essentials of cell biology*.

[Odena *et al.*(2016)] A. Odena, V. Dumoulin, et C. Olah, *Deconvolution and checkerboard artifacts*, Distill 10.23915/distill.00003 (2016).

[Olah *et al.*(2017)] C. Olah, A. Mordvintsev, et L. Schubert, *Feature visualization*, Distill 10.23915/distill.00007 (2017).

[Olah *et al.*(2018)] C. Olah, A. Satyanarayan, et al., *The building blocks of interpretability*, Distill 10.23915/distill.00010 (2018).

[Oleynikov et Singer(2003)] Y. Oleynikov et R. H. Singer, *Real-time visualization of zbp1 association with -actin mrna during transcription and localization*, Current Biology **13** (3), 199 (2003).

[Olivo-Marin(2002)] J.-C. Olivo-Marin, *Extraction of spots in biological images using multiscale products*, Pattern Recognition **35** (9), 1989 (2002).

[Otsu(1979)] N. Otsu, *A threshold selection method from gray-level histograms*, IEEE Transactions on Systems, Man, and Cybernetics **9** (1), 62 (1979).

[Ounkomol *et al.*(2018)] C. Ounkomol, S. Seshamani, et al., *Label-free prediction of three-dimensional fluorescence images from transmitted-light microscopy*, Nature Methods **15** (11), 917 (2018).

[Ouyang *et al.*(2019a)] W. Ouyang, F. Mueller, et al., *ImJoy: an open-source computational platform for the deep learning era*, Nature Methods **16** (12), 1199 (2019a).

[Ouyang *et al.*(2019b)] W. Ouyang, C. F. Winsnes, et al., *Analysis of the Human Protein Atlas Image Classification competition*, Nature Methods **16** (12), 1254 (2019b).

[Parker *et al.*(2017)] N. Parker, M. Schneegurt, et al. (2017), *Microbiology*, Open Textbook Library (OpenStax).

[Partel et Wählby(2021)] G. Partel et C. Wählby, *Spage2vec: Unsupervised representation of localized spatial gene expression signatures*, The FEBS Journal **288** (6), 1859 (2021).

[Pedregosa *et al.*(2011)] F. Pedregosa, G. Varoquaux, et al., *Scikit-learn: Machine learning in Python*, Journal of Machine Learning Research **12**, 2825 (2011).

[Peng *et al.*(2020)] S. Peng, W. Jiang, et al. (2020), *Deep snake for real-time instance segmentation*, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Perkel(2019)] J. M. Perkel, *Starfish enterprise: finding RNA patterns in single cells*, Nature **572** (7770), 549 (2019).

[Pichon *et al.*(2016)] X. Pichon, A. Bastide, et al., *Visualization of single endogenous polysomes reveals the dynamics of translation in live human cells*, Journal of Cell Biology **214** (6), 769 (2016).

[Pichon *et al.*(2018)] X. Pichon, M. Lagha, et al., *A growing toolbox to image gene expression in single cells: Sensitive approaches for demanding challenges*, Molecular Cell **71** (3), 468 (2018).

[Pichon *et al.*(2021)] X. Pichon, K. Moissoglu, et al., *The kinesin KIF1C transports APC-dependent mRNAs to cell protrusions*, RNA **27** (12), 1528 (2021).

[Pillai *et al.*(2005)] R. S. Pillai, S. N. Bhattacharyya, et al., *Inhibition of translational initiation by let-7 microrna in human cells*, Science **309** (5740), 1573 (2005).

[Poser *et al.*(2008)] I. Poser, M. Sarov, et al., *BAC TransgeneOmics: a high-throughput method for exploration of protein function in mammals*, Nature Methods **5** (5), 409 (2008).

[Price-Whelan *et al.*(2018)] A. M. Price-Whelan, B. M. Sipőcz, et al., *The astropy project: Building an open-science project and status of the v2.0 core package*, The Astronomical Journal **156** (3), 123 (2018).

