Forget Pixels: Adaptive Particle Representation of Fluorescence Microscopy Images (99 Char)
Short Title:
The Adaptive Particle Representation (41 Char)

Bevan L. Cheeseman[1,2*], Ulrik Günther[1,2], Mateusz Susik[1,2], Krzysztof Gonciarz[1,2], Ivo F. Sbalzarini[1,2*]

[1] Chair of Scientific Computing for Systems Biology, Faculty of Computer Science, TU Dresden, 01069 Dresden, Germany

[2] Center for Systems Biology Dresden, Max Planck Institute of Molecular Cell Biology and Genetics, Pfotenhauerstr. 108, 01307 Dresden, Germany

*Correspondence to: ivos@mpi-cbg.de, cheesema@mpi-cbg.de

Modern microscopy modalities create a data deluge with gigabytes of data generated each second, or terabytes per day. Storing and processing these data is a severe bottleneck. We argue that this is an artifact of the images being represented on pixels. To address the root of the problem, we here propose the Adaptive Particle Representation (APR) as an image-content-aware representation of fluorescence microscopy images. The APR replaces pixel images to overcome computational and memory bottlenecks in storage and processing pipelines for studying spatiotemporal processes in biology using fluorescence microscopy. We present the ideas, concepts, and algorithms and validate them using noisy 3D image data. We show how the APR adapts to the information content of an image without reducing image quality. We then show that the

adaptivity of the APR provides orders of magnitude benefits across a range of image-processing tasks. Therefore, the APR provides a simple, extendable, and efficient content-aware representation of images that could be useful for many imaging modalities in order to relax current data and processing bottlenecks.

(less than 15000 words, 6 Figures)

Introduction New developments in fluorescence microscopy (*1–3*), labeling chemistry (*4*), and genetics (*5*) provide the potential to capture and track biological structures at high resolution in both space and time. Such data is vital for understanding many spatiotemporal processes in biology (*6*). Unfortunately, fluorescence microscopes do not directly output the shapes and locations of objects through time. Instead, they produce raw data, potentially terabytes of 3D images (*7*), from which the desired spatiotemporal information must be extracted by image processing. Handling the large image data and extracting information from the raw microscopy images presents the main bottleneck (*7–9*). We propose that at the core of the problem is not the amount of *information* contained in the images, but how the *data* encodes this information – usually as pixels on a uniform grid.

The uniform grids of pixels in the images contain information on labeled objects quantifying local intensity variations of the fluorescence signal. These local intensity variations are a measurement of the spatial localization of fluorescent molecules. Inferring information about the shape and location of labeled structures is complicated by structures of equal interest showing different imaged intensities, hence defining locally different scales of intensity variation across the image. The wide range of spatial and temporal scales in biological processes often requires imaging a large field of view at both high spatial and temporal resolution. This uniformly high resolution exacerbates the data problem and amplifies the processing bottleneck.

Data and processing bottlenecks are effectively avoided by the human visual system, which solves a similar problem of inferring object shapes and locations from photon accounts. In part, the human visual system achieves this by adaptively sampling the scene depending on its content (*10*), while adjusting to the dynamic range of intensity variations (*11*). This adaptive sampling works by selectively focusing the attention of the eyes on areas with potentially high information content (*10*). This selective focus then enables the efficient inference of information about the scene at a high effective resolution by directing the processing capacity of the retina and visual cortex. As in fluorescence microscopy, the information in different areas of a scene is not encoded in absolute intensity differences, but in relative differences compared to the local brightness. The human visual system maintains effective adaptive sampling across up to nine orders of magnitude of brightness levels (*11*) by using local gain control mechanisms that adjust to, and account for, changes in the dynamic range of intensity variations. Together, adaptation and local gain control enable the visual system to provide a high rate of information content using as little as 1 MB/s of data from the retina (*12*). In contrast, the rate of information in pixel representations of fluorescence microscopy images is much lower and is defined by the spatial and temporal resolution of the images rather than by their contents.

Inspired by the adaptive sampling and local gain control of the human visual system, we propose a novel representation of fluorescence microscopy images – the Adaptive Particle Representation (APR). The APR adaptively resamples an image, guided by local information content, while taking into account an effective local gain control. Figure 1A illustrates the basic idea of adaptive sampling. The top panel shows a pixel representation of a fluorescence image acquired from a specimen of *Danio rerio*, with labeled cell nuclei. The pixel representation places the same computational and storage costs in areas containing labeled cell nuclei and in areas with only background signal. This uniform sampling results in processing costs that are proportional to the spatial and temporal resolution of the acquisition, rather than the actual in-

formation content of the image. The main difficulty in adaptation, however, is to give equal importance to imaged structures across a wide range of intensity scales. This is achieved by local gain control as illustrated in Figure 1B. Without local gain control, adapting effectively to both bright and dim regions in the same image is not possible (*centre left*). The APR provides local gain control by guiding the adaptation by a Local Intensity Scale (*center right*). As seen in Figure 1B (*right*), this samples dim and bright objects at comparable resolution, giving them equal importance. Combining adaptive sampling and local gain control, the APR shares two key features of the human visual system to alleviate processing and storage bottlenecks in current fluorescence microscopy.

While the APR also reduces data rates, its main intention is to facilitate downstream image processing, storage, analysis, and visualization across a wide range of applications. We posit that any image representation aiming to achieve this should fulfill the following *representation criteria (RC)*:

RC1: It must guarantee a user-controllable representation accuracy for noise-free images and must not reduce the signal-to-noise ratio of noisy images.

RC2: Memory and computational cost of the representation must be proportional to the information content of an image, and independent of the number of pixels.

RC3: It must be possible to rapidly convert a given pixel image into that representation with a computational cost at most proportional to the number of input pixels.

RC4: The representation must reduce the computational and memory cost of image-processing tasks without resorting to the original pixel representation.

