Untangling Overlapping Barcodes in Spatial Genomics Data

Thesis by
Jonathan Alexander White

In Partial Fulfillment of the Requirements for the
Degree of
Doctor of Philosophy in Bioengineering

CALIFORNIA INSTITUTE OF TECHNOLOGY
Pasadena, California

2026
Defended January 22, 2026

Jonathan Alexander White
ORCID: 0000-0002-7009-3332

ACKNOWLEDGEMENTS

First, I would like to thank my PhD advisor, Long Cai, who knows how to tell a compelling scientific story. He has helped me craft the tale I tell in the following pages. I would like to thank the members of my examining committee: Mitch Guttman, David Van Valen, and Matt Thomson, as well as the past members of my first-year conversation committee, Victoria Kostina and Lior Pachter, for their very helpful feedback. I would like to thank Jonathan Fox for assistance with animal procedures and tissue collection, Jina Yun for readout probe conjugation and experimental support, and Saori Lobbia for maintaining cell cultures and preparing cell samples. I would like to thank Linjing "Lynn" Fang for her advice and the rest of my lab mates for their camaraderie. I would like to thank Inna-Marie Strazhnik for illustrating figures for this thesis.

I would like to thank my parents, Janet Leventhal and David White, for teaching me to be curious about the natural world, supporting me, and encouraging me throughout my many years of study. I would also like to thank my grandparents, Isobel Leventhal, Louis Leventhal, Marilyn White, and George White, for supporting and encouraging me in my studies. Their legacy continues to inspire me to work for a better world. Their light has led my way, and their memory is a blessing.

I would also like to thank the many wonderful teachers that I've had over the years at Caltech, Swarthmore College, and the Athenian School. There are too many to list here, but each of them has helped shape me into the scholar I am today. In particular, I would like to thank my undergraduate thesis advisors, Brad Davidson and Carl Grossman. They taught me to be an independent scientist and helped me develop the skills to complete this work.

Finally, I would like to thank my wife, Heather Zhou, for coming with me on this journey to pursue our PhDs in the City of Angels/Roses and helping me with math when I need it.

ABSTRACT

Difficulty in resolving spatially overlapping barcodes is a major bottleneck for imaging-based spatial genomics methods. Here, we present an approach for untangling overlapping barcodes by using strong encoding and global optimization to reduce spurious solutions resulting from recombinations of barcodes. We demonstrate experimentally that cellular regions with average local densities of 127 barcodes per μm^2 can be decoded with an estimated FDR of less than 4%, enabling a new type of super-resolution microscopy by coding.

PUBLISHED CONTENT AND CONTRIBUTIONS

1. White, J. A., Lu, C., Ombelets, L. & Cai, L. Untangling overlapping barcodes in spatial genomics data. *bioRxiv*. eprint: https://www.biorxiv.org/content/early/2025/06/16/2025.06.10.658913.full.pdf. https://www.biorxiv.org/content/early/2025/06/16/2025.06.10.658913 (2025).
   J.A.W. and L.C. conceived the idea. J.A.W. wrote the seqFISH data processing packages, seqFISH data processing workflows, simulations, and codebook generation software. L.O. helped with the data processing workflows. J.A.W. and C.L designed the experiments. C.L. performed the experiments and the smFISH data processing. C.L. wrote the experimental methods section and J.A.W. wrote the manuscript with input from all authors.

TABLE OF CONTENTS

NOMENCLATURE

k **(parameter of an error-correcting code).** The number of information symbols contained in codewords of an error-correcting code.

n **(parameter of an error-correcting code).** The number of symbols used in codewords of an error-correcting code [1].

q **(parameter of an error-correcting code).** The size of the alphabet from which symbols are drawn in an error-correcting code [1].

barcode. The physical manifestation of a codeword in a seqFISH experiment. Barcodes consist of a collection of aligned dots found in different symbol blocks each of which represent a symbol in a codeword.

codebook. The set of codewords in an error-correcting code [1].

codeword. A sequence of n symbols drawn from an alphabet of size q that represents a piece of information encoded by an error-correcting code [1].

error-correcting code. A system including algorithms for robustly encoding information as codewords, a codebook of codewords, algorithms for robustly decoding codewords that may have been corrupted up to an acceptable degree to recover the originally encoded information [1].

information symbol. A symbol in a codeword that directly encodes information [1].

minimum Hamming distance (MHD) (of an error-correcting code). The minimum Hamming distance between any two codewords in an error-correcting code [1].

parity check symbol. A redundant symbol in a codeword that is computed by evaluating a parity check equation of the information symbols to add robustness to error correcting codes [1].

point spread function (PSF). The signal produced by diffraction-limited objects in microscopy images. In this thesis, I model point spread functions as Gaussian.

pseudo-color. The value of an error-correcting code symbol in a seqFISH experiment as determined by the particular symbol block image a dot appears in.

symbol block. A collection of FISH images that together encode a symbol from an error correcting code. We use one-hot encoding: barcodes contain at most one dot in each symbol block. Each image in a symbol block is called a

pseudo-color of that symbol block, similarly to how standard color images are composed of red, green, and blue images. Dots in a symbol block have the pseudo-color of the image in which they are found and represent a value of the symbol determined by their pseudo-color. A symbol block consists of q images in seqFISH implementations of error-correcting codes where all symbols of all values are readout or $q - 1$ images in implementations of seqFISH where zero-valued symbols are not readout and instead represented by the absence of a dot in a symbol block.

weight (w) (parameter of a codeword). The number of non-zero symbols in a codeword of an error-correcting code [1].

0.1 Introduction

Spatial genomics technologies measure gene expression of cells in their native tissue context. Imaging-based spatial transcriptomics, such as seqFISH [2], MERFISH [3], hybISS [4], and STARMAP [5] build on single-molecule fluorescent in situ hybridization (smFISH) [6, 7]. In FISH-based methods, sequential cycles of probe hybridization, imaging, and probe stripping (hybridization cycles) generate many images containing barcodes that uniquely identify RNA species. In seqFISH, the images are partitioned into blocks that each represent a symbol of an error-correcting code. We say that each image in a symbol block is a different pseudo-color of the symbol block image: all dots found in a block represent a symbol with a value determined by their pseudo-color number. Each gene is encoded by a unique sequence of symbols called a codeword. Barcodes, the physical manifestation of codewords, appear in seqFISH data as an aligned sequence of signals (dots) in different symbol blocks (Fig. 0.1A-B)[8–10].

We now give a brief history of encoding strategies for imaging-based spatial genomics experiments. The original seqFISH experiment encoded 12 genes using a repetition code to establish reproducibility of the hybridization cycles[2]. Each codeword symbol was represented by a fluorescent dot of a different optical color in a hybridization cycle image. Four fluorophores were used in two hybridization cycle images to uniquely represent each of the 12 genes, with a repetition to show reproducibility. The initial MERFISH study then encoded genes using Hamming codes, which perfectly pack codewords into symbol-sequence space given their specified minimum Hamming distance (MHD). They used both an MHD 4 Hamming code encoding 130 genes and correcting one error and an MHD 2 Hamming code that encoded 1,001 genes and detected but did not correct errors [3]. The next iteration of seqFISH introduced q-ary single parity check codes with four or five symbols (n = 4 or 5) for spatial genomics. As with the initial repetition code, dots from each hybridization encoded a symbol with its value determined by its optical color. Either the first three or four hybridization cycles served to represent information symbols (k = 3 or 4) and the final (fourth or fifth) hybridization represented a parity check symbol. With five optically distinguishable colors per hybridization (q = 5), this encoding method had a capacity to encode $q^k = 5^3 = 125$ or $q^k = 5^4 = 625$ genes [11]. RNA spots [8] and Intron seqFISH [9] extended this approach to increase multiplexing capacity by introducing pseudo-colors, where pseudo-colors may be probed in different hybridization cycles, to increase the alphabet size from which symbols are drawn (q) to beyond the number of colors that can be distinguished op-

tically in a single hybridization. The 2019 MERFISH paper used a non-linear code that achieves efficient codeword packing without the constraint of a linear structure [12]. Two-layer DNASeqFISH+ resolves genomic loci in cells by first barcoding regions of the genome encompassing 60 genomic loci of interest, reading out one of the loci in each region in each of 60 subsequent hybridization cycles [13]. Generally, MERFISH experiments pack coding sequence space tightly with codewords and resolve barcodes with expansion microscopy, whereas seqFISH uses codes that are sparser in coding sequence space and require more readout hybridization cycles but not expansion microscopy. In situ sequencing, another imaging-based spatial transcriptomics modality, reads out base pair sequences without using error-correcting codes [4].

We next give a brief history of the algorithms used to process smFISH and seqFISH images. The first demonstration of smFISH by Femino et al. identified transcripts as dots in deconvolved and thresholded images [6]. The second iteration of smFISH by Raj et al. instead applied a Laplacian of Gaussian filter, found local maxima, then manually chose a threshold for which the number of passing dots was insensitive to small changes of the threshold value for their weight parameters [7]. Processing seqFISH images, in addition to finding dots, requires aligning dots in different hybridization cycles and inferring which barcoded transcript (barcode) generated them. For the original seqFISH study, Lubeck et al. used Femino et al.'s method for finding PSFs, then corrected for drift between readout hybridization cycles by using phase cross-correlation of DAPI stains [14]. They then inferred that dots in different symbol blocks that were mutually nearest neighbors were generated from the same barcode. Intron seqFISH [9] used a version of Raj et al.'s dot finding algorithm followed by radial centers [15] to refine the estimated dot locations to sub-pixel resolution. The original seqFISH+ analysis [10] used this approach for finding dots and aligned by manually matching beads used as fiducial markers. An updated version of the mutual nearest neighbors decoding uses DAOStarFinder for dot detection and automates parameter setting using a support vector machine [16, 17]. Another recent method, Polaris, uses deep learning to eliminate user parameter tuning [18].

The initial MERFISH study [3] took a slightly different approach: they identified dots as PSFs using DAOSTORM [19], then used phase cross-correlation of DAPI stains for initial coarse alignment. For fine alignment, they then searched for the two closest fiducial markers between cross-correlation aligned images and used their

offsets to refine shift estimates. To identify barcodes, they looked for strings of PSFs across all hybridization cycles within a search radius of each other. Strings of PSFs corresponding to a codeword or error-correctable to a codeword were classified as barcodes, and all others were discarded. A previous work suggested a possible graph-based solution to resolve ambiguity from jitter[20].

Decoding methods discussed so far are known as spots-based decoding methods, since they begin by locating dots. Moffit et al. 2016 introduced pixel-based decoding which attempts to identify the barcode represented in each pixel. Recently, others have introduced hybrid methods for pixel-based decoding that also account for spot morphology. BarDensr uses regression to de-mix readout images to infer images of the signal produced by each kind of barcoded molecule individually, then uses peak finding to localize dots in de-mixed images [21]. Another approach, ISTDECO, deconvolves barcodes in images using a maximum likelihood approach [22]. JIST builds on these methods by using regularized regression to infer the location and identities of target molecules directly [23]. Another approach, Postcode, uses probabilistic programming to determine barcode identity after the locations of spots are identified in an anchor channel [24]. The STARFISH package implements some the previously discussed algorithms to help users construct their own image processing pipelines [25]. However, none of these approaches consider how to design barcodes so that they can be resolved when they overlap with one another or handle barcodes jittering between readout images.

0.2 Main Text

In this thesis, I introduce a new method to disentangle overlapping barcodes using global optimization with a spots-first variant (more robust to jitter and registration error) and a compressed sensing variant (more robust to overlapping dots in individual images). After preprocessing the images and fitting dots for the spots-first variant or finding dot locations for the compressed sensing variant (methods), we use dynamic programming to identify all sets of aligning dots that could represent a barcode. In dense datasets, recombinations of dots from two or more overlapping barcodes and dots originating from non-specific binding events can give rise to spurious solutions (Fig. 0.1C). We resolve this ambiguity by using global optimization to choose the best decoding solution. Our spots-first approach uses integer-linear programming with the constraint that no two barcodes containing a shared dot are allowed in a solution. The cost of each solution is calculated as a weighted sum of the position variances and log-brightness variances of dots comprising each barcode decoded in

the solution and the number of dots in the conflict network that are not decoded in the solution (Fig. 0.1D). Our compressed sensing approach instead uses LASSO regression to choose the subset of candidate barcodes whose expected signals best explain the observed pixel intensities in images of all readout hybridization cycles.

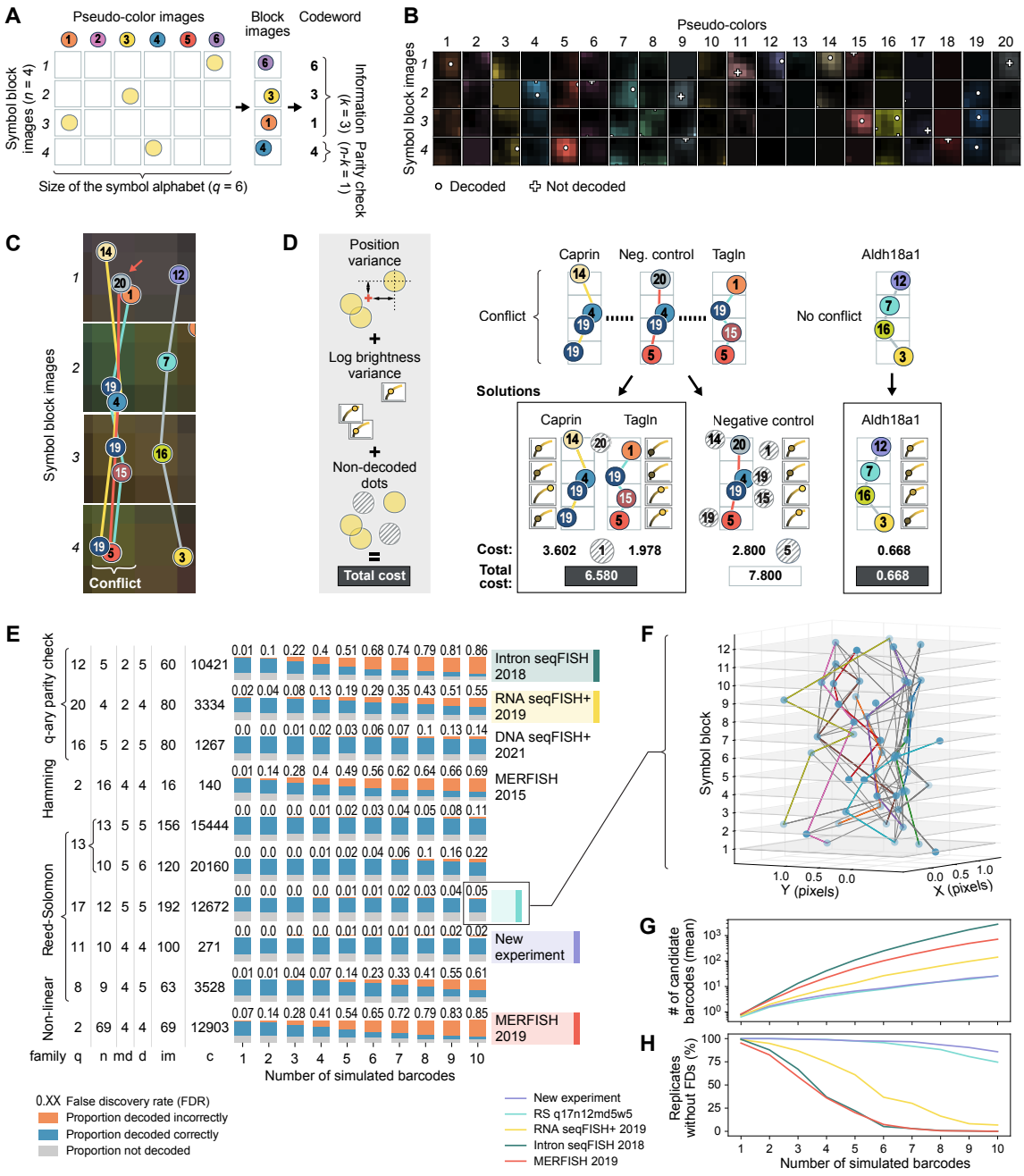

Figure 0.1:

Figure 0.1: Untangling spatially overlapping barcodes A) An illustration of how seqFISH encodes molecular identity. SeqFISH images are partitioned into symbol blocks (rows) in which symbol values (also called pseudo-colors, shown as columns) transmitted by a barcoded molecule are one-hot encoded. Individual molecules transmit signals that appear as a diffraction-limited dot of one pseudo-color in a symbol block. Sequences of spatially aligned dots of varying pseudo-colors in different symbol blocks represent a sequence of symbols called a codeword that encodes the identity of the barcoded molecule. Codewords are composed of information symbols, which alone specify molecular identity, and parity check symbols, which give the encoding robustness to error. Panels B-D show a decoding example of three superimposed barcodes from the 2019 seqFISH+ experiment. B) An aligned region of interest showing all 80 images in the experiment containing three superimposed barcodes and other fluorescent signals. Images are organized by symbol block (rows) and pseudo-color (columns). White circles are overlaid at the coordinates of fit dots that were decoded. White crosses are overlaid at the coordinates of fit dots that were not decoded. Panel C) shows candidate barcodes identified from the dots in panel B. Images of each symbol block show pixels pseudo-colored according to the aligned fluorescent signal of each pseudo-colored image. The locations of dots centered in the region of interest are marked by circles of their pseudo-color and labeled with the number of their pseudo-color. Candidate barcodes are connected by lines of a color unique to each barcode. The Aldh18a1 barcode (gray) is unambiguous, but the Tagln (blue), Caprin (yellow), and negative control (red) barcodes are conflicting. In this case, it appears that a non-specific dot, pseudo-color 20 in symbol block 1, introduces a spurious negative control barcode candidate straddling the Tagln and Caprin barcodes. D) To resolve the conflicts and find the most likely solution, we transform the network of conflicting barcodes into a graph where each node represents a barcode and edges represent conflicts. Feasible solutions cannot contain barcodes connected by an edge. Each solution is scored by summing the costs of the barcodes it contains and the costs of not decoding each dot that is not decoded. The lowest cost solution is chosen using integer programming. E) Decoding simulated data of numerous barcodes in a 1-pixel (100x100nm) area with our decoding algorithm determines the relative robustness of different error-correcting codes to spurious solutions arising from superimposed barcodes. The dendrogram on the left groups codes by their family first, then various parameters: q, symbol alphabet size; n, number of symbols; md, minimum Hamming distance between any pair of codewords; d, the number of dots are in each barcode; im, the number of images required for an experimental implementation; and c, the number of codewords used from the code. Each condition used 500 replicates. F) A simulation replicate containing 10 superimposed barcodes encoded by the q17 n12 Reed-Solomon code used to produce panel D was correctly decoded. The correct decoded barcodes are shown in colored paths. The gray paths are the candidate barcodes that are not decoded. In this simulation replicate, the algorithm correctly determined all the simulated genes. G-H) Panels G and H, respectively, show the number of candidate barcodes and the proportion of simulation replicates that were decoded without false discoveries for five selected error-correcting codes as found in simulations of different densities. The Reed-Solomon codes attain the highest robustness by using more images and stronger encoding to reduce the number of ways superimposed barcodes can be recombined into spurious candidate solutions.