[Qi *et al.*(2017a)] C. R. Qi, H. Su, et al. (2017a), *Pointnet: Deep learning on point sets for 3d classification and segmentation*, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

[Qi *et al.*(2017b)] C. R. Qi, L. Yi, et al. (2017b), *Pointnet++: Deep hierarchical feature learning on point sets in a metric space*, in *Advances in Neural Information Processing Systems*, Vol. 30 (Curran Associates, Inc.).

[Redmon *et al.*(2016)] J. Redmon, S. Divvala, et al. (2016), *You only look once: Unified, real-time object detection*, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

[Reeve et Prokop(1992)] R. J. Reeve et A. P. Prokop, *A Survey of Moment-Based Techniques For Unoccluded Object Representation and Recognition*, CVGIP: Graphical Models and Image Processing , 438 (1992).

[Ren et Malik(2003)] Ren et Malik (2003), *Learning a classification model for segmentation*, in *Proceedings Ninth IEEE International Conference on Computer Vision*, pp. 10–17 vol.1.

[Ripley(2005)] B. Ripley (2005), *Spatial Statistics*, Wiley Series in Probability and Statistics (Wiley).

[Robinson et Whelan(2004)] K. Robinson et P. F. Whelan, *Efficient morphological reconstruction: a downhill filter*, Pattern Recognition Letters **25** (15), 1759 (2004).

[Ronneberger *et al.*(2015)] O. Ronneberger, P. Fischer, et T. Brox (2015), *U-net: Convolutional networks for biomedical image segmentation*, in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015* (Springer International Publishing, Cham) pp. 234–241.

[Rumelhart *et al.*(1986)] D. E. Rumelhart, G. E. Hinton, et R. J. Williams, *Learning representations by back-propagating errors*, Nature **323** (6088), 533 (1986).

[Safieddine *et al.*(2022)] A. Safieddine, E. Coleno, et al., *HT-smFISH: a cost-effective and flexible workflow for high-throughput single-molecule RNA imaging*, Nature Protocols , 1 (2022).

[Safieddine *et al.*(2021)] A. Safieddine, E. Coleno, et al., *A choreography of centrosomal mRNAs reveals a conserved localization mechanism involving active polysome transport*, Nature Communications **12** (1), 1352 (2021).

[Samacoits *et al.*(2018)] A. Samacoits, R. Chouaib, et al., *A computational framework to study sub-cellular RNA localization*, Nature Communications **9** (1), 4584 (2018).

[Savulescu *et al.*(2021a)] A. F. Savulescu, E. Bouilhol, et al., *Prediction of rna subcellular localization: Learning from heterogeneous data sources*, iScience **24** (11), 103298 (2021a).

[Savulescu *et al.*(2021b)] A. F. Savulescu, R. Brackin, et al., *Interrogating RNA and protein spatial subcellular distribution in smFISH data with DypFISH*, Cell Reports Methods **1** (5), 100068 (2021b).

[Savulescu *et al.*(2019)] A. F. Savulescu, R. Brackin, et al. (2019), *DypFISH: Dynamic Patterned FISH to Interrogate RNA and Protein Spatial and Temporal Subcellular Distribution*.

[Schena *et al.*(1995)] M. Schena, D. Shalon, et al., *Quantitative monitoring of gene expression patterns with a complementary dna microarray*, Science **270** (5235), 467 (1995).

[Schindelin *et al.*(2012)] J. Schindelin, I. Arganda-Carreras, et al., *Fiji: an open-source platform for biological-image analysis*, Nature Methods **9** (7), 676 (2012).

[Schmidt *et al.*(2018)] U. Schmidt, M. Weigert, et al. (2018), *Cell detection with star-convex polygons*, in *Medical Image Computing and Computer Assisted Intervention - MICCAI 2018 - 21st International Conference, Granada, Spain, September 16-20, 2018, Proceedings, Part II*, pp. 265–273.