None of the existing multi-resolution and adaptive sampling approaches meets all of these criteria, mainly because they were developed for different applications. However, most do

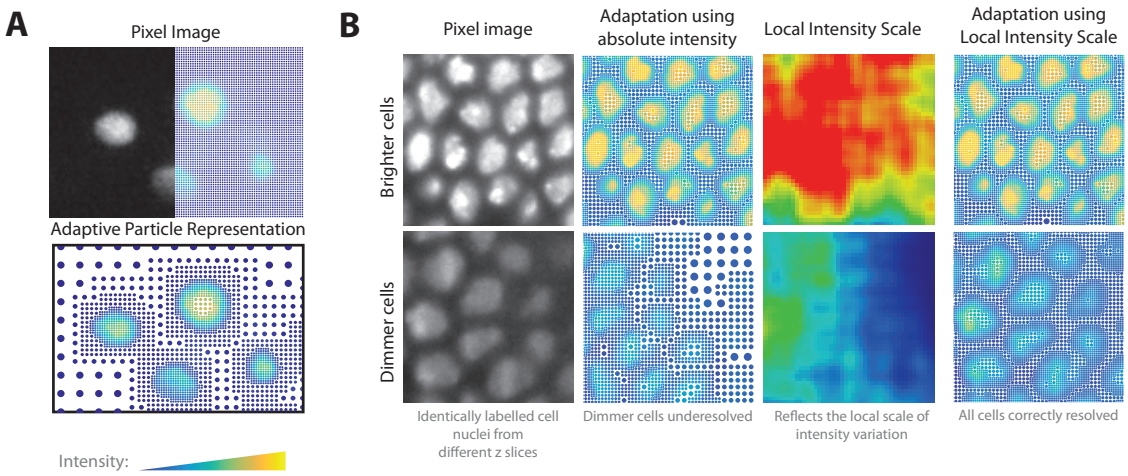

Figure 1: **Spatially adaptive representation of fluorescence microscopy images**. **A.** Example image of fluorescently labeled zebrafish cell nuclei (Dataset 7 from STable 3, courtesy of Huisken Lab, MPI-CBG & Morgridge Institute for Research), represented on a regular grid of pixels (*top*). In the right half of the image, the pixels are explicitly shown as points with color corresponding to local fluorescence intensity. The *bottom* panel shows the same image represented using the APR. Particles are shown as dots with their color indicating fluorescence intensity and their size reflecting local image structure. **B.** Adaptively representing objects of different intensity requires accounting for the local brightness levels. The panel compares two regions of labeled cell nuclei (Dataset 6 from STable 3, courtesy of Tomancak Lab, MPI-CBG) with different brightnesses (*left*). The *center left* panel shows adaptive representations based on the absolute intensity. The *right* panel shows the APR accounting for the Local Intensity Scale of the image as shown in the *center right* panel. Using the Local Intensity Scale, objects are correctly resolved across all brightness levels, without over-resolving the background.

share similar goals and use related concepts to achieve them. These approaches include super-pixels (*13, 14*), wavelet decompositions (*15–17*), error equidistribution methods (*18–20*), and dimensionality reduction (*21, 22*), as we discuss below. In contrast, we propose that the APR meets all of the above criteria. It provides a general framework combining concepts from wavelets, super-pixels, and equidistribution methods. Here, we present the APR for the first time and show that it fulfills all of the above representation criteria, making it an ideal candidate to replace pixel images in fluorescence microscopy.

1 The Adaptive Particle Representation

We illustrate the ideas and concepts behind the APR using a $1D$ Gaussian function as a didactic example. All of the concepts introduced extend to higher dimensions and to general continuous functions, as shown in the Supplement.

The APR takes an input pixel image and resamples it in a spatially adaptive way, representing it as a set of Particle Cells $\mathcal{V}$ and intensity values stored at particle locations $\mathcal{P}$. Particles, a generalization of pixels, are collocation points in space that carry properties, such as intensity, but are not restricted to sit on an evenly spaced grid and may have different sizes. The Particle Cells partition space and implicitly define the particle locations and a piecewise constant Implied Resolution Function $R^*(y)$ at all locations y in the image. Importantly, the Implied Resolution Function $R^*(y)$ also defines how the image is to be reconstructed at off-pixel and off-particle locations. It defines neighbor interactions between particles and specifies a local minimum resolution. From the APR, an image can be reconstructed at any location y by taking a non-negative weighted combination of particles that are within $R^*(y)$ distance of y. Formulated in this way, a pixel image is a set of particles placed on a regular grid with a constant Implied Resolution Function $R^*(y) = h$, where h is the pixel size (as shown in Figure 2A, *right*). In contrast to uniform pixel representations, the APR adaptively represents an original

image with particles whose locations, density, and sizes vary across the image (Figure 2A, *left*). As a result, computational and storage costs scale with the number of particles and no longer with the number of pixels. Therefore, reducing the number of particles by adjusting the resolution to the image content, the APR can reduce storage and computational costs and increase the information-per-data ratio. This is optimally achieved by minimizing the number of particles required to represent a given image. However, the APR and any results computed from it must still reflect the content of the original pixel image.

1.1 Reconstruction Condition

For the APR to optimally represent a given image, the Implied Resolution Function should be set as large as possible at every location, while still guaranteeing that the image can be reconstructed within a user-specified relative error E scaled by the Local Intensity Scale $\sigma(y)$. The Local Intensity Scale $\sigma(y)$ is an estimate of the range of intensities present locally in the image. Considering an arbitrary Resolution Function $R(y)$, we can formulate the problem as finding the largest $R(y)$ everywhere that satisfies

$$|I(y) - \hat{I}(y)| \leq E\sigma(y), \tag{1}$$

where $\hat{I}(y)$ is the reconstructed intensity calculated by a non-negative weighted average over particles within $R(y)$ distance of y. We call this the *Reconstruction Condition* and illustrate it in Figure 2B. For the 1D example shown in Figure 2, a constant local intensity scale $\sigma(y) = 1$ is used. We therefore focus solely on maximizing $R(y)$ at each location. Maximizing $R(y)$ minimizes $\frac{1}{R(y)}$, which is proportional to the locally required sampling density. Therefore, maximizing $R(y)$ results in the minimum number of particles used. Unfortunately, finding the optimal $R(y)$ that satisfies the Reconstruction Condition for arbitrary images requires a number of compute operations that scales with the square of the number of pixels N. This computational

cost is prohibitive even for modestly sized images. We therefore propose two conservative restrictions on the problem and show that the optimal solution to the restricted problem can be computed with a total number of operations that is proportional to N.