To measure the robustness of various error-correcting codes to ambiguity arising from spatially overlapping barcodes, we simulated dot locations from multiple overlapping barcodes in a 1-pixel (100×100nm) region of interest and evaluated the spots-first decoding results for each error-correcting code (Fig. 0.1E and Supplementary Fig. D.2A). We analyzed seqFISH and MERFISH codes from the literature. SeqFISH used q-ary parity check family codes that draw symbols from an alphabet of size $q > 2$ and included one or two parity check symbols [9, 10, 26]. MERFISH used binary ($q = 2$) codes: either Hamming codes [3] or non-linear codes [12]. Our simulations showed that the MERFISH and intron seqFISH codes cannot tolerate more than one barcode per region, while the codes used in RNA and DNA seqFISH+ [10, 26] can tolerate two-four spatially overlapping barcodes (Fig. 0.1E).

To explore whether other codes can provide significant improvements, we analyzed Reed-Solomon (RS) codes [27]. Reed-Solomon codes are also q-ary codes. RS codes provide a flexible design framework for designing larger codes with larger minimum distances. The parity check equations in RS codes are designed using abstract algebra structures to ensure that the minimum distance between any two of their codewords (MHD) is one plus the number of parity check symbols, increasing robustness. Simulated data encoded using Reed-Solomon codes can be robustly decoded at densities of greater than five barcodes per search radius (Fig. 0.1E,F). A $q17$ code covering 12,000 genes can achieve decoding of 10 spatially overlapping barcodes with 5% false discovery rate (FDR).

A key determinant of how well an error-correcting code can distinguish many overlapping barcodes in a dense region is how likely dots from overlapping barcodes encoded with that error-correcting code are to recombine into other candidate barcodes. For example, the 2019 MERFISH non-linear code and a subset of the q11n10k7 Reed-Solomon code with 271 weight-four codewords have the same MHD, but the 2019 MERFISH non-linear code produces approximately 20 times as many candidate barcodes in simulations of overlapping barcodes because its codewords are packed more closely in coding space (Fig. 0.1G). Consistently, simulation replicates for the Reed-Solomon code were much more likely to be decoded without false discoveries than they were for the MERFISH code (Fig. 0.1H). Specifically, simulated data for 10 overlapping barcodes encoded with the Reed-Solomon code were decoded with an overall FDR of 2%, while simulated data for single barcodes, not spatially overlapping with other barcodes, using the 2019 MERFISH non-linear code were decoded with an FDR of 7% (this was alleviated experimentally by ex-

panding the sample). Reed-Solomon code performance scales well when adding more readout dots per barcode and more readout images. We find that Reed-Solomon codes with larger MHD are more robust to density, but lose robustness when using codewords of weight larger than their MHD and that the additional readout dots increase the number of ways that barcodes can be recombined into spurious solutions (Supplementary Fig. D.2).

Even though all codes can be represented in binary form, using symbol blocks with many pseudo-colors (q-ary) rather than binary codes increases robustness because only one dot from any given symbol block is allowed in any barcode. This rule reduces the similarity of codewords and reduces ambiguity when barcodes are superimposed. It also means that changing the value of a symbol from one non-zero value to another requires both a removal and an addition of a dot for a change of 1 Hamming distance. Such a change would be measured as 2 Hamming distance in a binary coding scheme. Additionally, thinning codebooks to reduce the number of barcodes under consideration reduces the number of ways to recombine dots into spurious barcodes (Supplementary Fig. D.2B).

To validate the simulation results, we applied this decoding method to published seqFISH+ data probing mRNAs of 10,000 genes in NIH 3T3 cells [10]. We measure the sensitivity of our decoding method for a range of parameters by regressing against the average transcript counts of 60 genes measured by sequential single-molecule FISH (255 and 288 cells, respectively) (Fig. 0.2A,D). Our spots-first processing workflow achieves 46% sensitivity and our compressed sensing workflow achieves 48% sensitivity at 5% estimated FDR. There was an experiment-wide average of 12 barcodes per micron of cell mask area in each independently encoded optical channel. 27% of decoded barcodes are located within 100 nm of at least one other barcode and would have been discarded by the mutual nearest neighbor method of decoding used in previous seqFISH studies. Another decoding pipeline (Py-FISH) that discards overlaps without attempting to resolve them achieved a sensitivity of 39% with 5% estimated FDR [17], a sensitivity that is lower by almost exactly the proportion of barcodes that are overlapping (27%).

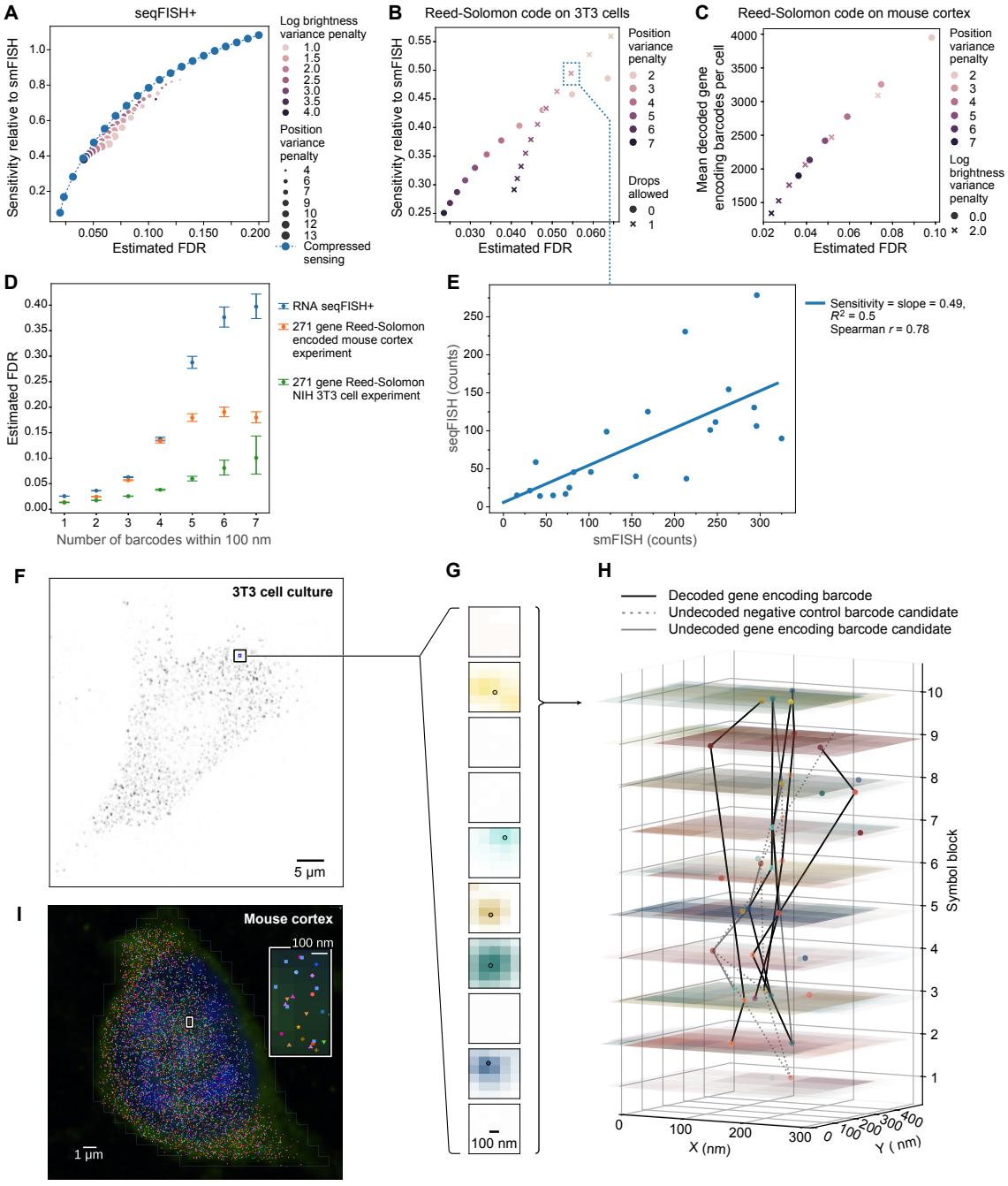
A
seqFISH+
Sensitivity relative to smFISH
Estimated FDR
Log brightness variance penalty
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Position variance penalty
4
6
7
9
10
12
13
Compressed sensing
B
Reed-Solomon code on 3T3 cells
Sensitivity relative to smFISH
Estimated FDR
Position variance penalty
2
3
4
5
6
7
Drops allowed
0
1
C
Reed-Solomon code on mouse cortex
Mean decoded gene encoding barcodes per cell
Estimated FDR
Position variance penalty
2
3
4
5
6
7
Log brightness variance penalty
0.0
2.0
D
Estimated FDR
Number of barcodes within 100 nm
RNA seqFISH+
271 gene Reed-Solomon encoded mouse cortex experiment
271 gene Reed-Solomon NIH 3T3 cell experiment
E
seqFISH (counts)
smFISH (counts)
Sensitivity = slope = 0.49,
R^2 = 0.5
Spearman r = 0.78
F
3T3 cell culture
5 µm
G
100 nm
H
Decoded gene encoding barcode
Undecoded negative control barcode candidate
Undecoded gene encoding barcode candidate
Symbol block
X (nm)
Y (nm)
I
Mouse cortex
100 nm
1 µm

Figure 0.2:

Figure 0.2: Experimental validation of our untangling approach with RNA seqFISH. A-C) Estimated sensitivity-FDR curves for decoding results on previously published RNA seqFISH+ data in NIH 3T3 fibroblasts (A), a new cross-channel Reed-Solomon encoded experiment in NIH 3T3 fibroblasts (B), and a new single-channel Reed-Solomon encoded mouse cortex experiment probing the same molecules as in the new 3T3 fibroblast experiment (C). Different parameters of the decoding objective functions tune the tradeoff in each experiment. Decoding results for all combinations of parameters in the mouse cortex experiment (C) were evaluated in only nine fields of view to save computation time. D) The estimated FDR of barcodes at different local densities (number of decoded barcodes including self within 100 nm search radius) using position variance penalty of 7 and other penalty parameters of zero for the Reed-Solomon encoded experiments and position variance penalty of 10.5 and log weight variance penalty of 4.0 for the previously published NIH 3T3 cell experiment. Error bars are a 95% confidence interval estimated by non-parametric bootstrap. E) The sensitivity relative to smFISH for the annotated data point in panel B, whose associated statistics were reported in text, was calculated. Decoding sensitivity is measured as the slope of the linear regression fit between average counts of genes found per cell by cross-channel Reed-Solomon encoded seqFISH against the average counts of the same genes found per cell by smFISH on NIH 3T3 cells taken from the same culture. This figure uses barcode counts found using the decoding parameters for which summary statistics are reported in the text. F) A single readout image of a cell from the cross-channel Reed-Solomon encoded 3T3 cell experiment showing dots of the first pseudo-color of the tenth symbol block. G) Images of the ten pseudo-colors of the inset region of the cell in panel F overlaid with fit dot locations in the tenth symbol block which are collapsed into a symbol block image in the top layer of panel H. H) An illustration of decoding results from the inset region of the cell shown in panel F from the cross-channel Reed-Solomon encoded experiment using a codebook including 1829 negative control codewords in addition to the 271 gene encoding codewords. Z-planes represent pseudo-colored symbol block images where each pseudo-color has been registered. Markers denote the fit locations and pseudo-colors of dots in each block. Dots connected by black lines represent decoded gene-encoding barcodes. Dots connected by solid gray lines represent undecoded candidate barcodes encoding for genes. Dots connected by dotted gray lines represent undecoded negative control candidate barcodes. No negative control barcodes were included in the lowest cost decoding solution for this example. I) A maximum projected image of a cell from the mouse cortex experiment with DAPI stain (blue) and poly-T mRNA FISH stain (green) overlaid with locations of decoded barcodes in all z-slices (markers).

To show experimentally that the untangling approach can decode high-density genes, we designed and performed a seqFISH experiment encoded using a Reed-Solomon code. The experiment probed 271 genes expressed at a sum-total of 43,528 FPKM out of a total of 442,744 FPKM expression for all genes in bulk RNA sequencing and an experiment-wide average of approximately 13 barcodes per square micron of cell mask area. Each transcript was read out in 4 out of 100 readout images acquired in 34 hybridization cycles across three optical channels. This contrasts with the previous 10,000 gene seqFISH+ experiment, which consisted of three separately encoded sub-experiments each probing 3,333 or 3,334 genes expressed at a total of ~41,000 FPKM over 80 hybridization cycles in a single optical channel. Even with chromatic aberration reducing the quality of cross-channel image alignment in the 271 gene experiment, we achieved 49% sensitivity relative to smFISH for 21 selected genes measured in cells taken from the same culture (217 and 218 cells, respectively), detecting a mean of 18,903 transcripts per cell (sample standard deviation=7,572) for all 271 measured genes, with an estimated FDR of 5% (Fig. 0.2B,D-H and Supplementary Fig. D.3). 32% of the barcodes were within 100 nm of another barcode and would have been discarded by mutual nearest neighbors decoding. Per-cell gene correlations calculated using the seqFISH data show a similar structure to that found using smFISH data (Supplementary Fig. D.3D-F).

To further validate our method, we used the Reed-Solomon encoded probe set to measure gene expression in the mouse cortex in a single optical channel (Fig. 0.2C-D,I), finding an average of 2,804 transcripts per cell over all fields of view. A 10X single-cell RNAseq dataset of the mouse cortex in the Allen Brain Atlas found an average of 564 UMI counts per cell for these genes and an average of 11,284 UMI counts per cell for all genes. The tissue was imaged in eight optical z-slices spaced at one micron, and the 2D algorithm was performed on each z-slice. The average barcode density per mask area over all z-slices was six barcodes per square micron using the least stringent decoding parameters in the nine FOVs on which all decoding parameters were tested. We compare the results of our experiment to sample commercial spatial transcriptomics mouse brain datasets acquired using MERFISH (Vizgen)[28], Cosmx (Bruker)[29], and Xenium (10X genomics)[30]. The sample CosMx and Xenium datasets include enough of the same genes as in our experiments to directly compare sensitivity. We find that our method has approximately four and ten times higher sensitivity, respectively (Supplementary Fig. D.4A). Furthermore, even though we did not select our gene panel to do so, we were able to predict whether or not each cell in our dataset was a neuron or not with

a median confidence of 99.5% using CONCORD label transfer from Allen Brain Cell Atlas single-cell sequencing data (Supplementary Fig. D.4B)[31, 32].

While spatial genomics borrows error-correcting codes from digital communications, the two fields operate in different regimes. In telecommunications, errors result from bit-flips in messages, but multiple messages are never sent along the same channel simultaneously. In contrast, the main bottleneck in imaging-based spatial genomics experiments arises from spatially overlapping barcodes and suffers less critically from missing signals, especially with amplification chemistry. The key insight enabling this advance is that error-correcting codes can not only correct transmission errors in telecommunications messages, but also reduce ambiguity arising from spatially overlapping messages in imaging-based spatial genomics data. The finding that spatially overlapping barcodes can be distinguished when they are sufficiently far apart in coding space allows for a new kind of code-based super-resolution microscopy that programs ON and OFF events rather than relying on stochastic switching.

0.3 Methods

Fitting dots with ADCG

Dense seqFISH+ images have overlapping dots, which are difficult to distinguish using Laplacian of Gaussians peak finding, the method used in prior works [2, 7, 9]. To better distinguish these dots, we adapted an algorithm that excelled in a benchmarking study for processing dense 2D single-molecule localization microscopy (SMLM) images, Alternating Descent Conditional Gradient (ADCG), to process seqFISH+ image stacks [33, 34]. ADCG fits models of images as linear combinations of Gaussian point spread functions (PSFs) emitted by fluorescent molecules,

$$E_{xy} = \sum_{k=1}^{n} b_k e^{-\frac{(x-x_k)^2+(y-y_k)^2}{2\sigma_k^2}} \delta(|x-x_k| < 3\sigma_{\max})\delta(|y-y_k| < 3\sigma_{\max}), \quad (1)$$

where E_{xy} denotes the model's expected intensity of the pixel in the xth column and yth row in the image; x_k, y_k, b_k, and σ_k are the x-coordinate, y-coordinate, brightness, and width parameters of the kth PSF in the model; δ represents an indicator function that saves computation time by not calculating the PSF tails; and σ_{min} and σ_{max} bound allowed values of σ_k. We include the width as a fit parameter because FISH dots may vary in size due to their distance from the imaging plane and because they are generated by multiple dye molecules attached to an RNA strand of

finite size and varying conformation. The fitting procedure uses a least squares loss function

$$\text{Loss} = \sum_{xy} (O_{xy} - E_{xy})^2, \tag{2}$$

where O_{xy} denotes the observed intensity of the xyth voxel in the image.