[Sepulveda *et al.*(2018)] G. Sepulveda, M. Antkowiak, et al., *Co-translational protein targeting facilitates centrosomal recruitment of pcnt during centrosome maturation in vertebrates*, eLife **7**, e34959 (2018).

[Serra(1983)] J. Serra (1983), *Image Analysis and Mathematical Morphology* (Academic Press, Inc., Orlando, FL, USA).

[Smal *et al.*(2010)] I. Smal, M. Loog, et al., *Quantitative Comparison of Spot Detection Methods in Fluorescence Microscopy*, IEEE Transactions on Medical Imaging **29** (2), 282 (2010).

[Soille(2003)] P. Soille (2003), *Morphological Image Analysis: Principles and Applications*, 2nd ed.

[Stamos et Weis(2013)] J. L. Stamos et W. I. Weis, *The -Catenin Destruction Complex*, Cold Spring Harbor Perspectives in Biology **5** (1), a007898 (2013).

[Stoeger *et al.*(2015)] T. Stoeger, N. Battich, et al., *Computer vision for image-based transcriptomics*, Methods Inferring Gene Regulatory Interactions from Quantitative High-Throughput Measurements, **85**, 44 (2015).

[Stringer *et al.*(2021)] C. Stringer, T. Wang, et al., *Cellpose: a generalist algorithm for cellular segmentation*, Nature Methods **18** (1), 100 (2021).

[Stueland *et al.*(2019)] M. Stueland, T. Wang, et al., *RDI Calculator: An Analysis Tool to Assess RNA Distributions in Cells*, Scientific Reports **9** (1), 8267 (2019).

[Sullivan *et al.*(2018)] D. P. Sullivan, C. F. Winsnes, et al., *Deep learning is combined with massive-scale citizen science to improve large-scale image classification*, Nature Biotechnology **36** (9), 820 (2018).

[Szegedy *et al.*(2016)] C. Szegedy, V. Vanhoucke, et al. (2016), *Rethinking the inception architecture for computer vision*, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Tan et Le(2019)] M. Tan et Q. Le (2019), *EfficientNet: Rethinking model scaling for convolutional neural networks*, in *Proceedings of the 36th International Conference on Machine Learning*, Proceedings of Machine Learning Research, Vol. 97 (PMLR) pp. 6105–6114.

[Thomas *et al.*(2019)] H. Thomas, C. R. Qi, et al. (2019), *Kpconv: Flexible and deformable convolution for point clouds*, in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

[Tsanov *et al.*(2016)] N. Tsanov, A. Samacoits, et al., *smiFISH and FISH-quant a flexible single RNA detection approach with super-resolution capability*, Nucleic Acids Research **44** (22), e165 (2016).

[Uhlmann *et al.*(2016)] V. Uhlmann, S. Singh, et A. E. Carpenter, *CP-CHARM: Segmentation-free image classification made accessible*, BMC Bioinformatics **17** (1), 10.1186/s12859-016-0895-y (2016).

[Uhlén *et al.*(2015)] M. Uhlén, L. Fagerberg, et al., *Tissue-based map of the human proteome*, Science **347** (6220), 1260419 (2015).

[Van Driesche et Martin(2018)] S. J. Van Driesche et K. C. Martin, *New frontiers in rna transport and local translation in neurons*, Developmental Neurobiology **78** (3), 331 (2018).

[Varoquaux(2018)] G. Varoquaux, *Cross-validation failure: Small sample sizes lead to large error bars*, NeuroImage **180**, 68 (2018).

[Varoquaux et Cheplygina(2022)] G. Varoquaux et V. Cheplygina, *Machine learning for medical imaging: methodological failures and recommendations for the future*, npj Digital Medicine **5** (1), 1 (2022).

[Vaswani *et al.*(2017)] A. Vaswani, N. Shazeer, et al. (2017), *Attention is all you need*, in *Advances in Neural Information Processing Systems*, Vol. 30 (Curran Associates, Inc.).