1.2 APR Solution

Next, we outline the two problem restrictions, and how they are used to formulate an efficient linear-time algorithm for creating the APR.

Resolution Bound The first restriction on the Resolution Function $R(y)$ requires that for all original pixel locations y it satisfies the inequality

$$R(y) \leq L(y^*), \tag{2}$$

where $|y - y^*| \leq R(y)$, and $L(y) = \frac{E\sigma(y)}{|\nabla I|}$. Here $|\nabla I|$ is the magnitude of the image intensity gradient, which in 1D is $|\frac{\mathrm{d}I}{\mathrm{d}y}|$ and can be computed directly from the image. We call this inequality the Resolution Bound, and $L(y)$ the Local Resolution Estimate. If we assume the continuous intensity distribution underlying the image to be differentiable everywhere and the Local Intensity Scale $\sigma(y)$ to be sufficiently smooth (See SuppMat 2.1 for 1D and SuppMatEq 19 for nD), satisfying the Resolution Bound guarantees satisfying the Reconstruction Condition (See SuppMat 2 for 1D, and SuppMat 3 for nD). In Figure 2C, we illustrate that the Resolution Bound in 1D requires that a box centered at y of height $R(y)$ and width $2R(y)$ does not intersect anywhere with the graph of $L(y)$. Since the Resolution Bound represents a tighter bound than the Reconstruction Condition, the optimal solution to the Resolution Bound $R_b(y)$ is always less than or equal to the optimal solution to the Reconstruction Condition $R_c(y)$, therefore providing the same or a higher image representation accuracy. The dashed lines in Figure 2D illustrates this for the 1D example. As mentioned above, solving for the optimal Resolution Function has a worst-case complexity in $O(N^2)$. However, we show next that the Resolution Bound can