ADCG iteratively fits models by adding one new PSF at a time. The first step of each iteration is to approximate the location of the next PSF to add to the model by convolving the image-model residuals with a 2-D Gaussian of width σ_{min}, then finding the local maxima of the convolved image. Each PSF's width is approximated by grid search, where linear least squares fits find associated brightness. Second, gradient descents refine the coordinate, then the width and brightness estimates. We do not differentiate the indicator functions when calculating gradients of the loss, but we retain them in our gradient function to save calculation time. Third, repeated gradient descents on the coordinates then brightness and width of all PSFs adjusts the parameters of all PSFs in the model until the loss converges or for a maximum number of iterations. Finally, PSFs with brightness below the minimum allowed are removed. PSFs are added to the model until no new PSFs with brightness larger than the minimum allowed are found, the improvement in the objective function is below a threshold, or a maximum allowed number of PSFs have been added to the model.

When the algorithm terminates because the brightness or objective change is below a threshold, it discards the changes from the final iteration to avoid overfitting. For computational efficiency, we break up images into overlapping tiles, run the fit on each tile, and then piece the results from each tile together for the full image, removing duplicate PSFs found in the overlapping regions. Our implementation of this algorithm is available at `https://github.com/CaiGroup/SeqFISH_ADCG.jl`.

Fiducial marker matching registration

Dots found in different symbol blocks of seqFISH experiments must be aligned to correct for drift over the course of the experiment and recognize which were read out from the same barcode. Previous studies [2, 3, 9, 35] have aligned hybridization images using phase cross-correlation [14] on independent images of either DAPI stains or fiducial markers acquired concurrently but at a different wavelength than the readout images. This approach introduces errors from optical aberration, differences

in light path alignment, and movement of the microscope objective between scans. We reduce these sources of error by aligning using diffraction-limited objects as fiducial markers in the readout images identified with a pattern matching algorithm [26]. We then register images in the same optical channel as each other using translations and images in different optical channels using affine transformations.

Fiducial markers do not move relative to the slide, so they appear in a constellation, i.e. a fixed pattern, that drifts with the field of view in the readout images (Supplementary Fig. D.5A-C). Our algorithm searches for the dots in a fiducial marker constellation found in a reference image of only the fiducial markers among all dots found in readout images. The constellation of fiducial markers is identified by matching the relative position vectors between pairs of fiducial markers. Each matching pair of separation vectors in the reference and readout images votes to classify the dots at their ends as matches. This is similar to another pattern matching algorithm that matches triangles, rather than vectors [36].

Algorithm 1 Matching fiducial markers algorithm definitions

Definitions:
$\mathcal{F}$: the set of reference fiducial markers
w : the width of the image
$\mathcal{D}$: the set of dots in a readout image
ϵ : allowed spatial distance matching error
u, v : a pair of reference fiducial markers
$\mathbf{u}$, $\mathbf{v}$: the position vectors of u and v, respectively
u_x, v_x : the x-components of $\mathbf{u}$ and $\mathbf{v}$, respectively. u and v are named such that $u_x < v_x$
u', v' : dots in the readout image that are candidate matches for a u and v, respectively.
$\mathbf{u}'$, $\mathbf{v}'$: the position vectors of u' and v', respectively
u_x, v_x : the x-components of $\mathbf{u}$ and $\mathbf{v}$, respectively
β_α : the brightness of dot α
ρ_{min} : the minimum allowed brightness ratio to match readout dot to a reference fiducial marker, of $\beta_{\alpha'}/\beta_\alpha$
ρ_{max} : the maximum allowed brightness ratio to match readout dot to a reference fiducial marker, $\beta_{\alpha'}/\beta_\alpha$
$\times$: cartesian product operator
$\setminus$: set subtraction operator
$\leftarrow$: assignment operator
$||\mathbf{a}||_2 : \sqrt{\sum_{i=1}^{n} a_i^2}$ ▷ the l_2-norm of vector $\mathbf{a}$
$|\cdot|$: The cardinality of a set

Algorithm 1 Fiducial marker matching algorithm

Input: reference fiducial markers $\mathcal{F}$, readout dots $\mathcal{D}$, alignment error threshold ϵ, brightness ratio thresholds ρ_{min} and ρ_{max}, vote tally needed to confirm a match, minimum matches
Output: confirmed matches $\mathcal{M}$

$\mathcal{P} \leftarrow \{(u, v) \in \mathcal{F} \times \mathcal{F} \mid u_x < v_x\}$ ▷ get list of pairs of fiducial markers
sort $\mathcal{P}$ in descending order of $||\mathbf{u} - \mathbf{v}||_2$
for $(u, v) \in \mathcal{P}$ **do**
Rule out candidate dots for u and v by their positions:
$\mathcal{U}' \leftarrow \left\{u' \middle| u' \in \mathcal{D}, u'_x + (v_x - u_x) \le w + \epsilon, \begin{cases} u'_y + (v_y - u_y) \le w + \epsilon, & \text{if } u_y \le v_y \\ u'_y - (u_y - v_y) \ge -\epsilon, & \text{if } u_y > v_y \end{cases}\right\}$
$\mathcal{V}' \leftarrow \left\{v' \middle| v' \in \mathcal{D}, v'_x - (v_x - u_x) \ge -\epsilon, \begin{cases} v'_y - (v_y - u_y) \ge -\epsilon, & \text{if } u_y \le v_y \\ v'_y + (u_y - v_y) \le w + \epsilon, & \text{if } u_y > v_y \end{cases}\right\}$
Rule out candidate dots for u and v by their brightness:
$\mathcal{U}' \leftarrow \left\{u' \middle| u' \in \mathcal{U}'; \rho_{min} \le \frac{\beta_{u'}}{\beta_u} \le \rho_{max}\right\}$
$\mathcal{V}' \leftarrow \left\{v' \middle| v' \in \mathcal{V}'; \rho_{min} \le \frac{\beta_{v'}}{\beta_v} \le \rho_{max}\right\}$
Sort $\mathcal{U}'$ and $\mathcal{V}'$ in ascending order of u'_x and v'_x, respectively

for $u' \in \mathcal{U}'$ **do**
for $v' \in \mathcal{V}'$ **do**
if $(v'_x - u'_x) - (v_x - u_x) < -\epsilon$ **then**
$\mathcal{V}' \leftarrow \mathcal{V}' \setminus v'$
Break and go to next v'
else if $(v'_x - u'_x) - (v_x - u_x) > \epsilon$ **then**
$\mathcal{U}' \leftarrow \mathcal{U}' \setminus u'$
Break and go to next u'
else if $||(\mathbf{v}' - \mathbf{u}') - (\mathbf{v} - \mathbf{u})||_2 < \epsilon$ **then** ▷ We have a potential match
Search for matches to other reference fiducial markers at their known relative positions to candidate matches:
$\mathcal{M} \leftarrow \{\}$ ▷ Initialize set of confirmed matches
$\mathcal{Q} \leftarrow \{(u, u'), (v, v')\}$ ▷ Initialize queue of candidate matches
$\mathcal{T} \leftarrow \{(u, u') \Rightarrow 1, (v, v') \Rightarrow 1\}$ ▷ Initialize match tally dictionary
while $\mathcal{Q} \neq \emptyset$ **do**
$(a, a') \leftarrow \text{pop}(\mathcal{Q})$
for $b \in \mathcal{F}$ **do**
Check if there is a dot at the same relative position to a' as b is to a:
if $\exists b' \in D \left| ||(\mathbf{b}' - \mathbf{a}') - (\mathbf{b} - \mathbf{a}))||_2 \le \epsilon \right.$ **then** ▷ b' is a candidate match to b
$\mathcal{T}[(a, a')] += 1$ ▷ Increment the match tallies
if $(b, b') \notin \mathcal{T}$ **then** ▷ Add (b, b') to $\mathcal{Q}$ and $\mathcal{T}$
$\mathcal{Q} \leftarrow \mathcal{Q} \cup \{((b, b')\}$
$\mathcal{T} \leftarrow \mathcal{T} \cup \{(b, b') \Rightarrow 0\}$
end if
if $\mathcal{T}[(a, a')]$ == vote tally needed to confirm a match **then**
$\mathcal{M} \leftarrow \mathcal{M} \cup \{(a, a')\}$ ▷ Add confirmed match
Break out of for loop, go to next candidate match in $\mathcal{Q}$
end if
end if
end for
end while
if $|\mathcal{M}| \ge$minimum matches **then**
return $\mathcal{M}$
end if
end if
end for
end for
end for
return $\emptyset$

The most challenging task is to find the first matching pair of separation vectors. We begin by assembling a list of the most distant pairs of fiducial markers from the reference image, excluding any fiducial marker that is within the allowed matching error distance, ϵ, of another fiducial marker. We denote the two fiducial markers in a pair as u and v such that the x-component, u_x, of u's position vector, $\mathbf{u}$, is less than the x-component, v_x, of v's position vector, $\mathbf{v}$. The list is sorted in descending order of $||\mathbf{v}-\mathbf{u}||_2$. Initially searching for the most distant pairs of fiducial markers lets us narrow the region of the readout image where we must search for u and v to around the corners.

For each (u, v) pair, we rule out dots as candidates for u and v. We rule out u' and v' candidates at locations in the image where their partner dot would be off the image. We rule out more dots by setting bounds on the brightness ratio between fiducial markers in the reference image and readout image. When drifts are small, we can also restrict the search to dots within a maximum allowed shift between images. To find an initial match, we sort the lists of dots in the readout image that may match to u and v by their x-coordinate from lowest to highest. We denote individual candidate dots u' and v'. We first compare the x-components of the position vectors separating each u' and v', $\mathbf{v}'-\mathbf{u}'$, with the position vector separating reference dots, $\mathbf{v}-\mathbf{u}$. We start with the first u' and v' candidates in the lists, then proceed through the following logical flow: if $(v'_x - u'_x) - (v_x - u_x) < -\epsilon$, then v' cannot match v, so we discard it and start again at the beginning of the revised lists. Likewise, if $(v'_x - u'_x) - (v_x - u_x) > \epsilon$, then u' cannot match u, so we discard it and start at the beginning of the revised lists. If $||(\mathbf{v}'-\mathbf{u}')-(\mathbf{v}-\mathbf{u})||_2 > \epsilon$, then we check the next v', in the list. If $||(\mathbf{v}'-\mathbf{u}')-(\mathbf{v}-\mathbf{u})||_2 < \epsilon$, the relative position vectors match and we proceed to search for the rest of the fiducial markers relative to it. When there is fewer than one candidate dot for every error window of 2ϵ, this algorithm will find matches in $O(n)$ time.

To begin the search for the rest of the fiducial markers in the readout image, we initialize a queue containing the first two candidate fiducial marker matches, (u, u') and (v, v'). We then begin taking fiducial marker matched dots from the queue and searching for other fiducial markers from the reference pattern at their known relative position from the already matched dot in the readout image with a KDTree. For example, if we already have matched the dot a' to the fiducial marker a, we search for another dot b' that matches another fiducial marker b such that $||(\mathbf{b}'-\mathbf{a}')-(\mathbf{b}-\mathbf{a}))||_2 < \epsilon$. If such a b' is found, the candidate-reference pair, (b, b'),

is added to the candidate match queue. Each (b, b') candidate match that is found relative to an (a, a') candidate match is a matching edge that votes for the (a, a') and (b, b') matches. Once an (a, a') candidate matching pair has received ten votes from matching separation vectors, the match is confirmed, and then the algorithm proceeds to search for matches with the next (a, a') candidate-reference pair in the queue. If no (b, b') candidate matches are found in the first ten searches from a given (a, a') pair of candidate matches, the (a, a') candidate match is discarded. If fewer than five matches have been confirmed with ten matching edge votes after exhaustively searching from initial (u, u') and (v, v') candidate matches, all matches are discarded and the algorithm searches for new initial matches.

Search parameters may either be set manually by the user or the algorithm may first conduct a grid search for parameters to match the fiducial markers in two reference images of only the fiducial markers acquired at the beginning and end of the experiment. In the latter case, the algorithm chooses the strictest value for the search radius in which it finds 90% of the matches that if found with the least strict parameters, the minimum brightness considered for a match to an initial dot (ρ_{min}) to be 50% less than the ratio of the greatest reduction in brightness from the initial to the final reference image, and the maximum allowed brightness ratio of matches to initial fiducial markers (ρ_{max}) to be the maximum observed brightness ratio of initial to final reference dot. Our implementation of this algorithm is available at `https://github.com/CaiGroup/seqfish_fm_match`.

SeqFISH representation of error-correcting codes

Reed-Solomon codes are a family of error-correcting codes with an elegant mathematical construction that are maximum distance separable: the minimum Hamming distance between any two codewords (MHD) in the error-correcting code is the maximum possible for a linear error-correcting code given the number of parity check symbols it contains. To represent Reed-Solomon codes using seqFISH, we use only codewords with a fixed weight (number of non-zero symbols) and probe only non-zero symbols in each codeword. Zero-valued symbols are represented by symbol blocks without a dot. We encoded our experiment using weight-four codewords from the Reed-Solomon code with 10 symbols ($n = 10$), including seven information symbols ($k = 7$) and three parity check symbols ($n - k = 3$), from an alphabet of size 11 ($q = 11$). Our simulations include codewords from other Reed-Solomon codes using codewords of weight four, five, and six (Fig. 0.1E and Supplementary Fig. D.2). Our supplementary simulations include a Reed-Solomon

code whose symbols are elements of the extended finite field of eight elements. For this and other Reed-Solomon codes defined over extended finite fields, the symbol value represented by a dot is the primitive element of the finite field exponentiated by the dot's pseudo-color number minus one.

Generating codebooks and parity check matrices

Reed-Solomon codes are defined using the abstract algebra structures of polynomial rings over finite fields. An equivalent representation to the generator polynomial is a generator matrix with elements from the same finite field as the generator polynomial coefficients [1]. To compute the codewords with the desired number of non-zero symbols, we find the systematic form (row reduced echelon form) of the generator matrix, which computes codewords that have parity check symbols appended to the information symbols in the message.

$$c = mG, \text{ where} \tag{3}$$

$$G = [I|P], \tag{4}$$

m is a message vector containing the information symbols encoding a gene, c is the codeword that consists of m with parity check symbols appended. G is the generator matrix in systematic form, which consists of an identity matrix concatenated with columns representing the parity check equations. Using the systematic form of the generator matrix allows us to consider only messages with the desired number of non-zero information symbols or fewer, compute the parity check symbols for the message, then discard codewords that have more than the desired number of non-zero symbols. We then check that we found the same number of codewords of the desired weight as predicted by the theoretical weight enumeration formula [37]. We provide notebooks implementing Reed-Solomon codeword generation on both CPUs and GPUs for various kinds of Reed-Solomon codes `https://github.com/CaiGroup/UntanglingBarcodes/tree/main/codebook_generation/get_RS_codebooks`.

Assigning genes to codewords

We optimize the assignment of genes to codewords to minimize the maximum sum FPKM expression of all genes probed in any single hybridization image. Alignment across optical channels is worse than alignment within a single optical channel, so after assigning genes to codewords, we assigned symbol block and pseudo-color values to hybridization and optical channel images to maximize the number of barcodes that are probed in the same optical channel either three out of four or four

out of four times. Notebooks are available at `https://github.com/CaiGroup/UntanglingBarcodes/tree/main/codebook_generation/assign_genes_to_codewords`.

Identifying candidate barcodes with syndrome decoding

To identify candidate barcodes, we use dynamic programming to find dots that align in different symbol blocks whose pseudo-color values satisfy the error-correcting code's parity check equations. First, we illustrate the algorithm using the error-correcting code from the original seqFISH+ experiment, a q-ary parity check code with four symbol blocks and 20 pseudo-colors [10]. All combinations of pseudo-colors are allowed in the first three symbol blocks, but the final pseudo-color is determined by the parity check equation:

$$c_4 = c_1 + c_2 - c_3, \tag{5}$$

where, c_i denotes the value of the ith symbol in an error-correcting code. When describing a seqFISH experiment that transmits information using this code, we say that c_i denotes the pseudo-color number of an aligning dot in symbol block i. Addition and subtraction operations are modulo 20, the number of pseudo-colors. With this code, any three aligning dots in each of the first three symbol blocks with any combination of pseudo-color values may represent a barcode. In the terminology of error-correcting codes, these dots represent information symbols that alone are sufficient to represent all the information encoded in a message. Dots in the final symbol block represent parity check symbols, which add robustness to the message. The parity check equation defines the syndrome,

$$s = c_1 + c_2 - c_3 - c_4, \tag{6}$$

such that the syndrome, s, will be 0 if and only if the sequence of symbols represented by the aligning dots' pseudo-colors is a codeword. In general, linear error-correcting codes are defined by a parity check matrix, H, such that

$$Hc = 0 \qquad\qquad \forall c \in \mathbb{C}, \tag{7}$$

where $\mathbb{C}$ is the set of codewords in the code. Codewords, c, are sequences of symbols drawn from an alphabet of size q. Arithmetic operations are modulo-q. The parity check code used in reference [10] uses a parity check matrix,

$$H = \begin{bmatrix} 1 & 1 & -1 & -1 \end{bmatrix}. \tag{8}$$

The parity check equation defines the syndrome,

$$s = Hc, \tag{9}$$

such that all elements of s will be 0 if and only if the sequence of symbols represented by the aligning dots is a codeword. The linearity of the syndrome computation allows all candidate barcodes to be found with dynamic programming, a technique that is also known by the more descriptive name, recursive optimization. This procedure involves breaking the matrix multiplication summing operations into steps and saving intermediate sums to reduce the number of arithmetic operations that must be performed.