[Vincent et Soille(1991)] L. Vincent et P. Soille, *Watersheds in digital spaces: an efficient algorithm based on immersion simulations*, IEEE Transactions on Pattern Analysis and Machine Intelligence **13** (6), 583 (1991).

[Virtanen *et al.*(2020)] P. Virtanen, R. Gommers, et al., *SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python*, Nature Methods **17**, 261 (2020).

[Walker et Jackway(1996)] R. F. Walker et P. Jackway (1996), *Statistical geometric features - extensions for cytological texture analysis*, in *ICPR - International Conference on Pattern Recognition*.

[Walt *et al.*(2014)] S. v. d. Walt, J. L. Schönberger, et al., *scikit-image: image processing in Python*, PeerJ **2**, e453 (2014).

[Walter(2020)] T. Walter (2020), *Bioimage informatics for phenomics.*

[Walter *et al.*(2010)] T. Walter, M. Held, et al., *Automatic identification and clustering of chromosome phenotypes in a genome wide RNAi screen by time-lapse imaging.*, Journal of structural biology **170** (1), 1 (2010).

[Walter *et al.*(2007)] T. Walter, P. Massin, et al., *Automatic detection of microaneurysms in color fundus images.*, Medical Image Analysis **11** (6), 555 (2007).

[Wang *et al.*(2008)] J. Wang, X. Zhou, et al., *Cellular phenotype recognition for high-content RNA interference genome-wide screening.*, Journal of biomolecular screening **13** (1), 29 (2008).

[Wang *et al.*(2017)] T. Wang, S. Hamilla, et al., *Extracellular matrix stiffness and cell contractility control RNA localization to promote cell migration*, Nature Communications **8** (1), 896 (2017).

[Wang *et al.*(2019)] Y. Wang, Y. Sun, et al., *Dynamic graph cnn for learning on point clouds*, ACM Trans. Graph. **38** (5), 10.1145/3326362 (2019).

[Wattenberg *et al.*(2016)] M. Wattenberg, F. Viégas, et I. Johnson, *How to use t-sne effectively*, Distill 10.23915/distill.00002 (2016).

[Wilk *et al.*(2016)] R. Wilk, J. Hu, et al., *Diverse and pervasive subcellular distributions for both coding and long noncoding RNAs*, Genes & Development **30** (5), 594 (2016).

[Wu *et al.*(2016)] B. Wu, C. Eliscovich, et al., *Translation dynamics of single mrnas in live cells and neurons*, Science **352** (6292), 1430 (2016).

[Wu et Akhmanova(2017)] J. Wu et A. Akhmanova, *Microtubule-organizing centers*, Annual Review of Cell and Developmental Biology **33** (1), 51 (2017).

[Wu *et al.*(2005)] K. Wu, E. Otoo, et A. Shoshani (2005), *Optimizing connected component labeling algorithms*, in *Medical Imaging 2005: Image Processing*, Vol. 5747, International Society for Optics and Photonics (SPIE) pp. 1965–1976.

[Wu *et al.*(2019)] W. Wu, Z. Qi, et L. Fuxin (2019), *Pointconv: Deep convolutional networks on 3d point clouds*, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).*

[Wählby *et al.*(2002)] C. Wählby, J. Lindblad, et al., *Algorithms for cytoplasm segmentation of fluorescence labelled cells.*, Analytical cellular pathology : the journal of the European Society for Analytical Cellular Pathology **24** (2-3), 101 (2002).

[Xia *et al.*(2019)] C. Xia, J. Fan, et al., *Spatial transcriptome profiling by merfish reveals subcellular rna compartmentalization and cell cycle-dependent gene expression*, Proceedings of the National Academy of Sciences **116** (39), 19490 (2019).

[Zhao *et al.*(2021)] H. Zhao, L. Jiang, et al. (2021), *Point transformer*, in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 16259–16268.