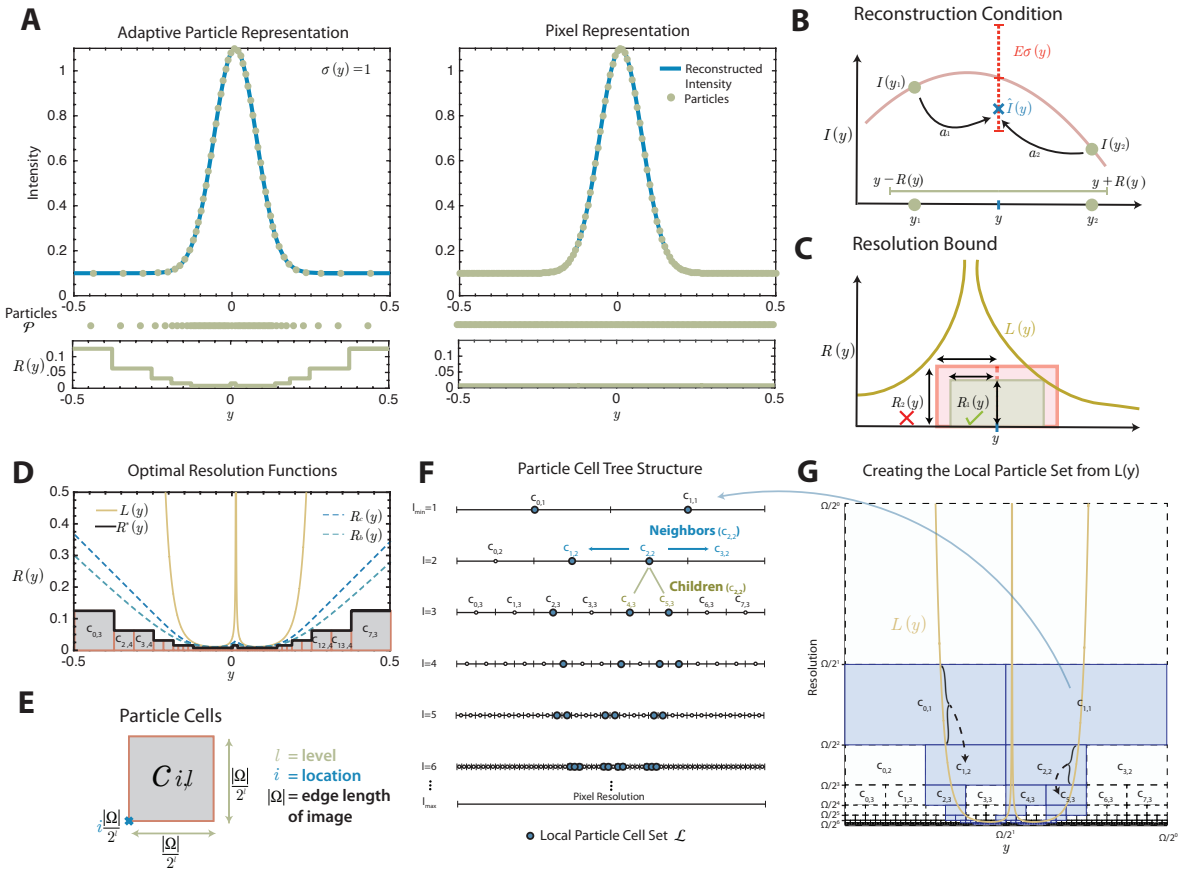

Figure 2: **Concepts and definitions of the Adaptive Particle Representation (APR) illustrated in 1D**. See main text for explanations. **A.** APR (*left*, $E = 0.1$, $\sigma(y) = 1$) and uniform pixel (*right*, $h = 0.0078$) representation of the shifted 1D Gaussian $I(y) = \exp\left[\frac{-(y-0.01)^2}{0.009}\right] + 0.1$. The *bottom* plots show the corresponding Resolution Functions $R(y)$ with the set of particles $\mathcal{P}$ shown as dots above. **B.** Illustration of the Reconstruction Condition, requiring that for all original pixel locations y, any non-negative weighted average of the particles (green dots) within $R(y)$ distance of y reconstructs an intensity value with a deviation less than $E\sigma(y)$ (red dashed interval). **C.** Illustration of the Resolution Bound, requiring for all locations y that a rectangle centered at y with width $2R(y)$ and height $R(y)$ does not intersect the curve of the Local Resolution Estimate $L(y)$. For the choices shown in the figure, fulfilling the Resolution Bound guarantees fulfilling the Reconstruction Condition, given assumptions on σ. **D.** Comparison of the optimal (largest everywhere) Resolution Function satisfying the Reconstruction Condition $R_c(y)$ (blue dashed) with the optimal $R_b(y)$ satisfying additionally also the Resolution Bound (green dashed) and with the optimal Implied Resolution Function $R^*(y)$ (bold black) for the 1D Gaussian example from **A**. The Implied Resolution Function is composed of blocks called Particle Cells (gray). They never intersect the optimal Resolution Function (R_b), therefore providing a conservative approximation. **E.** Definition of a 1D Particle Cell as described by its level l and location i. **F.** The set of all possible Particle Cells can be represented as a binary tree reaching down to single-pixel resolution. **G.** The Local Particle Cell set $\mathcal{L}$ is constructed from $L(y)$. The link between sections of $L(y)$ and a Particle Cells in $\mathcal{L}$ are shown in with braces and dotted lines. All possible Particle Cells are shown as blocks and those belonging to $\mathcal{L}$ are shaded blue ($\Omega = |\Omega|$ in labeled axis for brevity).

be found optimally with a linear complexity in N if we restrict the Resolution Function to be composed of square blocks.

Finding the Resolution Function with Particle cells The second restriction is that the blocks constituting the Resolution Function must have edge lengths that are powers of 1/2 of the image edge length. The piecewise constant Resolution Function defined by the uppermost edges of these blocks is called the *Implied Resolution Function* $R^*(y)$ and is shown in black in Figure 2D. The blocks we call *Particle Cells*. They have sides of length $\frac{|\Omega|}{2^l}$, where $|\Omega|$ is the edge length of the image, measured in pixels. The number l is a positive integer we call the *Particle Cell Level*. Each Particle Cell $c_{i,l}$, is therefore uniquely determined by its level l and location i. Figure 2E illustrates these definitions for a single Particle Cell (See SuppMat 4 for the nD formal definition). The size of the blocks on the lowest level is half the size of the image ($l_{min} = 1$), and the highest level of resolution l_{max} contains boxes the size of the original pixels. For image edge lengths that are not powers of 2, $|\Omega|$ is rounded upwards to the nearest power of two.

Using these two restrictions, the problem of finding the optimal Resolution Function can be reduced to finding the smallest set $\mathcal{V}$ of blocks that defines an Implied Resolution Function $R^*(y)$ that satisfies the Resolution Bound (SuppMat 4.1). We call this set $\mathcal{V}$ of Particle Cells the *Optimal Valid Particle Cell* (OVPC) set.

In order to construct an algorithm that efficiently finds the OVPC set for a given Local Resolution Estimate $L(y)$, we first formulate the Resolution Bound in terms of Particle Cells. This formulation requires arranging the set $\mathcal{C}$ of all possible Particle Cells $c_{i,l}$ by level l and location i in a tree structure, as shown in Figure 2F. In 1D this is a binary tree, in 2D a quad-tree, and in 3D an oct-tree. When arranged as a tree structure, we can naturally define children and neighbor relationships between Particle Cells, as respectively shown in green and blue in

the example. Also, we define the descendants of a Particle Cell as the set of all children and children's children up to the maximum resolution level l_{max}. Given these definitions, the Local Resolution Estimate $L(y)$ can be represented as a set of Particle Cells $\mathcal{L}$ by iterating over each pixel y^*, and adding the Particle Cell with level $l = \lceil \log_2 \frac{|\Omega|}{L(y)} \rceil$ and location $i = \lfloor \frac{2^l y^*}{|\Omega|} \rfloor$ to $\mathcal{L}$ if it is not already in $\mathcal{L}$ (assuming the lower-left boundary of the image is at zero). Figure 2G illustrates how $\mathcal{L}$ relates to $L(y)$, with $\mathcal{L}$ also represented in Figure 2F in the tree structure. We call this set of Particle Cells the *Local Particle Cell* (LPC) set $\mathcal{L}$ (See SuppMat 4.2).

We can then represent the Resolution Bound in terms of $\mathcal{L}$. A set of Particle Cells $\mathcal{V}$ will define an Implied Resolution Function that satisfies the Resolution Bound for $L(y)$, if and only if the following statement is true: *for every Particle Cell in $\mathcal{V}$, none of its descendants, or neighbors' descendants, are in the LPC set $\mathcal{L}$* (SuppTheorem 1). We call any set of Particle Cells satisfying this statement *valid*. The OVPC set $\mathcal{V}$ is then defined as the valid set for which replacing any combination of Particle Cells with larger Particle Cells would result in $\mathcal{V}$ no longer being valid (SuppTheorem 2).