We take a divide and conquer approach to efficiently compute syndromes and identify candidate barcodes through independent searches in overlapping square tile regions covering the image. For a collection of dots to qualify as a valid candidate barcode, all its dots must align to within the search radius of one another. To guarantee that all valid candidate barcodes are found, a circle with diameter equal to the search radius must be guaranteed to be contained entirely in at least one tile of the tile cover of the image. We achieve this guarantee by defining a tile cover of the image composed of square tiles of width twice the search radius centered at each point in a square grid with spacing of one search radius. This leads to some individual barcodes being found in multiple tiles, but reduces overall computational cost by eliminating chains of subsequently aligning dots that extend long distances and reducing the number of intermediate sums that need to be held in memory concurrently. After finding candidate barcodes in each tile, we remove duplicate candidate barcodes found in multiple tiles.

We now use a recursive framework to describe the dynamic programming algorithm performed on dots in each tile region for the simpler case where barcodes are probed in every symbol block (Supplementary Fig. D.1A). In the initialization step, we assign a partial syndrome sum array to each dot in the first symbol block that contains the pseudo-color number (symbol value) of that dot as its lone element. We then proceed to construct partial syndrome sum arrays for dots in each subsequent symbol block. To construct the partial syndrome sum array for each dot, we first use a KDTree search to find each dot's neighboring dots in the previous symbol block that align to within a search radius. The partial syndrome sum arrays assigned to the aligning dots are concatenated. Finally, the searching dot's pseudo-color is added element-wise (or subtracted) from the concatenated array, resulting in that

dot's partial syndrome sum array. Syndromes with value 0 in the final symbol block indicate that a chain of subsequently aligning dots has pseudo-colors that are consistent with a valid barcode. We can trace the dots that contributed to zero-valued syndromes. Traced collections of dots for which all dots are within the search radius of each other are valid candidate barcodes. Traced collections of dots for which any two dots are farther apart from one another than the search radius are discarded.

Algorithm 2 Syndrome decoding for single parity check codes: Definitions

Definitions
D_i : The array of all dots in ith symbol block
$D_i[j]$: The jth dot in the ith symbol block
n : the number of symbol blocks
$p(D_i[j]))$: the pseudo-color of dot $D_i[j]$)
d : a dot
$\mathbf{x_d}$: the position vector of dot d
S_i : An array holding arrays of partial syndrome sums, where the jth element holds the array of partial syndrome sums for the jth element of D_i.
r : the alignment search radius
H : the parity check matrix. Parity check matrices for single parity check codes contain only one row.
$+_q$: element-wise modulo-q addition
q : the size of the symbol alphabet used in the error-correcting code
$+$: Real number scalar addition
$\big|$: such that
$\forall$: for all
$\leftarrow$: assignment operator
$\cup$: union of sets
$==$: logical equals
$||\cdot||_2$: the l_2-norm of a vector
$\{\cdot\}$: a set
KDTreeBallSearch$(D_i, \mathbf{x_d}, r)$: Do a KDTree search for dots in D_i that are within r of $\mathbf{x_d}$ and return a sorted list of indices of the resulting dots in D_i.

Algorithm 2 Syndrome decoding for single parity check codes by recursive sums

Input: $D_1, D_2, ..., D_n, r, H$
Output: $\mathcal{B}$, the list of candidate barcodes

Initialization Step: allocate partial sum arrays for dots in the first symbol block
for j in 1 : length(D_1) **do**
  $S_1[j] \leftarrow [p(D_1[j])H[1]]$
end for
Recursive steps: compute the partial syndrome sum arrays for dots in the remaining symbol blocks by adding each dot's contribution to the syndrome to the partial sums of aligning dots in the previous symbol block
for symbol block i in 2 : n **do**
  for j in 1 : length(D_i) **do**
    $S_i[j] \leftarrow []$
    for dots k found in KDTreeBallSearch($D_{i-1}, \mathbf{x_d}, r$) **do**
      $S_i[j] \leftarrow$ Concatenate $\left(S_i[j], S_{i-1}[k]\right)$
    end for
    $S_i[j] \leftarrow S_i[j] +_q p(D_i[j])H[i]$
  end for
end for
Trace the aligning dots with pseudo-colors summing to 0, thus comprising candidate barcodes.
$\mathcal{B} \leftarrow \emptyset$
for j in 1 : length(S_n) **do**
  if $S_n[j] == 0$ **then**
    $\mathcal{B} \leftarrow \mathcal{B} \cup \{\text{TraceDots}(n, j)\}$
  end if
end for
Keep Barcodes for which all pairs of component dots are within r of each other and discard the rest
$\mathcal{B} \leftarrow \left\{ B \,\middle|\, ||\mathbf{x_{d^*}} - \mathbf{x_{d'}}||_2 \leq r, \forall\, d^*, d' \in B, B \in \mathcal{B} \right\}$
return $\mathcal{B}$

function TraceDots(i, j) ▷ Traces the dots that contributed to zero-valued syndrome
  if $i == 1$ **then**
    return $\{D_i[j]\}$
  else
    counter $\leftarrow 0$
    for dots of index k found in KDTreeBallSearch($D_{i-1}, \mathbf{x_d}, r$) **do**
      if counter + length($S_{i-1}[k]$) > j **then**
        return TraceDots$\left(i-1, j-\text{counter}\right) \cup \left\{D_i[j]\right\}$
      end if
      counter $\leftarrow$ counter + length($S_{i-1}[k]$)
    end for
  end if
end function

Now we describe the procedure for the more general and complex case where barcodes are not probed in every symbol block (Supplementary Fig. D.1B). In this case, we must align dots from each symbol block with dots from multiple previous symbol blocks and track the number of dots contributing to each partial sum. To do this, partial sums for each dot are stored in one of w arrays, where w is the weight of the codewords used in the experiment, i.e., the number of dots readout from each barcode. Elements in the ith of the w arrays hold partial sums of pseudo-colors from i subsequently aligning dots in different earlier symbol blocks. The first array of partial sums for each dot contains only that dot's contribution to a syndrome. The ith partial sums array for each dot, where $i > 1$, is found by concatenating the $(i-1)$th nested syndrome partial sum array for dots in

previous symbol blocks within the search radius, then adding the syndrome contribution of the searching dot element-wise. The wth array holds the syndromes for paths of aligning dots. Entries in the wth array with value 0 represent candidate barcodes and we can trace the dots that contributed to it. When allowing incomplete barcodes with 1 missing dot, all paths of length $w - 1$ are traced and searched in a BKTree of codewords to check whether they are Hamming distance 1 from a valid codeword. Our implementation of this algorithm is available at `https://github.com/CaiGroup/SeqFISHSyndromeDecoding.jl`.

Resolving conflicting decoding solutions with integer programming

In dense experiments, syndrome decoding will yield conflicting barcode candidates that include at least one of the same dots (Fig. 0.1C). The densest regions may have large networks of conflicting barcode candidates. To find the non-conflicting barcodes that best explain the observed dots (Fig. 0.1D), we define a cost function for each feasible solution

$$C(\mathcal{B}) = \sum_{B \in \mathcal{B}} \left(a_1(\mathrm{var}(x_B) + \mathrm{var}(y_B)) + a_2 \mathrm{var}(\log(b_B)) + a_3 \delta_B \right) + N_{nd}, \tag{10}$$

where $\mathcal{B}$ is the set of all barcode candidates in the network; B is a single barcode candidate; x_B, y_B, and b_B are the sets of x-coordinates, y-coordinates, and brightness of each dot in the candidate barcode B; δ_B is an indicator function that is equal to 0 if the barcode B is complete or 1 if the barcode is missing a dot; and N_{nd} is the number of dots in the network that are not in a barcode candidate chosen by the solution. var is the sample variance of its input set defined as

$$\mathrm{var}(\alpha) = \frac{\sum_{j=1}^{n} (\alpha_j - \bar{\alpha})^2}{n - 1}. \tag{11}$$

The logarithm on the set of brightnesses of the dots in a barcode is applied element-wise. a_1, the lateral position variance penalty; a_2, the brightness variance penalty; and a_3, the missing dot penalty, are user set parameters. We use an off-the-shelf integer programming solver to find the lowest cost, non-conflicting set of barcode candidates [38] (Gurobi). We solve the optimization problem as written in standard form:

$$\text{Minimize } \mathbf{c} \cdot \mathbf{b} \tag{12}$$

$$\text{Subject to } A\mathbf{b} \preceq 1, \tag{13}$$

where **b** is a vector of indicator variables for each candidate barcode, **c** is the vector of corresponding costs for each candidate barcode where the cost of the barcode of index i is calculated as

$$c_i = a_1(\text{var}(x_i) + \text{var}(y_i)) + a_2\text{var}(\log(b_i)) + a_3\delta_i - n_i, \tag{14}$$

where n_i is the number of dots in the ith barcode, $\preceq$ is the element-wise less than or equal to comparator, and A is the adjacency matrix for the candidate barcodes. x_i, y_i, b_i, δ_i, a_1, a_2, and a_3 are defined as in equation 10. Barcodes that are composed of at least one of the same dots are neighbors in A. Barcode candidates with non-negative costs according to equation 14 are discarded. We implemented a function to perform this optimization in our package located at `https://github.com/CaiGroup/SeqFISHSyndromeDecoding.jl`.

To apply this to experimental data, we set the search radius to four pixels in our 3T3 cell experiment spots-first workflows and three pixels in the mouse cortex spots-first workflow. We discard candidate barcode conflict networks that have more than 2500 candidate barcodes and have a ratio of candidate barcodes to bounding box area greater than 10 for the seqFISH+ reanalysis or more than 700 candidate barcodes and a ratio of candidate barcodes to bounding box area greater than 700 for the cross-channel Reed-Solomon encoded 3T3 cell experiment and mouse cortex experiments. We tried a range of reasonable values for the cost parameters to gauge the trade-off between sensitivity and false discovery rate (FDR) and plotted ROC-like sensitivity-FDR curves (Fig. 0.2A-C).

SeqFISH+ 2019 spots-first image processing workflow

We use the snakemake workflow manager [39] to define a reproducible workflow for processing seqFISH images. The seqFISH+ 2019 experimental data is divided into three independently coded optical channels each encoding 3,333 or 3,334 genes in 80 hybridization images divided into four symbol blocks of 20 pseudo-colors. This experimental design avoided error from cross-channel image registration and reduced ambiguity by partitioning barcodes that may have overlapped into independent subdivisions. The workflow first fits fluorescent beads used as fiducial markers with ADCG in all images, then finds the translations of readout images relative to a bead-only reference image of the same optical channel by matching fiducial marker patterns. Before fitting dimmer FISH dots, the workflow removes the fiducial markers and autofluorescence from readout images by pixel-wise subtracting the aligned reference images from them. It then subtracts the scattering background intensity es-

timated by applying a median filter with a 5×5 pixel kernel and then using the rolling ball algorithm with a radius of 3.3 pixels [40] and masks out intensity from pixels between cells. The same hand-drawn masks used in the original study are used [10]. The workflow then finds FISH dots in the preprocessed images with ADCG, aligns their fit coordinates, splits them into groups aligning to each cell mask, and then uses syndrome decoding to infer which barcodes generated the PSFs in each cell. Only complete barcodes are allowed. The implementation of this workflow is available at `https://github.com/CaiGroup/UntanglingBarcodes/tree/main/real_data_processing_workflows/2019_seqfishplus_reprocessing`.

Cross-channel Reed-Solomon encoded seqFISH workflow

Our Reed-Solomon encoded experiment uses 100 image stacks acquired in three optical channels and 34 hybridization cycles to represent ten symbol blocks of ten pseudo-colors. 2D segmentations were drawn manually with Cellpose[41] from FISH images of cells hybridized with poly-T probes. First, beads used as fiducial markers were fit in the brightest z-slice (4th z-slice) where cells were masked out by segmentation masks. Beads in reference images containing only the beads and no readout probes were matched between optical channels and to beads found in images with readouts in the same optical channel. The optical channel with the most fiducial markers from the readout images matched to corresponding fiducial markers in its reference image was used as the reference optical channel for each position. Images in the reference optical channel were registered to their reference image by translation, and images in the other optical channels were registered by affine transformation to the reference optical channel image. Background subtraction for images in each optical channel was performed similarly to the 2019 seqFISH+ experiment processing workflow, but in 3D. After background subtraction, readout dots were fit using 2D ADCG in the brightest (4th) z-slice. For this experiment, the increased minimum distance of the code allows incomplete barcodes with only three dots (one missing dot) and the missing dot penalty (a_3 in equation 10 and 14) is set to 2. The implementation of this workflow is available at `https://github.com/CaiGroup/UntanglingBarcodes/tree/main/real_data_processing_workflows/Reed-Solomon_encoded_experiment_workflow`.

Mouse cortex processing workflow

The mouse experiment used the same genes and codebook as in the cross-channel Reed-Solomon encoded 3T3 cell experiment, however all images were acquired

in 561 optical channel using Cy3B fluorescent dye. Z-planes were acquired at intervals of 1 micron. Fiducial marker matching was done in the sixth z-slice to calculate translational offsets. Segmentations were computed on 1/8th downsampled image stacks of the cells with DAPI staining and Poly-T FISH staining using Cellpose-SAM [42]. 2D spots-first decoding was performed on each z-plane independently using a search radius of three pixels. The sensitivity-estimated FDR curve was calculated using the first nine fields of view. The remaining fields of view were decoded with lateral variance penalty of five and log brightness variance penalty of zero. The implementation of this workflow is available at `https://github.com/CaiGroup/UntanglingBarcodes/tree/main/real_data_processing_workflows/mouse_cortex`

Label transfer in mouse cortex

Cell types in seqFISH mouse cortex data were assigned using CONCORD to transfer labels from Allen Brain Atlas 10X single-cell sequencing data (Supplemental Fig. D.4B). Cells in the 10X dataset with fewer than 10 genes, with missing class labels, or that were members of a cell class with fewer than nine cells of that class were removed. Expression counts were normalized in the remaining cells. Cells in the seqFISH dataset with fewer than 100 genes or that were in the manually annotated PIA layer region were discarded. PIA layer cells are difficult to dissociate in single-cell sequencing protocols and are thus likely underrepresented in 10X datasets, leading to poor label transfer results for these cells. Since some cell types are far more represented in the 10X data than others, the training dataset was balanced by choosing a random subsample of 1000 cells for each cell type class, of which there were more than 1000 cells. We trained ten models, each with different balanced subsamples of the training dataset, using only the genes that were profiled in our seqFISH experiment. We used each CONCORD model to predict the cell types of each cell in the seqFISH dataset. We aggregate classes of neurons into two superclasses of GABAergic neurons and glutamatergic neurons. We then find cells that have consensus cell type predictions from all models.

Compressed sensing decoding

The spots-first decoding procedure described above cannot distinguish spatially overlapping dots generated from two or more transcripts in the same optical image. To address this, we develop an alternate decoder that uses a compressed sensing framework to fit barcodes in all optical images simultaneously. The drawbacks

compared to the spots-first approach are that candidate dots must be located on a predefined grid and that the computational complexity increases more quickly with increasing search radius, which limits its capacity to handle larger readout dot jitter or registration error between readout images. These limitations may be overcome with future work.

After performing image registration and preprocessing as described for the spots-first workflows, the next step is to identify candidate dot locations in each readout image. Preliminary candidate dot locations, $\mathcal{D}$, are found by convolving (i.e., blurring) readout images with a Gaussian kernel (the point spread function). Centers of pixels in the convolved (blurred) image with value greater than a threshold are the preliminary candidate dot coordinates,

$$\mathcal{D} = \left\{ (x_i, y_i) \middle| (\Psi * \mathbf{I})[y_i, x_i] \geq \beta_{\text{threshold,dot}}, x_i \in [1, 2, ..., h], y_i \in [1, 2, ..., w] \right\}, \tag{15}$$

where $\mathbf{I}$ is the image matrix of dimension $h \times w$,

$$\Psi = \begin{bmatrix} \psi_i(-3,3) & \psi_i(-2,3) & \psi_i(-1,3) & \psi_i(0,3) & \psi_i(1,3) & \psi_i(2,3) & \psi_i(3,3) \\ \psi_i(-3,2) & \psi_i(-2,2) & \psi_i(-1,2) & \psi_i(0,2) & \psi_i(1,2) & \psi_i(2,2) & \psi_i(3,2) \\ \psi_i(-3,1) & \psi_i(-2,1) & \psi_i(-1,1) & \psi_i(0,1) & \psi_i(1,1) & \psi_i(2,1) & \psi_i(3,1) \\ \psi_i(-3,0) & \psi_i(-2,0) & \psi_i(-1,0) & \psi_i(0,0) & \psi_i(1,0) & \psi_i(2,0) & \psi_i(3,0) \\ \psi_i(-3,-1) & \psi_i(-2,-1) & \psi_i(-1,-1) & \psi_i(0,-1) & \psi_i(1,-1) & \psi_i(2,-1) & \psi_i(3,-1) \\ \psi_i(-3,-2) & \psi_i(-2,-2) & \psi_i(-1,-2) & \psi_i(0,-2) & \psi_i(1,-2) & \psi_i(2,-2) & \psi_i(3,-2) \\ \psi_i(-3,-3) & \psi_i(-2,-3) & \psi_i(-1,-3) & \psi_i(0,-3) & \psi_i(1,-3) & \psi_i(2,-3) & \psi_i(3,-3) \end{bmatrix} \tag{16}$$

is the Gaussian kernel, $*$ is the convolution operator, $(\Psi * \mathbf{I})[y_i, x_i]$ is the value of the pixel in the y_ith row and x_ith column of the convolved image, and $\beta_{\text{threshold,dot}}$ is a user defined threshold. We use a 2D Gaussian point spread function defined as

$$\psi_i(x, y) = \begin{cases} e^{-\frac{(x-x_i)^2+(y-y_i)^2}{2\sigma^2}}, & \text{if } \sqrt{(x-x_i)^2 + (y-y_i)^2} \leq 3\sigma \\ 0, & \text{otherwise} \end{cases} \tag{17}$$

where x_i are y_i are the x-coordinate and y-coordinate of the center of the point spread function and σ is the width parameter.