RÉSUMÉ

Tous les ARNs ne se sont pas uniformément distribués dans la cellule. Pour certains gènes, une distribution asymétrique et localisée des transcripts peut être observée. Ce phénomène participe à la régulation de l'expression génétique. Actuellement, pour étudier la localisation intracellulaire des ARNs, une méthode appropriée est le *single molecule FISH* (smFISH). Les molécules d'ARN sont ciblées avec des marqueurs fluorescents et peuvent ainsi être observées au microscope. Cette technique, et ses variantes, permet d'étudier des milliers de transcripts à l'échelle de la cellule. Toutefois, les outils informatiques utilisées pour ces expériences présentent des limites. Lors d'une expérience avec des images de smFISH, une analyse complète nécessite souvent d'intégrer des outils différents, de les calibrer et de développer du code propre à l'étude. Cela pénalise le passage à l'échelle sur des études avec un grand volume de données, ainsi que leur reproductibilité. Avec cette thèse, je présente une nouvelle version de FISH-quant, un outil destiné à l'analyse des images de smFISH. Les actions les plus communes et nécessaires pour traiter les images d'ARNs y sont disponibles et améliorées : la détection des ARNs, la segmentation des cellules et des noyaux, ainsi qu'une sélection d'indicateurs statistiques, le tout réuni dans un seul outil. En outre, nous avons étudié la possibilité d'entraîner un modèle d'apprentissage à partir de nuages d'ARNs simulés, afin de pouvoir classifier différents schéma de localisation. Enfin, nous avons utilisé FISH-quant dans différentes études avec des données réelles. Ces expériences ont permis d'étudier plusieurs dizaines de gènes à travers plus de 100000 cellules identifiées. Nous avons d'abord développé un modèle de classification pour reconnaître les schémas de localisation d'ARN les plus fréquents. Ensuite, nous nous sommes intéressés à quantifier et décrire statistiquement différents types de localisation autour du centrosome ou dans les extensions périphériques de la cellule. Nous avons notamment caractérisé des schémas de localisation liés à la traduction des protéines ou au cycle cellulaire.

MOTS CLÉS

Transcriptomique, Vision par ordinateur, Apprentissage automatique, Localisation de l'ARN, Nuage de points.

ABSTRACT

RNAs do not necessarily have a random distribution in the cell. Specific localization patterns have been observed and such asymmetric distributions are of functional importance for the spatial regulation of gene expression. Current methods to study intra-cellular RNA localization use single molecule FISH (smFISH) protocols or its multiplexing variants. Each RNA molecule is targeted with fluorescent probes and can be imaged by fluorescence microscopy. These techniques enable single cell level analyses and can be scaled to thousands of transcripts. However, computational methods to quantify RNA localization suffer different flaws, such as the consistency of spot detection between experiments. A complete analysis pipeline typically requires the use of different frameworks and the development of in-house pieces of code, making smFISH analysis difficult to scale and reproduce. In this thesis, I present a new version of FISH-quant, a comprehensive computational framework to process smFISH images. It provides improved modules for the most common operations needed for smFISH analysis such as RNA detection, nucleus and cell segmentation as well as feature engineering, all in the same framework, designed to be flexible and scalable. In addition, I investigate the use of point cloud models directly trained from simulations to address the classification of RNA localization patterns, without the need to compute hand-crafted features. Lastly, I apply FISH-quant to experimental datasets in biological studies that include several dozens of transcripts, analyzed through 100,000 individual cells. For this, I design a classification pipeline to discriminate several generic RNA localization patterns. This allows us to discover translation factories, a new mechanism of spatial control of gene expression. Third, we focus on the centrosomal and protrusion patterns to provide a statistical description of several genes of interest. Altogether, these studies reveal regulation mechanisms more complex than expected with various spatial and temporal dynamics.

KEYWORDS

Transcriptomics, Computer vision, Machine learning, RNA localization, Point cloud.