Pulling Scheme We present an efficient algorithm for finding the OVPC set $\mathcal{V}$ called the *Pulling Scheme*. The name is motivated by the way a single Particle Cell in $\mathcal{L}$ *pulls* the resolution function down to enforce smaller Particle Cells across the image. The Pulling Scheme finds the OVPC set $\mathcal{V}$ directly, without explicitly checking for validity or optimality. The result is by construction guaranteed to be valid and optimal. In order to derive the algorithm, we leverage three properties of OVPC sets:

1. *Predictable and self-similar structure*: Neighboring Particle Cells never differ by more than by one level and are arranged in a fixed pattern around the smallest Particle Cells in the set. This local structure is independent of absolute level l and endows the set with a self-similar structure. Using this structural feature, the OVPC set $\mathcal{V}$ for a LPC set $\mathcal{L}$ with

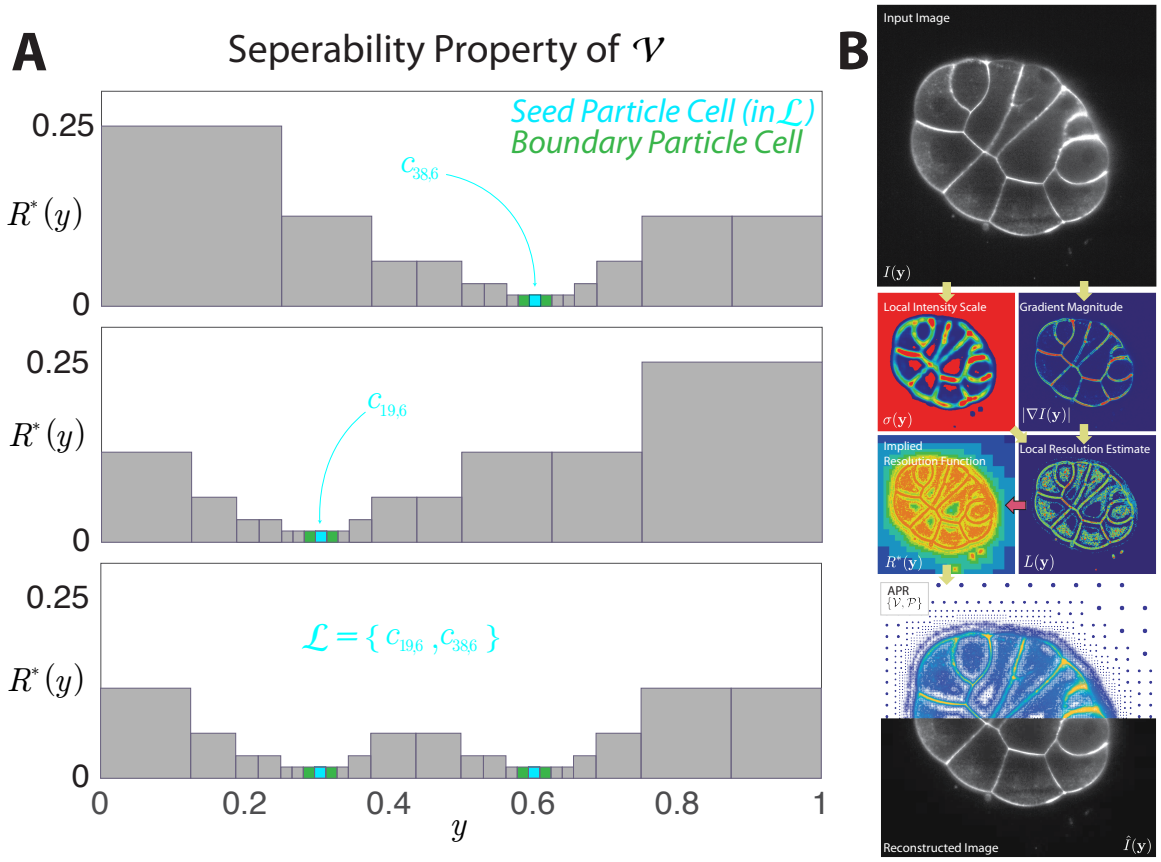

Figure 3: **Schematic and illustration of the Separability property, Pulling Scheme (Algorithm 1), and 3D processing pipeline. A.** The Pulling Scheme computes the local Optimal Valid Particle Cell (OVPC) set $\mathcal{V}$ for a given Local Particle Cell (LPC) set $\mathcal{L}$. Due to the separability property (see main text), this can be done separately for each Local Particle Cell (*top* and *middle*). The complete result for the combined set is then formed by taking the smallest Particle Cell at each location (*bottom*). **B.** Illustration of the steps for creating the APR of an example 2D fluorescence image (Dataset 10 in STable 3, courtesy of Lemaire lab, CRBM (CNRS) and Hufnagel lab, EMBL). First the Local Intensity Scale $\sigma(\mathbf{y})$ and the gradient magnitude $|\nabla I(\mathbf{y})|$ are calculated. These two are then combined to compute the Local Resolution Estimate $L(\mathbf{y})$. The Pulling Scheme (*red* arrow) then uses $L(y)$ to compute the optimal Implied Resolution Function $R^*(\mathbf{y})$. This is then used to define the OVPC set $\mathcal{V}$ and the particle locations $\mathcal{P}$, which generate the APR (*bottom* panel). The top half of the *bottom* panel shows the particles of the APR with color encoding intensity. The bottom half shows a piecewise constant reconstruction $\hat{I}(y)$ of the image for visualization.

only one Particle Cell $c_{i,l}$ can be generated directly for any i and l.

2. *Separability*: We can find the OVPC set given a LPC set $\mathcal{L}$ by considering each cell in $\mathcal{L}$ separately and then combining the smallest Particle Cells from all sets that cover the image (see SuppLemma 1). Figure 3A illustrates this separability property.

3. *Redundancy*: The redundancy property tells us that when constructing $\mathcal{V}$ we can ignore all Particle Cells in $\mathcal{L}$ that have descendants in $\mathcal{L}$. This is because descendants provide equal or tighter constraints on the resolution function than their parent Particle Cells (see SuppLemma. 2 for the proof).