Candidate locations identified by thresholding the convoluted image are further thinned using a LASSO regression model similar to a model that has previously been used for single-molecule localization microscopy [43]. Column vectors, $\mathbf{d_i}$,

of the sensing matrix, A, are flattened representations of the signal expected to be readout from a molecule centered at coordinates $(x_i, y_i) \in \mathcal{D}$. We can write the optimization problem as

$$\hat{\boldsymbol{\beta}} = \text{argmin}\ \frac{1}{2wh}\left\|\mathbf{y} - A\boldsymbol{\beta} - \beta_0\right\|_2^2 + \lambda\left\|\boldsymbol{\beta}\right\|_1, \text{ where} \tag{18}$$

$$A = \begin{bmatrix} | & | & & | & & | \\ \mathbf{d_1} & \mathbf{d_2} & \dots & \mathbf{d_i} & \dots & \mathbf{d_{n_{\text{dots}}}} \\ | & | & & | & & | \end{bmatrix}, \tag{19}$$

$$\mathbf{d_i} = \big[\psi_i(1,1), \psi_i(2,1), \ \dots, \psi_i(h,1), \psi_i(1,2), \psi_i(2,2), \ \dots, \psi_i(h,2), \ \dots, \psi_i(1,w), \psi_i(2,w), \ \dots, \psi_i(h,w)\big] \tag{20}$$

$$\text{with the constraint that } \boldsymbol{\beta} \succeq 0, \tag{21}$$

where $\mathbf{y}$ is the vector of observed pixel values in the image, $\boldsymbol{\beta}$ is the coefficient vector, β_0 is the (scalar) intercept, λ is the regularization parameter, n_{dots} is the number of candidate dots, h is the number of rows, w is the number of columns of pixels in the region of interest, and $\succeq$ is the element-wise greater than or equal to symbol. To reduce computational cost, rows of A with all zeros and the corresponding elements of $\mathbf{y}$ and $\boldsymbol{\beta}$ are removed. Only candidate dots with $\beta_i \geq \beta_{\text{threshold,dot}}$ at the smallest value of λ for which β_0 is non-negative are kept. Remaining candidate dot locations are then registered using translations calculated from matched fiducial markers, and candidate barcodes are identified using the syndrome decoding algorithm as described for the spots-first workflow with a user-supplied search radius of r. This yields a list of candidate barcodes

$$C = \left\{\mathcal{B}_1, \mathcal{B}_2, ..., \mathcal{B}_j, ...\mathcal{B}_{n_{\text{barcodes}}}\right\}, \text{ where} \tag{22}$$

$$\mathcal{B}_j = \left\{d_{j1}, d_{j2}, ..., d_{jk}, ..., d_{j\omega}\right\}, \tag{23}$$

where $\mathcal{B}_j$ denotes the jth candidate barcode, d_{jk} denotes the kth candidate dot in the jth candidate barcode, and ω is the weight of the codewords, i.e. the number of dots in a barcode.

Next, for each cell, a LASSO model is constructed to select candidate barcodes. In this model, the $\mathbf{y}$ vector contains the pixel values of all readout images of the cell and columns of the sensing matrix, A, indicate the expected illumination patterns for each of the n_{barcodes} candidate barcodes in all images

$$A_b = \begin{bmatrix} | & | & & | & & | \\ \mathbf{b}_1 & \mathbf{b}_2 & \dots & \mathbf{b_j} & \dots & \mathbf{b}_{n_{\text{barcodes}}} \\ | & | & & | & & | \end{bmatrix}. \tag{24}$$

The column vectors, $\mathbf{b_j}$, representing the signal expected to be emitted by the jth candidate barcode in all images, are constructed by concatenating a sequence of vectors representing the expected signal, $\mathbf{s_i}$, from the candidate barcode in each image,

$$\mathbf{b_j} = \text{Concatenate}(\mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_l, ..., \mathbf{s}_{n_{images}}). \tag{25}$$

The expected signal for images is

$$\mathbf{s_l} = \begin{cases} \mathbf{d}_l \big| d_l \in \mathcal{B}_j, d_l \in \mathcal{I}_l, & \text{if } \exists d_l \in \mathcal{B}_j \big| d_l \in \mathcal{I}_l \\ \mathbf{0}, & \text{otherwise} \end{cases}, \tag{26}$$

where $\mathbf{d_l}$ is the flattened dot signal vector defined in equation 20 for the component dot, d_l, in the barcode, $\mathcal{B}_j$, and in $\mathcal{I}_l$, the set of dots in the lth image, $\mathbf{0}$ is a vector of zeros of length wh, the number of pixels in each image. The LASSO optimization problem for selecting barcodes is

$$\hat{\boldsymbol{\beta}} = \text{argmin} \frac{1}{2wh} \left\| \mathbf{y} - A_b \boldsymbol{\beta} - \beta_0 \right\|_2^2 + \lambda \left\| \boldsymbol{\rho} \odot \boldsymbol{\beta} \right\|_1 \tag{27}$$

$$\text{with the constraint that } \boldsymbol{\beta} \succeq 0. \tag{28}$$

The $\odot$ operator represents element-wise multiplication, and $\succeq$ is the element-wise greater than or equal to symbol. The LASSO penalty for each barcode is modulated by the variance of the positions of the candidate dots comprising each barcode. $\boldsymbol{\rho}$ is a vector of penalties for each barcode and its jth element is calculated as

$$\rho_j = p(\text{var}(x_j) + \text{var}(y_j)) + 1, \tag{29}$$

where var is defined in equation 11, and x_j and y_j are the registered x and y-coordinates of the dots in the jth candidate barcode respectively, and p is a parameter controlling the severity of the penalty. Again, to reduce computational cost, we remove rows of A that are all zeros and corresponding elements of $\mathbf{y}$ and $\boldsymbol{\beta}$. The glmnet package returns a LASSO path with many values of the LASSO penalty, λ, starting with the largest value that returns a non-null model, λ_0, and ending with $\lambda = \lambda_0/1000$ [44]. The LASSO path includes a B matrix where each ith column vector is the fit value of $\hat{\boldsymbol{\beta}}$ for the ith value of λ.

Compressed sensing relies on the sensing matrix having low mutual coherence (i.e., distinct columns) to guarantee unique and accurate solutions [45, 46]. However, matrices A_b as defined in equation 24 will have many highly coherent columns that

represent slight variations in candidate dot locations to explain the signal emitted from an individual molecule. Many of these overlapping candidate barcodes for transcripts of the same gene will be selected at lower values of λ, diluting their coefficient values. Thus, at any particular value of λ, candidate barcode j that have had a fit coefficient value greater than the threshold, $\hat{\beta}_j > \beta_{\text{threshold,barcode}}$, at any $\lambda \geq \lambda_m$ are considered selected at λ_m. We compensate for this by removing duplicate overlapping barcodes for the same gene selected at each value of λ. To identify duplicate barcodes, we first construct networks of selected barcodes that encode each kind of RNA molecule. Barcodes in a network are neighbors if they are centered within the decoding search radius, r, of each other. Of all barcodes in a connected network, we only keep the barcode that is included in the model with the largest fit coefficient value, β_j, at the largest λ where any of the barcodes in the network are included in the model. The other selected barcodes in the network of barcodes encoding the same gene are discarded. In future implementations, it may be desirable to remove duplicate overlapping barcodes either using l_0-regulated regression as demonstrated in supplementary simulations (Supplementary Fig. D.6B, Appendix C.2) or to group overlapping barcodes in a grouped LASSO model to learn the spot shape of non-diffraction-limited target molecules [47–49].

The sensitivity-estimated FDR curve shown in Fig. 0.2A is calculated by first running the above algorithm with every combination of candidate $\beta_{\text{threshold,dot}}$, r, ρ, and $\beta_{\text{threshold,barcode}}$ requested by the user. Then the best estimated FDR curve for each cell is found by keeping the results only for which there are no other parameter sets with a lower estimated FDR and a higher sensitivity. The overall estimated FDR-sensitivity curve is found by averaging the decoding results from each cell when decoded with the value of the parameters that yielded the closest estimated FDR to the target value. The implementation of this workflow is available at `https://github.com/CaiGroup/UntanglingBarcodes/tree/main/real_data_processing_workflows/compressed_sensing`

Simulations to determine density robustness of codes

Imageless simulations of decoding performance with various error-correcting codes included seqFISH q-ary parity check codes, the Hamming code used in the original MERFISH experiment [3], the non-linear sequence space cover code used in later MERFISH experiments [12], Reed-Solomon Codes, and a new non-linear code (Fig. 0.1E, Supplementary Fig. D.2, Appendix A). The units of all distances in this section are pixels, which in our experiments are approximately 100 nm in width. Simulation

workflows first generate the ground truth (the coordinates and identity of each simulated transcript) for each replicate of each condition for each error-correcting code. x and y-coordinates of each transcript were drawn from a uniform distribution from 0 to 1. The codeword of each object was randomly selected from the given codebook with equal probability for each codeword. Then, the workflow generated simulated observed readout dot locations from each simulated transcript within the appropriate symbol block and of the appropriate pseudo-color. The localization error of each simulated readout dot from the simulated true location of each simulated transcript was drawn from a 2D normal distribution with $\sigma = 0.5$. Each readout dot had a 1/20 probability of being dropped in the simulation data. Dots of each pseudo-color of each symbol block have a 1/50 probability of including a non-specific dot with x and y-coordinates drawn from a uniform distribution from 0 to 1. Simulated dots were fed into the SeqFISHSyndromeDecoding.jl package with a search radius of 10, a lateral variance penalty of 10, and no drops allowed. The simulation workflow is available here `https://github.com/CaiGroup/UntanglingBarcodes/tree/main/simulations/Code_overlap_robustness`.

Performance metrics for simulations

To evaluate the decoding performance in simulations, we calculate the sensitivity and FDR as

$$\text{Sensitivity} = \frac{\sum_{i=1}^{N} (x_i - \max(x_i - \hat{x}_i, 0))}{\sum_{i=1}^{N} x_i}, \tag{30}$$

$$\text{FDR} = \frac{\sum_{i=1}^{N} \max(\hat{x}_i - x_i, 0)}{\sum_{i=1}^{N} \hat{x}_i}, \tag{31}$$

where x_i is the number of simulated transcript counts and $\hat{x}_i$ is the number of decoded transcript counts for the ith gene. N is the number of genes probed in the simulation.

Estimating sensitivity relative to smFISH in real data

For the sensitivity estimate of our method in cell culture, we compare the average number of barcodes of selected genes found per cell using our multiplexed seqFISH method to a comparison experiment that counts copies of mRNAs of the same genes using un-multiplexed sequential single-molecule FISH (smFISH) in a separate sample of cells from the same culture. The expression level distribution of the genes we chose to count with smFISH is representative of the expression level distribution of the entire panel probed using seqFISH (Supplementary Fig. D.3B), We then fit a linear regression model of the average counts of mRNAs of each gene found per

cell by seqFISH explained by each gene's average mRNA count per cell as found by smFISH. The slope of the regression curve is the estimated sensitivity (Fig. 0.2E).

Estimating overall false discovery rate in real data

To estimate the FDR in experiments, we decode each experiment twice: once using a codebook containing only codewords for which we designed probes to readout real transcripts and once using a codebook containing all of the codewords in the first codebook and all additional unused codewords of the same weight in the error-correcting code that are not probed for. These additional codewords serve as computational negative controls. Using two decoding runs improves the quality of the decoding results and the FDR estimate by eliminating competition from negative control codewords when decoding gene encoding codewords. We assume that negative control codewords are erroneously decoded at the same rate as gene encoding codewords on average (Appendix B). We estimate the average rate at which codewords are decoded erroneously as

$$\hat{\text{FDR}}_{\text{per codeword}} = \frac{B_{nc}}{C_{nc}}, \tag{32}$$

where B_{nc} is the number of negative control barcodes found when decoding including negative control codewords, and C_{nc} is the number of negative control codewords. We can estimate the number of falsely discovered gene encoding barcodes by multiplying the estimated false discovery rate per codeword, B_{nc}/C_{nc}, by the number of gene encoding codewords, C_g. To estimate the proportion of gene encoding barcodes that are found in error (FDR), we divide the expected number of falsely discovered gene encoding barcodes by the total number of gene encoding barcodes found when decoding only gene encoding codewords, B_g:

$$\hat{\text{FDR}} = \frac{C_g \frac{B_{nc}}{C_{nc}}}{B_g}. \tag{33}$$

This equation has been used in previous works [16, 17, 22, 23].

Estimating FDR at different densities in real data

We estimate false discovery rates in real data at different local barcode densities using the same approach as for estimating the overall FDR, but partitioning the barcodes by local density. To find the local density of each barcode in each cell, we first construct a KDTree containing the spatial coordinates of barcodes found in the gene-encoding-only decoding run and a KDTree containing the negative

control barcodes found for that cell. We then use each cell's KDTree to find the number of barcodes within a 100 nm radius of each gene-encoding barcode found in the gene-encoding-only decoding run (including itself) and each negative control barcode found in the decoding run that included negative control barcodes. We then estimated FDR at each density using equation 33 with the number of gene encoding and negative control barcodes found at that density (Fig. 0.2D).

Primary-probe design

Gene-specific primary probes were designed as previously described with some modifications[10]. The probe length was set to be 33 nucleotides (nt), and a local BLAST query was run on each probe against the mouse transcriptome and a local database constructed from the probe sequences to ensure specificity. We managed to find 24 probes for the 271 selected highly expressed genes. We designed four secondary readout probe hybridization sites on each primary probe, with two readout probe hybridization sites on each end of the primary probe and the RNA-binding site in the middle. We selected 49 genes from the 271 genes for the smFISH experiment, with 36 probes per gene and two readout sites on each primary probe. The probe sequences are available as Supplementary Data.

Primary-probe construction

Primary probes were generated from oligoarray pools (Twist Bioscience) as previously described [10]. In brief, probe sequences were amplified from the oligonucleotide pools with limited two-step PCR cycles, and PCR products were purified using QIAquick PCR Purification Kit (Qiagen 28104). Then, in vitro transcription (NEB E2040S) was performed, followed by reverse transcription (Thermo Fisher EP0751). After reverse transcription, the single-stranded DNA (ssDNA) probes were alkaline hydrolyzed with 1 M NaOH at 65°C for 20 min to degrade the RNA templates and neutralized with 1 M acetic acid. Finally, probes were purified with SPRIselect beads (Beckman Coulter B23318) and eluted in nuclease-free water. Probe concentration was measured by Nanodrop.

Readout-probe design and synthesis

Readout probes of 15 nt in length were designed as previously described [10]. In brief, a set of probe sequences was randomly generated with combinations of A, T, G or C nucleotides. Readout-probe sequences within a GC-content range of 40–60% were selected. We performed a BLAST search against the mouse transcriptome to

ensure the specificity of the readout probes. To minimize cross-hybridization of the readout probes, any probes with ten contiguously matching sequences between readout probes were removed. The reverse complements of these readout-probe sequences were included in the primary probes according to the designed barcodes. Readout probes of 15 nt in length were ordered from Integrated DNA Technologies as 5'-amine modified. The construction of readout probes was similar to that previously described [10]. In brief, 5 nmol DNA probes were mixed with 50 nmol of Alexa Fluor 647–NHS ester (A20006, Thermo Fisher) or Cy3B (PA63101, Cytiva) or Alexa Fluor 488–NHS ester (A30005, Thermo Fisher) in 0.5 M sodium bicarbonate buffer containing 5% dimethyl sulfoxide. The reaction was incubated at 37°C. After 1 hr, fresh 50 nmol of Alexa Fluor 647–NHS ester or Cy3B or Alexa Fluor 488–NHS ester was added to the solution. After another hour, 50 nmol of Alexa Fluor 647–NHS ester or Cy3B or Alexa Fluor 488–NHS ester was added a third time, making the final concentration of 3 mM in 0.5 M sodium bicarbonate buffer containing 15% dimethyl sulfoxide. Then, the DNA probes were subjected to ethanol precipitation, high-pressure liquid chromatography (HPLC) purification and column purification to remove all contaminants. Once resuspended in water, the readout probe concentration was quantified using Nanodrop and a 500 nM working stock was made. All the readout probes were stored at -20°C.

Coverslip functionalization

The coverslips were rinsed with 100% ethanol three times, and heat-dried in an oven at >90°C for 30 min. Then, the coverslips were cleaned with a plasma cleaner on a high setting (PDC-001, Harrick Plasma) for 5 min. Next, the coverslips were treated with 100 μg μ l^{-1} of poly-d-lysine (P6407; Sigma) in water for >3 hrs at room temperature, followed by three rinses with water. The coverslips were then air-dried and kept at 4°C for no longer than 7 days.