These properties enable us to efficiently construct $\mathcal{V}$ by propagating solutions from individual Particle Cells in $\mathcal{L}$, one level at a time, starting from the highest level (l_{max}) of the smallest Particle Cells in $\mathcal{L}$. Here we use a simple implementation that explicitly represents all possible Particle Cells in an image pyramid structure[1]. The *Pulling Schemne* is summarized in Algorithm 1, and Figure 3B illustrates the steps for each level. SuppMat 5.5 and SuppMat 12.5 provide additional details. The computational complexity of the neighbor operations in the algorithm scales with the number of Particle Cells in $\mathcal{V}$ and guarantees validity and optimality by construction. Computing the OVPC set $\mathcal{V}$ using the Pulling Scheme incurs a computational cost that is at most proportional to the number of pixels N.

The computational and memory performance of the Pulling Scheme is reduced by a factor of 2^d, where d is the image dimensionality, while obtaining the same solution and algorithm as above, using the Equivalence Optimization (See SuppMat 5.4 and SuppMat 5.7). This restricts calculations on the full image to filtering operations for the gradient magnitude and greatly improves memory and computational efficiency of the method. A second optimization restricts the neighborhood of particle cells to further reduce computational cost, as described in Supp-

[1] Alternative implementations are possible that do not require the explicit storage of the full tree structure.

Mat 5.6. We use both optimizations for results presented in this paper.

Placing the Particles $\mathcal{P}$ Given the Implied Resolution Function computed by the Pulling Scheme, the last step of forming the APR is to determine the locations of the particles $\mathcal{P}$. This must be done such that around each pixel location y there is at least one particle within a distance of $R^*(y)$. This is most easily achieved by placing one particle at the center of each Particle Cell in $\mathcal{V}$. Specifically, for each Particle Cell $c_{i,l}$ in $\mathcal{V}$, we add a particle p to $\mathcal{P}$ with location $y_p = \frac{|\Omega|}{2^l}(i + 0.5)$. For each particle p we store the image intensity at that location $I_p = I(y_p)$ interpolated from the original pixels as described in SuppMat 6. This way of arranging the particles has the advantage that the particle positions do not need to be stored, as they can be directly computed from $\mathcal{V}$.

Although simple, this sampling is optimal for a given Implied Resolution Function, in the sense that the number of particles is equal to the integral of the minimally required particle density over the whole image. The required particle density at y is given by $1/R^*(y)$. Hence, in addition to providing an optimal Implied Resolution Function, the APR also uses the smallest number of particles on average (see SuppMat 6.1).

Forming the APR=$\{\mathcal{V}, \mathcal{P}\}$ In Figure 3B we outline the steps required to form the APR from an input image. The APR can be stored as the combination of $\{\mathcal{V}, \mathcal{P}\}$. We represent the OVPC set $\mathcal{V}$ by storing the integer level l and the integer location i for each Particle Cell. $\mathcal{V}$ then defines the Implied Resolution Function $R^*(y)$ for all y in the image. The second component, the particle set $\mathcal{P}$, stores the properties of each particle p, i.e., its intensity and type. Since the particle positions do not need to be stored, the APR is efficiently represented in memory.

Practical Considerations Determining L(y) requires computing the intensity gradient ∇I over the input image. In practice, the pixel intensities are noisy, which leads to uncertainty in

Data: Local Particle Cell set $\mathcal{L}$
Result: Optimal Valid Particle Cell set $\mathcal{V}(\mathcal{L})$

```
Function pulling_scheme(L)
    Represent all possible Particle Cells C from l_max to l_min in a multi-resolution pyramid
      and set all Particle Cells type to EMPTY;
    forall Particle Cells c ∈ C where c ∈ L do
        c.type = SEED
    end
    for l_c = l_max : l_min do
        /* Fill neighbors (Step 1)                                          */
        forall neighbors n of c ∈ C(l_c) where c.type is (SEED or PROPAGATE) do
            if n.type is EMPTY then
                n.type = BOUNDARY
        end
        /* Set Parents (Step 2)                                             */
        forall parents p of c ∈ C(l_c) where c.type is (SEED, PROPAGATE, or
          ASCENDANT) do
            p.type = ASCENDANT
        end
        if l_c > l_min then
            /* Set Ascendant Neighbors (Step 3)                             */
            forall neighbors n of c ∈ C(l_c − 1) where c.type is ASCENDANT do
                if n.type is EMPTY then
                    n.type = ASCENDANT_NEIGHBOR
                if n.type is SEED then
                    n.type = PROPAGATE
            end
            /* Set Fillers (Step 4)                                         */
            forall children d of c ∈ C(l_c − 1) where c.type is (ASCENDANT_NEIGH or
              PROPAGATE) do
                if (d.type is EMPTY then
                    d.type = FILLER
            end
    end
    return all type SEED, BOUNDARY and FILLER Particle Cells in C as V;
```

Algorithm 1: The Pulling Scheme algorithm. The Pulling Scheme efficiently computes the OVPC set $\mathcal{V}$ from the Local Particle Cell set $\mathcal{L}$ using a temporary pyramid mesh data structure. $\mathcal{C}(l)$ denotes all Particle Cells on level l. See SuppMatAlgorithm 1 for detailed pseudocode, and SFigure 6 for a schematic of the main steps.

the computed L(y). In SuppMat 7, we provide theoretical results how this uncertainty imposes a lower bound on the achievable E. However, errors in $L(y)$ can be compensated for by increasing E (SuppMat 7.2). Moreover, the gradient estimate converges at the optimal statistical rate (SuppMat 7.4). In the next section, we confirm these theoretical results by direct benchmarking of the APR under noisy and noise-free conditions.