Cell culture experiment

NIH/3T3 cells (ATCC) were cultured as previously described on the functionalized coverslips to ~80–90% confluence [10]. Then, the cells were washed with 1× PBS once and fixed with freshly made 4% formaldehyde (28906, Thermo Fisher) in 1× PBS (AM9624, Invitrogen) at room temperature for 10 min. The fixed cells were permeabilized with 70% ethanol for 2 hrs at room temperature or overnight at -20°C. The cell samples were dried, and custom-made flow cells were attached to the coverslip. The samples were rinsed with three flow cell volumes (approximately 120

μl) of 1X PBS and post-fixed with freshly made 7.5 mM BS-PEG5 (A35396, Thermo Fisher) in 1x PBS for 15 min at room temperature twice. Samples were rinsed with 3 flow cell volumes of 1X PBS to remove excess BS-PEG5. Afterwards, amines on the sample were blocked with 100 mM N-(Propionyloxy)succinimide (93535, Sigma) in 1x PBS. Samples were treated with 100mM N-(Propionyloxy)succinimide (NHS) in 1X PBS and 10% Dimethyl sulfoxide for 1 hrs at room temperature for three times (3 hrs total). The NHS solution was removed, and the samples were rinsed with 5 flow cell volumes (approximately 200 μl) of 25% wash buffer (25% Formamide (4610, Sigma), 2x SSC, 0.1% Triton-X100 (93443-100ML, Sigma)). Hybridization buffer containing 0.1 mg/ml yeast tRNA (AM7119, Thermo Fisher) and 2 μM polyTTG 200 nt (IDT) was added and the samples were incubated at 37°C for 1 hrs. Finally, primary probes (5 nM per probe for 24 probes per gene) were hybridized to the 271 genes in the hybridization buffer. The hybridization was allowed to proceed for 36 hrs in a humid chamber at 37°C. After hybridization, the samples were washed with 3 flow cell volumes of 30% wash buffer (30% Formamide, 2x SSC, 0.1% Triton-X100), followed by an incubation at 37°C for 30 min. Afterwards, the samples were rinsed with 6 flow cell volumes (approximately 240 μl) of 4X SSC. Next, Tetraspeck beads (T7280, Thermo Fisher) were sonicated for 5 mins, diluted 1:100 in 4X SSC and vortexed. Tetraspeck beads were applied to the cell samples and incubated at room temperature for 5 min. Then, the sample was washed with 3 flow cell volumes of 1X PBS and post-fixed with 4% PFA in 1x PBS for 10 min at room temperature. Finally, the sample was washed with 6 flow-cell volumes of 4X SSC and sealed. The sample can be stored at 4°C for 2 weeks before imaging.

Mice

All animal care and experimental procedures were approved by the Caltech Office of Laboratory Animal Resources (OLAR). Wild-type C57BL/6J mice (8–12 weeks old; The Jackson Laboratory) were used for RS-code tissue experiments.

Mice were euthanized by CO_2 inhalation, and brains were rapidly dissected following confirmation of death. Tissues were embedded in optimal cutting temperature (OCT) compound and snap-frozen in 2-methylbutane pre-cooled on dry ice. Embedded samples were stored at -80°C until cryosectioning.

Tissue slices experiment

Fresh-frozen mouse brain sections (10μm) were fixed in 4% paraformaldehyde (PFA) in 1× PBS for 10 min at room temperature, washed with 1× PBS, dehydrated

in 10% isopropanol for 1 min, air-dried and stored at -80°C. Before processing, sections were permeabilized in 70% ethanol overnight at 4°C, treated with 8% Triton X-100 in 1× PBS for 1h at room temperature, rinsed with 70% ethanol and air-dried. Flow cells were attached prior to subsequent processing.

Sections were post-fixed twice with 7.5 mM BS-PEG5 in 1× PBS for 15 min at room temperature. Free amines were blocked with 100 mM N-(Propionyloxy)succinimide in 1× PBS (4 × 45 min at room temperature), followed by five washes with 25% wash buffer (25% formamide, 2× SSC, 0.1% Triton X-100). Subsequent procedures were identical to those described for cell culture experiments, except that primary probe hybridization was performed for 48h at 37°C in a humidified chamber.

SeqFISH imaging for 3T3 cell culture

The imaging platform and automated fluidics delivery system were similar to those previously described with some modifications [10]. In brief, the flow cell on the sample was first connected to the automated fluidics system. Then the region of interest (ROI) was registered using nuclei signals stained with 30 μg ml^{-1} DAPI (D8417; Sigma). Each serial hybridization buffer contained three unique sequences with different concentrations of 15-nt readouts conjugated to either Alexa Fluor 647 (50 nM), Cy3B (50 nM), or Alexa Fluor 488 (100 nM) in 10% hybridization buffer (10% formamide (4610; Sigma), 10% dextran sulfate (D4911; Sigma), and 4× SSC). The 150 μl of serial hybridization buffers for 34 cycles of seqFISH imaging with a repeat for cycle 1 (in total 35 cycles) was pipetted into a 96-well plate. During each serial hybridization, the automated sampler moves to the well of the designated hybridization buffer and moves the 150 μl hybridization buffer through a multichannel fluidic valve (EZ1213-820-4; IDEX Health & Science) to the flow cell (requires $\sim 40\mu$l) using a syringe pump (63133-01, Hamilton Company). The serial hybridization solution was incubated for 20 min at room temperature. After serial hybridization, the sample was washed with 500 μl of 15% formamide wash buffer (15% formamide and 0.1% Triton X-100 in 2× SSC) to remove excess readout probes and non-specific binding. Then, the sample was rinsed with 500 μl of 4× SSC, before being stained with DAPI solution (30 μg ml^{-1} of DAPI in 4× SSC) for 2 mins. Next, an anti-bleaching buffer solution made of 10% (w/v) glucose, 1:100 diluted catalase (C3155, Sigma), 1 mg ml^{-1} glucose oxidase (G2133, Sigma), 5 mM Trolox (648471, Sigma), and 50 mM pH 8 Tris-HCl in 4× SSC was flowed through the samples. Imaging was done with a microscope (Leica DMi8) equipped with a confocal unit and lasers (Andor Dragonfly 200), a sCMOS camera (Andor

Zyla 4.2 Plus), a 63× oil objective lens (Leica 1.40 NA) and a motorized stage (TMC 63-563). Snapshots were acquired with 0.25-μm z-steps for 17 z-slices per field of view across 647-nm, 561-nm, 488-nm, and 405-nm fluorescent channels. Alexa Fluor 647, Cy3B, and Alexa Fluor 488 channels were acquired with 10% laser power and a 3,000-ms exposure time, while DAPI images were acquired with 10% laser power and a 200-ms exposure time. After imaging, stripping buffer (60% formamide and 0.1% Triton-X 100 in 2× SSC) was flowed through for 1 min, followed by an incubation time of 3 min before rinsing with 4× SSC solution. The serial hybridization, imaging and signal extinguishing steps were repeated for 34 cycles. Then, staining buffer for segmentation (50 nM LNA T20-Alexa Fluor 647 in 4× SSC) was flowed in and allowed to incubate for 20 min at room temperature before imaging. Stripping buffer was applied to the sample to strip off the segmentation marker. Final blank images containing beads only were imaged as the last hybridization cycle. The integration of the automated fluidics delivery system and imaging was controlled by a custom-written script in μManager.

SeqFISH imaging for mouse brain slices

For mouse brain seqFISH imaging, images were acquired across 11 z-slices with 1-μm z step. All 100 readout probes were conjugated to Cy3B and sequentially imaged over 100 hybridization cycles. Cy3B images were acquired with 10% laser power and a 3,000-ms exposure time. All other imaging and fluidics procedures were performed as described above for 3T3 cell culture seqFISH imaging.

smFISH experiment for 3T3 cell culture

smFISH experiments were performed as previously described [10]. The fixed cells were blocked and hybridized with the primary probes (10 nM per probe) in 25% hybridization buffer (25% formamide, 10% dextran sulfate and 4× SSC) at 37°C for 24 hrs. The sample was washed with 30% wash buffer for 30 min at 37°C before imaging. The imaging platform was the same as the one in the seqFISH experiment. The serial hybridization, imaging, and signal extinguishing steps were repeated for 17 cycles. The sum of the gene counts per cell was analyzed using a previously published Python pipeline [16, 17].

References

1. Lin, S. & Costello, D. J. *Error control coding. Fundamentals and applications* 2. ed. Formerly CIP. 1260 pp. ISBN: 0-13-042672-5 (Pearson/Prentice Hall, Upper Saddle River, NJ, 2004).

2. Lubeck, E., Coskun, A. F., Zhiyentayev, T., Ahmad, M. & Cai, L. Single-cell in situ RNA profiling by sequential hybridization. *Nature Methods* **11,** 360–361. ISSN: 1548-7105 (Mar. 2014).

3. Chen, K. H., Boettiger, A. N., Moffitt, J. R., Wang, S. & Zhuang, X. Spatially resolved, highly multiplexed RNA profiling in single cells. *Science* **348.** ISSN: 1095-9203 (Apr. 2015).

4. Gyllborg, D. *et al.* Hybridization-based in situ sequencing (HybISS) for spatially resolved transcriptomics in human and mouse brain tissue. *Nucleic Acids Research* **48,** e112–e112. ISSN: 1362-4962 (Sept. 2020).

5. Wang, X. *et al.* Three-dimensional intact-tissue sequencing of single-cell transcriptional states. *Science* **361.** ISSN: 1095-9203 (July 2018).

6. Femino, A. M., Fay, F. S., Fogarty, K. & Singer, R. H. Visualization of Single RNA Transcripts in Situ. *Science* **280,** 585–590. ISSN: 1095-9203 (Apr. 1998).

7. Raj, A., van den Bogaard, P., Rifkin, S. A., van Oudenaarden, A. & Tyagi, S. Imaging individual mRNA molecules using multiple singly labeled probes. *Nature Methods* **5,** 877–879. ISSN: 1548-7105 (Sept. 2008).

8. Eng, C.-H. L., Shah, S., Thomassie, J. & Cai, L. Profiling the transcriptome with RNA SPOTs. *Nature Methods* **14,** 1153–1155. ISSN: 1548-7105 (Nov. 2017).

9. Shah, S. *et al.* Dynamics and Spatial Genomics of the Nascent Transcriptome by Intron seqFISH. *Cell* **174,** 363–376.e16. ISSN: 0092-8674 (July 2018).

10. Eng, C.-H. L. *et al.* Transcriptome-scale super-resolved imaging in tissues by RNA seqFISH+. *Nature* **568,** 235–239. ISSN: 1476-4687 (Mar. 2019).

11. Shah, S., Lubeck, E., Zhou, W. & Cai, L. In Situ Transcription Profiling of Single Cells Reveals Spatial Organization of Cells in the Mouse Hippocampus. *Neuron* **92,** 342–357. ISSN: 08966273. `https://linkinghub.elsevier.com/retrieve/pii/S0896627316307024` (2022) (Oct. 2016).

12. Xia, C., Fan, J., Emanuel, G., Hao, J. & Zhuang, X. Spatial transcriptome profiling by MERFISH reveals subcellular RNA compartmentalization and cell cycle-dependent gene expression. *Proceedings of the National Academy of Sciences* **116,** 19490–19499. ISSN: 1091-6490 (Sept. 2019).

13. Takei, Y. *et al.* Spatial multi-omics reveals cell-type-specific nuclear compartments. *Nature* **641,** 1037–1047. ISSN: 1476-4687 (Apr. 2025).

14. Guizar-Sicairos, M., Thurman, S. T. & Fienup, J. R. Efficient subpixel image registration algorithms. *Optics Letters* **33,** 156. ISSN: 1539-4794 (Jan. 2008).

15. Parthasarathy, R. Rapid, accurate particle tracking by calculation of radial symmetry centers. *Nature Methods* **9,** 724–726. ISSN: 1548-7105 (June 2012).

16. Polonsky, M. *et al.* Spatial transcriptomics defines injury specific microenvironments and cellular interactions in kidney regeneration and disease. *Nature Communications* **15** (Sept. 2024).

17. Colón, K. L. In Situ Signal Amplification for Spatial Transcriptomics Using Programmable DNA Assemblies. en (2025).

18. Laubscher, E. *et al.* Accurate single-molecule spot detection for image-based spatial transcriptomics with weakly supervised deep learning. *Cell Systems* **15,** 475–482.e6. ISSN: 2405-4712 (May 2024).

19. Babcock, H., Sigal, Y. M. & Zhuang, X. A high-density 3D localization algorithm for stochastic optical reconstruction microscopy. *Optical Nanoscopy* **1,** 6. ISSN: 2192-2853 (2012).

20. Partel, G. & Wählby, C. *Graph-based image decoding for multiplexed in situ RNA detection* in *2020 25th International Conference on Pattern Recognition (ICPR)* (2021), 3783–3790.

21. Chen, S. *et al.* BARcode DEmixing through Non-negative Spatial Regression (BarDensr). *PLOS Computational Biology* **17** (ed Robinson, E. C.) e1008256. ISSN: 1553-7358 (Mar. 2021).

22. Andersson, A., Diego, F., Hamprecht, F. A. & Wählby, C. ISTDECO: In Situ Transcriptomics Decoding by Deconvolution (Mar. 2021).

23. Bryan, J. P. *et al.* Optimization-based decoding of Imaging Spatial Transcriptomics data. *Bioinformatics* **39** (ed Peng, H.) ISSN: 1367-4811 (June 2023).

24. Gataric, M. *et al.* PoSTcode: Probabilistic image-based spatial transcriptomics decoder (Oct. 2021).

25. Axelrod, S. *et al.* starfish: scalable pipelines for image-based transcriptomics. *Journal of Open Source Software* **6,** 2440. ISSN: 2475-9066 (May 2021).

26. Takei, Y. *et al.* Integrated spatial genomics reveals global architecture of single nuclei. *Nature* **590,** 344–350. ISSN: 1476-4687 (Jan. 2021).

27. Reed, I. S. & Solomon, G. Polynomial Codes Over Certain Finite Fields. *Journal of the Society for Industrial and Applied Mathematics* **8,** 300–304. ISSN: 2168-3484 (June 1960).

28. Vizgen. *MerFISH 2.0 Data Release* Web. `https://vizgen.com/data-release-program/`.

29. Bruker. *CosMx SMI Mouse Brain FFPE Dataset* Web. `https://brukerspatialbiology.com/products/cosmx-spatial-molecular-imager/ffpe-dataset/cosmx-smi-mouse-brain-ffpe-dataset/`.

30. 10X-Genomics. *Explore Mouse Brain Neurobiology* Web. `https://www.10xgenomics.com/products/xenium-in-situ/mouse-brain-dataset-explorer`.

31. Yao, Z. *et al.* A high-resolution transcriptomic and spatial atlas of cell types in the whole mouse brain. *Nature* **624,** 317–332. ISSN: 1476-4687 (Dec. 2023).

32. Zhu, Q. *et al.* Revealing a coherent cell-state landscape across single-cell datasets with CONCORD. *Nature Biotechnology.* ISSN: 1546-1696 (Jan. 2026).

33. Sage, D. *et al.* Super-resolution fight club: assessment of 2D and 3D single-molecule localization microscopy software. *Nature Methods* **16,** 387–395. ISSN: 1548-7105 (Apr. 2019).

34. Boyd, N., Schiebinger, G. & Recht, B. The Alternating Descent Conditional Gradient Method for Sparse Inverse Problems. *SIAM Journal on Optimization* **27,** 616–639. ISSN: 1095-7189 (Jan. 2017).

35. Moffitt, J. R. *et al.* High-throughput single-cell gene-expression profiling with multiplexed error-robust fluorescence in situ hybridization. *Proceedings of the National Academy of Sciences* **113,** 11046–11051. ISSN: 1091-6490 (Sept. 2016).

36. Groth, E. J. A pattern-matching algorithm for two-dimensional coordinate lists. *The Astronomical Journal* **91,** 1244. ISSN: 0004-6256 (May 1986).

37. N.J.A. Sloane, F. J. M. *The Theory of Error-Correcting Codes* ISBN: 0-444-8500-0 (North-Holland Publishing Company, NY, 1977).

38. Dunning, I., Huchette, J. & Lubin, M. JuMP: A Modeling Language for Mathematical Optimization. *SIAM Review* **59,** 295–320. ISSN: 1095-7200 (Jan. 2017).

39. Mölder, F. *et al.* Sustainable data analysis with Snakemake. *F1000Research* **10,** 33. ISSN: 2046-1402 (Apr. 2021).

40. Sternberg, S. R. Biomedical Image Processing. *Computer* **16,** 22–34. ISSN: 0018-9162 (Jan. 1983).

41. Pachitariu, M. & Stringer, C. Cellpose 2.0: how to train your own model. *Nature Methods* **19,** 1634–1641. ISSN: 1548-7105 (Nov. 2022).

42. Pachitariu, M., Rariden, M. & Stringer, C. Cellpose-SAM: superhuman generalization for cellular segmentation (May 2025).

43. Zhu, L., Zhang, W., Elnatan, D. & Huang, B. Faster STORM using compressed sensing. *Nature Methods* **9,** 721–723. ISSN: 1548-7105 (Apr. 2012).

44. Friedman, J., Hastie, T. & Tibshirani, R. Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of Statistical Software* **33.** ISSN: 1548-7660 (2010).

45. Candes, E. & Tao, T. Decoding by linear programming. *IEEE Transactions on Information Theory* **51,** 4203–4215 (2005).

46. Davenport, M. A., Duarte, M. F., Eldar, Y. C. & Kutyniok, G. Introduction to compressed sensing, 1–64 (May 2012).

47. Hazimeh, H. & Mazumder, R. Fast Best Subset Selection: Coordinate Descent and Local Combinatorial Optimization Algorithms. *Operations Research* **68,** 1517–1537. eprint: https://doi.org/10.1287/opre.2019.1919. https://doi.org/10.1287/opre.2019.1919 (2020).

48. Hazimeh, H., Mazumder, R. & Nonet, T. L0Learn: A Scalable Package for Sparse Learning using L0 Regularization. *Journal of Machine Learning Research* **24,** 1–8. http://jmlr.org/papers/v24/22-0189.html (2023).

49. Yuan, M. & Lin, Y. Model Selection and Estimation in Regression with Grouped Variables. *Journal of the Royal Statistical Society Series B: Statistical Methodology* **68,** 49–67. ISSN: 1369-7412. https://doi.org/10.1111/j.1467-9868.2005.00532.x (Dec. 2005).

Appendix A

AN EXPLANATION OF ERROR-CORRECTING CODES IN FISH-BASED SPATIAL GENOMICS

Linear error-correcting codes use codewords that are composed of sequences of symbols. Codewords can be divided into information symbols, which alone represent a message containing all the information encoded in a codeword, and parity check symbols, which are linear combinations of the information symbols that add redundancy and robustness to the codeword. The size of the alphabet from which the symbols are drawn is denoted as q, the number of symbols in a codeword is denoted as n, the number of information symbols is denoted as k, and the number of parity check symbols is denoted as $n - k$. The number of non-zero symbols in a codeword is called the weight of the codeword and is denoted as w.