Data structures Appropriate data structures must be used to efficiently store and process on the APR. Ideally, these structures allow direct memory access at low overhead. Here, we propose a multi-level data structure for the APR, as described in SuppMat 17. Each APR level is encoded similar to sparse matrix schemes. These data structures efficiently encode $\mathcal{V}$ and $\mathcal{P}$ by explicitly encoding only one spatial coordinate per Particle Cell, while allowing random access. We call this data structure the *Sparse APR* (SA). It relies on storing one red-black tree per x, z and level, caching access information for contiguous blocks of Particle Cells. When storing image intensity using 16 bits, the SA data structure requires approximately 50% more memory than the intensities alone.

APR image file format We store the APR using the HDF5 file format (*23*) and the BLOSC HDF5 plugin (*24*) for lossless Zstd compression of the Particle Cell and intensity data.

Summary The APR resamples an image by optimally adapting a set of particles to the content of the image. This results in an image representation that has a computational and memory cost that scales with image content, while guaranteeing a representation error below a user-defined threshold E (RC1, RC2). Using the Particle Cell formulation and Pulling Scheme, the APR can be formed rapidly and efficiently, scaling to large 3D images and extending to arbitrary dimensions (RC3).

3D Fluorescence APR Implementation

We assess the properties of the APR for noisy 3D images. We do this by first outlining a specific 3D implementation and then benchmarking the APR using synthetic data. Illustrative results in 1D are given in SuppMat 11 (code available from github.com/cheesema/APR_1D_demo). Figure 3B illustrates the main steps of the implementation using a 2D example.

When implementing the APR, three design choices have to be made. First, one has to decide how to calculate the gradient magnitude $|\nabla I(\mathbf{y})|$. Second, one has to decide how to compute the Local Intensity Scale $\sigma(\mathbf{y})$. Third, one has to decide how to interpolate the image intensity at particle locations $I_p = I(\mathbf{y}_p)$. We describe our choices briefly below, with full details and descriptions of parameters given in SuppMat 12. All design decisions are made to optimize robustness against imaging noise and computational efficiency.

To calculate the gradient magnitude over the input image we use smoothing cubic B-Splines (*25*), as they provide robust gradient estimation in the presence of noise. They require the setting a smoothing parameter λ depending on the noise level. Using a recursive implementation (*25*), however, renders the computational cost independent of the choice of λ.

For the Local Intensity Scale $\sigma(\mathbf{y})$, we use a smooth estimate of the local dynamic range of the image, as described in SuppMat 12.3. Examples are shown in Figures 1B and 3B, and a schematic in SFigure 19. The size of the smoothing window is set by a coarse estimate of the standard deviation of the point-spread function (PSF) of the microscope. Further, a minimum threshold is introduced to prevent resolving background noise (SFigure 21).

We find that the method is relatively insensitive to the choice of these parameters and a discussion on parameter selection for real datasets and their impact on the APR is given in Suppmat 13 and parameters used for real datasets are given in STable 3.

This form of the local intensity scale accounts for variations in the intensities of labeled

objects, similar to gain control in the human visual systems. We ensure that σ is sufficiently smooth (See SuppMat 4.4) by computing it over the image downsampled by a factor of two.

Two methods are combined to interpolate pixel intensities to particle locations. For particles in Particle Cells at pixel resolution, the intensities are directly copied from the respective pixels. For particles in larger particle cells, we assign the average intensity of all pixels in that Particle Cell (*26*).

Because we intend to benchmark the APR by comparing with pixel images, we also need a method to reconstruct a pixel image from an APR. Note that this is only done for benchmarking. In real applications, all downstream processing and visualization happens directly in the APR without ever going back to pixels. As described above, a pixel image satisfying the Reconstruction Bound can be reconstructed from the APR using any non-negative weighted average of particles within $R^*(y)$ of pixel y. In SuppMat 10 we discuss possible weight choices, providing examples of smooth, piecewise constant, and worst-case reconstructions. The worst-case reconstruction produces the worst point-wise error of all reconstruction methods and is therefore useful for some benchmarking tasks, but not for practical use. For displaying figures, and benchmarking, unless otherwise stated, we use the piecewise constant reconstruction. This reconstruction sets the pixels inside every Particle Cell equal to the intensity of the particle in that cell and thus has the best computational efficiency, and surprising visual quality. SFigure 23 provides an example of the comparison of a smooth and piecewise constant reconstruction with an original image.

Validation

All benchmarks use the open-source C++ APR software library *libAPR* (github.com/cheesema/LibAPR) compiled with with gcc 5.4.0 and OpenMP shared-memory parallelism on a 10-core Intel Xeon E5-2660 v3 (25 MB cache, 2.60 GHz, 64 GB RAM) running Ubuntu Linux 16.04. SuppMat 15

provides a detailed description of each benchmark and the parameters used.

Benchmarks on synthetic data

We first assess the performance of the APR using synthetic benchmark data. SuppMat 14 and SFigure 22 outline the synthetic data generation pipeline. The key advantage of synthetic data is that all relevant image parameters can be varied and the ground-truth image is known. Synthetic images are generated by placing a number of blurred objects into the image domain and corrupting with modulatory Poisson noise. We study the influence of image size, content, and noise level on the performance of the APR. Spherical objects are used for simplicity unless otherwise indicated.

Reconstruction Bound We experimentally confirm that the APR satisfies the Reconstruction Condition in the absence of noise, illustrating the theoretical results presented above. Figure 4A shows the empirical relative error $E^* = |I(\mathbf{y}) - \hat{I}(\mathbf{y})|_\infty$ for increasing imposed error bounds E. In all cases, $E^* < E$ (dashed line), as required by the Reconstruction Condition. As expected, we find that the number of particles used by the APR to represent the image decreases with increasing E (right axis). The results are unchanged when using more complex objects than spheres (SFigure 25) and using different reconstruction methods (SFigure 25). Figure 4C provides examples of the quality of APR reconstruction at different levels of E compared to ground truth. Hence, in the absence of noise, the APR satisfies the Reconstruction Condition everywhere, guaranteeing a reconstruction error below the user-specified threshold, hence fulfilling the first part of RC1.