In seqFISH, we conceptualize symbols as being transmitted from barcoded objects through a multicolored image where signals appear as dots and encode a symbol value determined by their color. In the first seqFISH experiments, each symbol was represented by a distinct fluorophore emitting a different color in the same hybridization [2, 11]. Later implementations of seqFISH expanded symbol block images to include pseudo-colors from images acquired in different hybridization cycles [8, 9]. This enabled experiments with expanded multiplexing capacity and reduced optical density by allowing larger alphabets ($q > 5$) from which symbols of the error-correcting code are drawn. SeqFISH, until now, has used q-ary parity check codes with $n = 4$ and $k = 3$ or $n = 5$ with $k = 3$ or 4 where all symbols in the alphabet, including zero, are represented by a pseudo-color. MERFISH uses weight-four codewords from binary error-correcting codes ($q = 2$) where zeros are represented by the absence of dots and are not probed. MERFISH has used Hamming codes [3] and non-linear codes [12].

One measure of an error-correcting code's robustness is its minimum Hamming distance between any pair of its codewords (MHD). Reed-Solomon codes are members of a special class of linear error-correcting codes, called maximum distance separable codes, for which MHD $= n - k + 1$. There are ways to increase the robustness of a code in ways that are not measured by the MHD. For example, Reed-Solomon codes use symbols from alphabets with size equal to a non-negative integer power of prime

numbers. This enables using symbol blocks with multiple pseudo-colors, adding an extra rule that reduces ambiguity in superimposed barcodes: each barcode can only have one value-representing dot in each symbol block. Increasing the average Hamming distance between codewords without increasing the MHD of a code can also improve robustness. We find that thinning a codebook to not include all available codewords or adding parity checks that increase the average Hamming distance but not MHD can also increase the robustness of codes (Supplementary Fig. D.2A). For linear error-correcting codes, the MHD is equal to the minimum weight of its non-zero codewords [1]. However, non-linear error-correcting codes, which do not have parity check symbols that are linear combinations of information symbols, are not subject to this constraint. This may provide an opportunity to encode information more efficiently and robustly at the cost of more computationally expensive decoding. To test the potential of non-linear error-correcting codes, we designed a non-linear error-correcting code. We first chose a minimum Hamming distance of 6, iterated over all combinations of 100 choose 5, and then added any combination to our codebook that is at least 6 Hamming distance away from any combination that had previously been added (Supplementary Fig. D.2A). We expect that it is possible to construct better non-linear codes using more sophisticated discrete mathematics.

Appendix B

NOTE ON FALSE DISCOVERY RATE ESTIMATES

Our false discovery rate estimator, main text equation 33, relies on the assumption that, on average, negative control barcodes are decoded at the same rate per negative control codeword as gene encoding barcodes are erroneously decoded per gene encoding codeword. We have found that barcodes for some negative control codewords are found more often than others. However, we can explain most of the variation ($R^2 = 0.51$) in the decoding frequency of different negative control codewords using a linear model with explanatory variables constructed by counting the number of instances in which spatially overlapping gene encoding barcodes found in the decoding runs including only gene encoding barcodes recombine or nearly recombine into negative control barcodes and the total expression counts found of gene encoding barcodes in images where each negative control barcode is searched for (Supplementary Fig. D.3C).

We now describe in detail how the model is constructed. G and C are binary representations of the codebook matrices for gene encoding and negative control codewords, respectively. G is of dimension $n_g \times (n * (q-1))$ and C is of dimension $n_c \times (n * (q-1))$, where n_g is the number of gene encoding codewords, n_c is the number of negative control codewords, n is the number symbols in a codeword of the error-correcting code, q is the size of the symbol alphabet for the error-correcting code, and $q-1$ is the number of pseudo-colors. Each row of these matrices represents a codeword. Each column corresponds to a readout image in the seqFISH experiment. Elements of G and C, s_{ij}, are valued as

$$s_{ij} = \begin{cases} 1, & \text{if the seqFISH representation of the } i\text{th codeword includes a signal in the } j\text{th image} \\ 0, & \text{otherwise.} \end{cases} \tag{B.1}$$

We next define upper triangular overlap recombination matrices of dimension $n_g \times n_g$, $O(k)$, for each negative control codeword of arbitrary index k. Upper triangular elements of $O(k)$, $o_{ij}(k)$, count the number of dots needed to represent the kth negative control codeword that are available to recombine from spatial overlaps of

the ith and jth gene encoding codewords

$$o_{ij}(k) = \begin{cases} (\mathbf{g_i} \vee \mathbf{g_j})\mathbf{c_k}', & \text{if } j > i \\ 0, & \text{otherwise,} \end{cases} \tag{B.2}$$

where $\mathbf{g_i}$ and $\mathbf{g_j}$ are the ith and jth row vectors of G, the binary representations of the ith and jth gene encoding codewords, respectively. $\mathbf{c_k}$ is the kth row vector of C, the kth binary negative control codeword vector, and $\vee$ is the element-wise logical or operator.

We construct recombination frequency matrices of dimension $n_g \times n_g$, $P(r)$, that hold the number of co-occurrences of gene encoding barcodes of different kinds within a search radius, r. Elements of $P(r)$, $p_{ij}(r)$, are equal to the number of instances in the entire dataset in which barcodes of the ith codeword are located within a spatial distance r of the barcodes of the jth codeword. To aid us in writing this definition formally, let be $\mathbf{u_{ik}}$ the position vector of the kth barcode of the ith gene $\mathbf{v_{jl}}$ is the position vector of the lth barcode of the jth negative control codeword. Then,

$$p_{ij}(r) = \begin{cases} \text{sum}\left(\left\{\delta\left(||\mathbf{u_{ik}} - \mathbf{v_{jl}}||_2 \leq r\right) \middle| k \text{ in } \{1, 2, ..., n_i\}, l \text{ in } \{1, 2, ..., n_j\}\right\}\right), & \text{if } j > i \\ 0, & \text{otherwise,} \end{cases} \tag{B.3}$$

where n_i and n_j are the number of barcodes that were decoded for the ith and jth codewords respectively. Now we can define the vector-valued function

$$\boldsymbol{\rho}(n_\rho, r) = [\rho_1(n_\rho, r), \rho_2(n_\rho, r), ..., \rho_k(n_\rho, r), ..., \rho_{n_c}(n_\rho r)], \tag{B.4}$$

that generates explanatory variables of length n_c for our linear model. The kth element of $\boldsymbol{\rho}(n_\rho, r)$ is calculated as

$$\rho_k(n_\rho, r) = \text{sum}\Big(P\big(r\big) \odot \big(O(k) == n_\rho\big)\Big), \tag{B.5}$$

where n_ρ is the desired number of dots from a target codeword found in an overlap, r is the search radius within which barcodes are considered overlapping, == is the element-wise logical equal to operator, and $\odot$ is the element-wise multiplication operator (Hadamard product operator). Our linear model predicts the frequencies at which barcodes for each negative control codeword, $\mathbf{d_c}$, are found as

$$\hat{\mathbf{d}}_\mathbf{c} = \beta_1 + \beta_2(CG'\mathbf{d_g}) + \beta_3\boldsymbol{\rho}(3,1) + \beta_4\boldsymbol{\rho}(4,1) + \beta_5\boldsymbol{\rho}(3,2) + \beta_6\boldsymbol{\rho}(4,2), \tag{B.6}$$

where $CG'\mathbf{d_g}$ gives the exposure of each negative control codeword to dots transmitted by gene encoding codewords encoded in the same images, $\mathbf{d_g}$ is the vector containing the number barcodes for each gene that were found in the decoding run that considered only gene encoding barcodes, β_1 is the intercept and β_2 to β_6 are fit coefficients.

Our model does not include possible effects from registration errors (Supplementary Fig. D.5D), which are unevenly distributed in readout images because they are different in each field of view and would require a much more complicated model to account for. Other factors that may contribute to variation in the frequency at which different negative control barcodes are decoded may include non-specific binding and failure to distinguish spatially overlapping dots read out from two different molecules in the same image. It is likely that negative control barcodes are found in error more often than gene encoding barcodes because we construct codebooks to maximize the distance between gene encoding barcodes, meaning that gene encoding barcodes are on average more similar to negative control barcodes than to each other.

Appendix C

SEQFISH+ SIMULATIONS AND LIMITS ON RESOLVING DENSE BARCODES FROM IMAGE FITTING

C.1 Spots-first simulations

The fitting algorithm we use in our spots-first decoding workflow, ADCG, cannot effectively resolve separate point spread functions in the same optical image centered within a full width at half maximum of each other. Partially overlapping dots in the same image may also cause fitting errors, resulting in fit dots whose coordinates do not correspond to those of the actual molecules. To investigate the impact of errors in image fitting on decoding, we simulated synthetic image data and dot location data modeled to approximate the conditions of the previously published 10,000 gene seqFISH+ experiment [8]. In these simulations, not only were simulated dot locations fed directly into the decoder, but synthetic images were produced as a linear combination of Gaussian point spread functions centered at each simulated dot location. Images were fit with ADCG[34], then the fit dot locations recovered from the image fitting were decoded. We simulated 70 replicates of seqFISH+ data in a 5×5 pixel ROI with various densities of barcodes and non-specific dots with locations drawn from the uniform distribution. The x and y-coordinates of dots read out from each barcode are drawn from a normal distribution with a standard deviation of 0.5 pixels around the simulated true location of the barcode. Simulated images are generated as linear combinations of Gaussian point spread functions with a sigma of 1.2 pixels and the same brightness from both barcodes and non-specific binding. ADCG attempts to recover the dot locations from the simulated images, and both the ADCG recovered dots and the directly simulated dot locations are decoded by syndrome decoding with a position variance penalty of 6 and a log brightness variance penalty of 0. We found a decrease in decoding sensitivity and increased robustness to extra non-specific dots in the image-based simulations compared to the image-less simulations (Supplementary Fig. D.6A). Performance metrics for the simulation were calculated as described in the methods section for simulations.

C.2 Resolving overlapping barcodes in single images using regulated regression

The spots-first approach to resolving ambiguity arising from overlapping barcodes only addresses cases in which the overlapping barcodes are not probed in any of the same images. To further increase the density at which we can resolve barcodes, we will need to be able to handle situations when two spatially overlapping barcodes are probed in the same image. It has previously been shown that a compressed sensing approach can resolve overlapping dots in STORM images [43]. Others have used a similar approach to fit sparse models of spatial genomics image data as generated by barcode-specific source functions using l_1-regularization [23]. Here, we show in simulation that this idea can be extended to resolving many imperfectly aligned overlapping barcodes by first identifying candidate barcodes to use as explanatory variables and then using more stringent l_0-regulated regression.

We simulated image data where the location $(x_i,\ y_i)$ of the ith barcode is drawn from a 2D uniform distribution between 0 and 5. Synthetic images are 9×9 pixels. Readout point spread functions with sigma of one from the jth dot of each ith barcode are added to the synthetic images in which they are expected and centered at $x_{ij} = x_i + 2 + \epsilon_{x_{ij}}$ and $y_{ij} = y_i + 2 + \epsilon_{xij}$. The addition of 2 to x_i and y_i serves to pad the edges of the ROI, ameliorating edge effects. The localization errors are drawn as $\epsilon_{xij} \sim N(x_{ij}, 0.2)$ and $\epsilon_{yij} \sim N(y_{ij}, 0.2)$. The first step in the decoding process is to find candidate locations using the same method described in the compressed sensing decoding subsection of the methods section. Next, candidate barcodes are identified by syndrome decoding with a search radius of 0.2 pixels.

We next use an l_0-regulated regression model to choose the final set of barcodes from the candidate list,

$$\hat{\boldsymbol{\beta}} = \text{argmin}\frac{1}{2}\,||\mathbf{y} - A_b\boldsymbol{\beta} - \beta_0||_2^2 + \lambda||\boldsymbol{\beta}||_0. \tag{C.1}$$

This model is nearly identical to equation 27 of the main text. The only differences are that it uses l_0 regularization instead of l_1 regularization, there are no differential penalties for different non-zero elements of $\boldsymbol{\beta}$, and the normalization factor of the sum squared errors term (decided by the L0learn package)[47, 48]. Using this procedure, we were able to nearly perfectly decode dense simulated images encoded using the same Reed-Solomon code experimentally implemented in this manuscript (Supplementary Fig. **??**B) without a duplicate removal step that is needed following optimizing with equation 27 of the main text (Supplementary Fig. D.6B).

Appendix D

SUPPLEMENTARY FIGURES

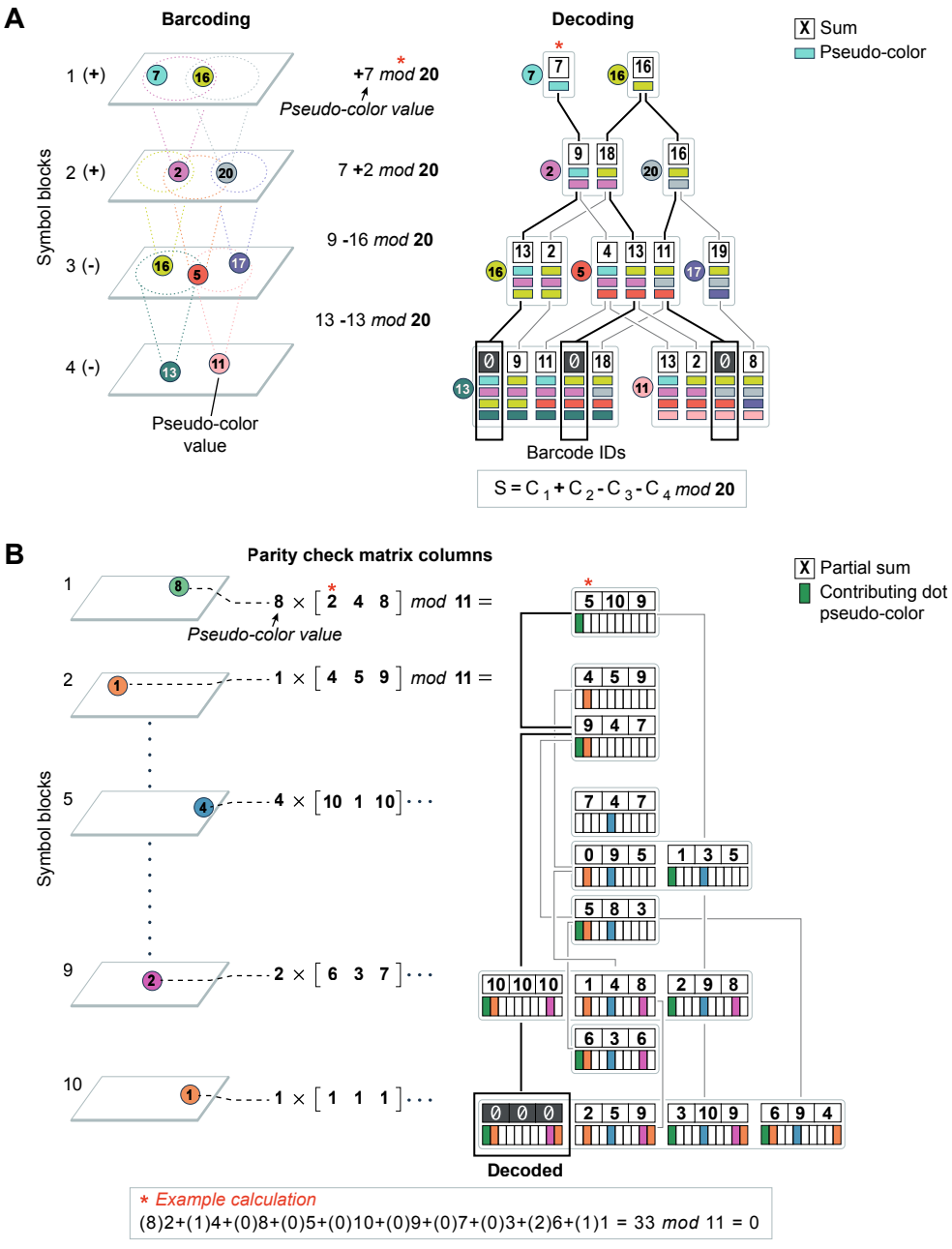

Figure D.1:

Figure D.1: A) We illustrate the simplified dynamic programming algorithm for evaluating syndromes in the q-ary parity check seqFISH code used in the published 10,000 gene seqFISH+ experiment. Candidate barcodes are identified by summing syndromes from numbered pseudo-colors of dots that align in subsequent symbol blocks. Dots in the first symbol block are assigned arrays holding their pseudo-color numbers. Each dot in the second symbol block is assigned an array found by adding its pseudo-color number element-wise to the concatenation of partial sum arrays of dots in the first symbol block that align within a search radius. This procedure is repeated for dots in the third and fourth symbol blocks, but subtracting their pseudo-color number. Sums evaluating to zero in the fourth symbol block indicate valid barcodes where contributing dots are traced. B) We illustrate a more general case of our dynamic programming algorithm for syndrome decoding with the error-correcting code used in our new experiment. This case differs from the case in panel A in two ways. First, three parity check equations are summed, so three partial sums are stored where a single partial sum would be stored in panel A. Second, zero-valued symbols are not probed and are represented by the absence of a dot in a symbol block. Consequently, the dots are aligned to neighbors in multiple preceding symbol blocks and the number of dots contributing to each partial sum is tracked. The dots in the second through $n - w + 1$st symbol block must search for aligning neighbors in all previous symbol blocks. Dots in symbol blocks $b \in [n-(w-2), n]$ must only continue adding to partial sums with neighboring dots containing partial sums of $n - b$ or more dots since w dots are needed to form a barcode.

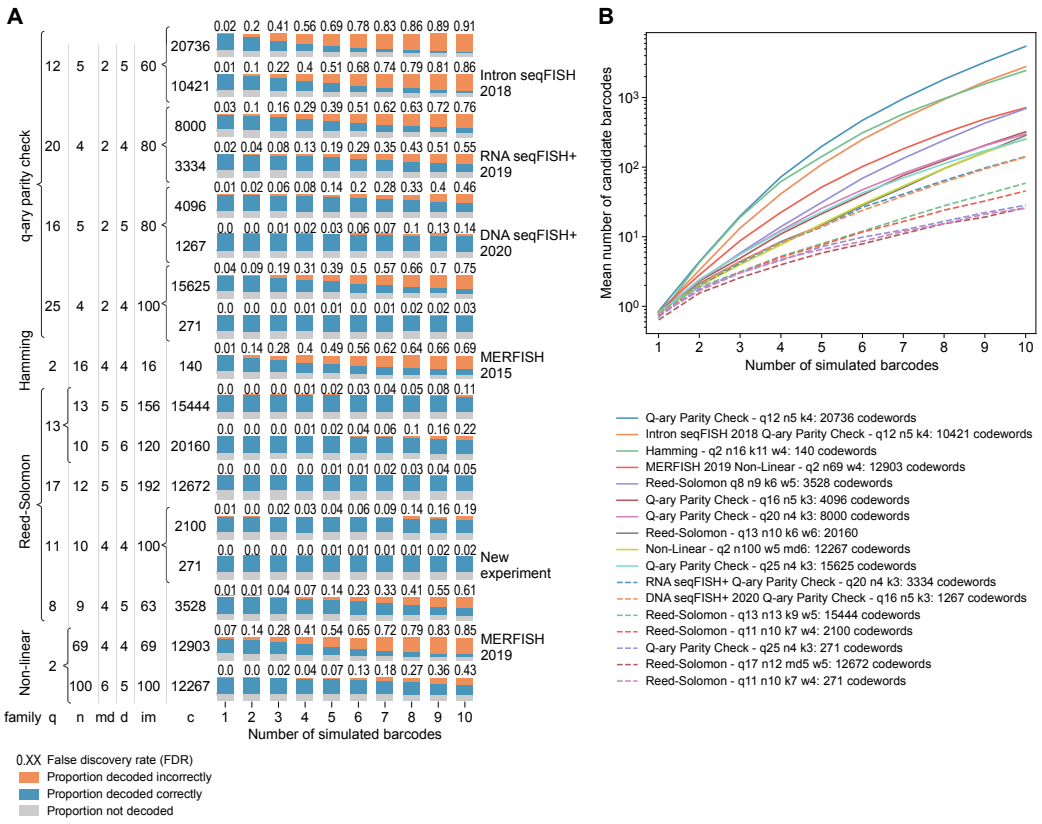

Figure D.2: A) Expanded version of Fig. 0.1E. The first eight rows show simulations for codes from the q-ary parity check code family used in previous seqFISH experiments. Brackets denote branching families of related codes. The first q-ary parity check code in each bracket pair uses all codewords from the code of the given weight. The second uses a subset of the codewords in the first that were used experimentally in this or a previous paper, except for the 271 codeword subset of the q25n4k3 single parity check code, which matches the experimental characteristics of the Reed-Solomon encoded experiment in this work. B) A plot of the mean number of candidate barcodes arising in simulations of various codes as a function of the number of true barcodes in the simulations. These results are drawn from the same simulations shown in panel A. Codes that produce fewer spurious candidate solutions in the simulations have higher decoding sensitivity and lower false discovery rates in the same simulations than codes that produce more spurious candidate solutions.

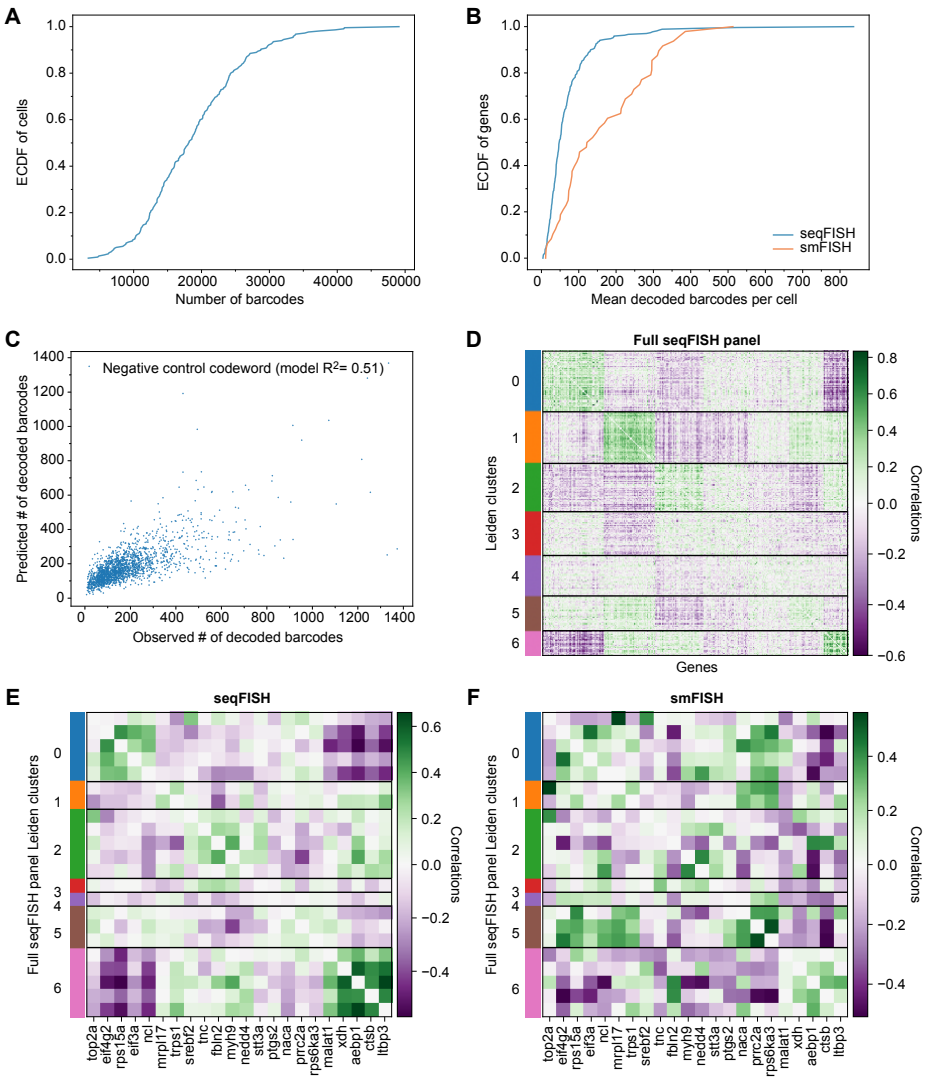

Figure D.3: Summary plots for the Reed-Solomon encoded cross-channel NIH 3T3 cell experiment. A) An empirical cumulative distribution function of the number of gene encoding barcodes decoded in each cell of the cross-channel Reed-Solomon encoded seqFISH using decoding parameters for which summary statistics are reported in the text: lateral variance penalty = 3 and one drop allowed. B) The distribution of the mean barcodes per cell found for genes in the multiplexed seqFISH experiment compared with the un-multiplexed sequential smFISH experiment. The distribution for the seqFISH counts is approximately the same as the distribution for the sequential smFISH counts multiplied by the estimated sensitivity, 0.49 (Fig. 0.2E). C) Barcodes for different negative control barcodes are found at different frequencies. We can explain most of the variation using a linear model that uses variables constructed from the list of gene encoding barcodes per-cell encoded in the gene encoding only decoding run. D) Per-cell gene-gene correlations in our 271 gene 3T3 cell experiment. There are seven Leiden clusters of genes that are correlated with each other and anti-correlated with genes of at least one other cluster. E-F) Panel E shows a subset of the per-cell gene-gene correlation matrix shown in panel D, which is similar to the per-cell gene-gene correlation matrix found using the reference smFISH measurements of the same genes (F).

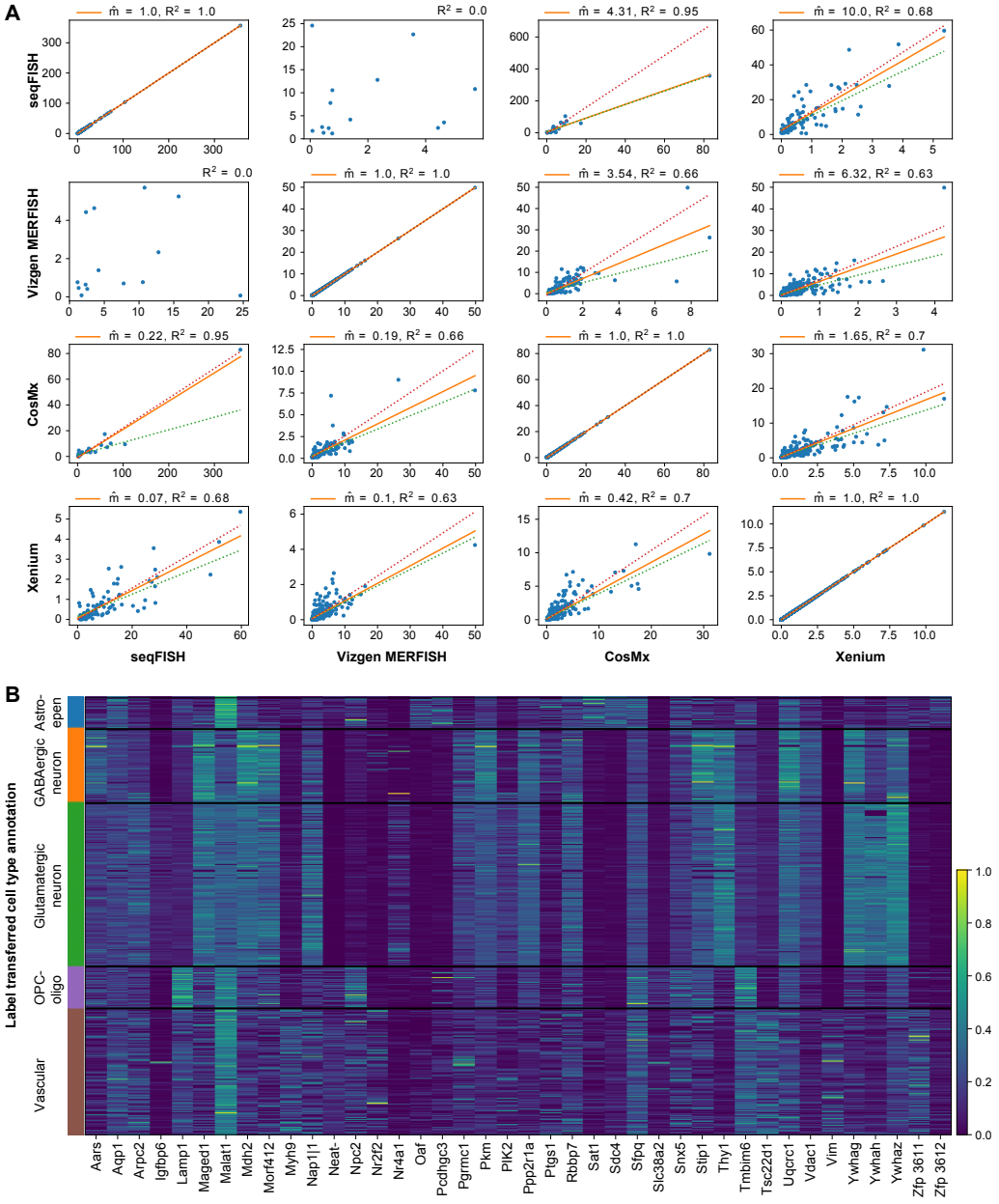

Figure D.4: A) Correlations and relative sensitivities between our method and selected commercial imaging-based spatial genomics methods when applied to the mouse brain. $\hat{m}$ is the predicted slope (relative sensitivity) of the best-fit model using all genes measured by both methods. Dotted lines show the upper (red) and lower (green) bounds of the 95% confidence interval of the slope found using a non-parametric bootstrap. There were too few genes measured in both our seqFISH experiment and the Vizgen sample dataset to find a model with a correlation. B) Gene by cell matrix heatmap for mouse cortex cells profiled by seqFISH, including only cells whose cell type was predicted by a consensus of CONCORD models trained on different balanced subsets of Allen Brain Atlas 10X single cell sequencing data.

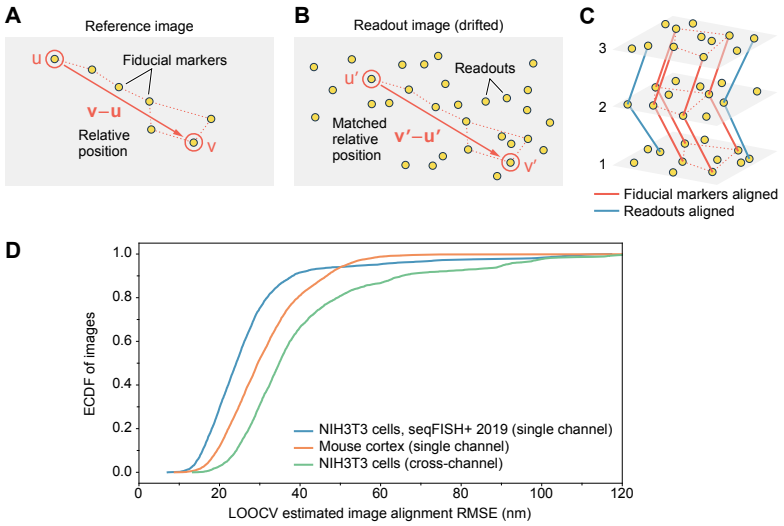

Figure D.5: We register images using diffraction-limited objects as fiducial markers. A) Reference images of the fiducial markers alone show a constellation of fiducial markers. Each pair of fiducial markers, u and v, have a known relative position from each other, $\mathbf{v} - \mathbf{u}$, all of which together describe the shape of the fiducial marker constellation mathematically. B) In readout images, the dots from the fiducial marker constellation are still visible, but there are also additional dots read out from target molecules. To find the constellation of fiducial marker dots in readout images, we first search for a pair of dots, u' and v', at approximately the same relative position to each other as a distant pair of fiducial markers, u and v, are to each other in the reference image ($||(\mathbf{v} - \mathbf{u}) - (\mathbf{v}' - \mathbf{u}')||_2 \leq \epsilon$). After finding a matching u' and v' at the same relative position as u and v, we can find the remaining fiducial markers in the fiducial marker constellation at their expected relative locations to u' and v' in the readout image. C) We correct drifts and chromatic aberration in readout images by aligning the matched fiducial markers (red paths) to align signals from barcoded molecules (blue paths). D) We estimated the alignment error attained using fiducial marker matching registration with leave-one-out cross-validation in each of the three datasets we analyzed. Alignment accuracy had a median RMSE of 24 nm in single-channel encoded 3T3 cell culture, 29 nm in mouse cortex (for fields of view that were not discarded), and 35 nm for cross-channel encoded 3T3 cell culture.

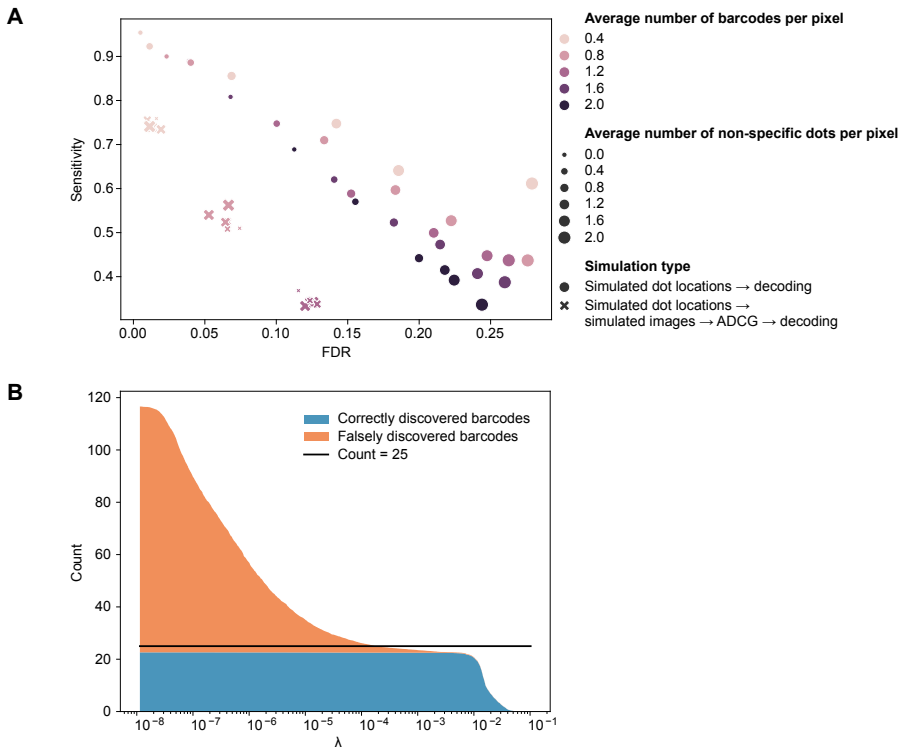

Figure D.6: A) Decoding performance on simulated seqFISH+ images and dot locations with various barcode and non-specific dot densities. Barcode and non-specific dot densities are for the entire simulated experiment, not individual images. B) Decoding by regulated regression can distinguish overlapping barcodes in simulated images. This plot shows the average performance of 100 simulation replicates of 25 barcodes in a simulated 5x5 pixel (500x500 nm) area with 2 pixels of padding in the synthetic images to ameliorate edge effects. The x-axis varies the regularization parameter, λ, in the final L0-regularized regression to choose barcodes that explain the simulated data. In these simulations, the optimal value of the regularization parameter is around 0.005. Lower regularization parameters lead to overfitting and return excessive false solutions.