Robustness against noise In real applications, images are corrupted by noise. We find that the introduction of noise introduces a lower limit on the relative error E^* that can be achieved (see first plot in SFigure 26). This observation agrees with theoretical analysis (SuppMat 7.3).

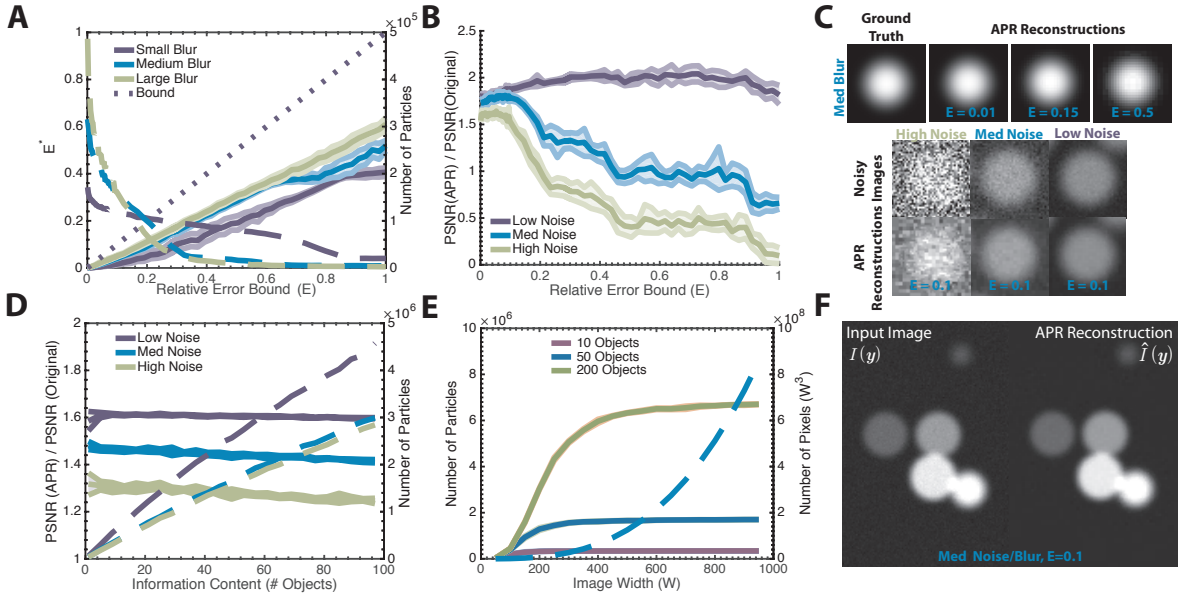

Figure 4: **Benchmarking the APR on synthetic data**. All results are shown as mean (*lines*) and standard deviation (*bands*). **A.** Effectively observed reconstruction error E^* (solid lines, *left axis*) between the ground truth and the piecewise constant APR reconstruction (SuppMat. 15.2) for noise-free images. Number of particles used by the APR (dashed lines, *right axis*) for different user-defined error thresholds E. Results are shown for images of different sharpness (*inset legend*). The APR reconstruction error is below the specified threshold in all cases. More accurate APRs require more particles. **B.** Peak signal-to-noise ratio (PSNR) of the APR relative to the PSNR of the original pixel image for different error thresholds E and image noise levels (inset legend) (SuppMat. 15.3). For low E and noisy images, the APR has a better PSNR than the input images. **C.** Examples of test images of spherical objects with different noise level and E used in the benchmarks. The *top* row shows the APR reconstruction of the medium-blur noise-free test image at different E compared to the ground truth. The *bottom* rows compare the original image with the APR reconstructions of noisy images for $E = 0.1$ and illustrate the inherent denoising property of the APR. **D.** PSNR ratio (solid lines, *left axis* and number of particles used (dashed lines, *right axis*) for images containing different numbers of objects, i.e., different information content, for $E = 0.1$. (SuppMat. 15.4). In all cases, the PSNR of the APR is better than that of the input image, and the number of particles scales at most linearly with image information content. **E.** Number of APR particles (solid line, *left axis* and input image pixel (dashed line, *right axis*) for images of different width W containing a fixed number of objects (SuppMat. 15.5). The number of particles of the APR plateaus once the objects in the image are well resolved. **F.** Visual comparison of a medium-blur, medium-noise image I containing six objects (*left*) with its APR reconstruction $\hat{I}$ (*right*) for $E = 0.1$.

This lower bound is entirely due to the noise in the pixel intensity values, while the adaptation of the Implied Resolution Function $R^*(y)$ is robust to noise. This is demonstrated in the second plot in SFigure 26, where noisy particle intensities are replaced with ground-truth values for the reconstruction step. Adaptation is still done on the noisy pixel data. Again, E^* can be made arbitrarily small, indicating that the construction of the APR is robust against imaging noise. This result also agrees with the theoretical analysis of the impact of errors in $L(y)$ on the Implied Resolution Function (SuppMat 7.2).

To understand how to best set E in the presence of noise, we compute the observed peak signal-to-noise ratio (PSNR) of the reconstructed image and compare with the PSNR of the original image. Figure 4B shows that decreasing E to zero does not maximize the PSNR. Instead, for medium to high quality input images, the PSNR is highest between an E of 0.08 and 0.15. For low-quality input images, we find a monotonic relationship between the PSNR and E, as de-noising from downsampling dominates. Also, for $E < 0.2$ the reconstruction error is always less than the noise in the input image, reflected in a PSNR ratio greater than one. Therefore, for noisy images with medium to high quality, there is an optimal range for E between 0.08 and 0.15. In this range, the reconstruction errors are less than the imaging noise, and the signal-to-noise ratio of the APR is better than that of the input pixel image, fulfilling also the second part of RC1.

Response to image content In Figure 4D we show how the APR adapts to image content. This adaptation is manifested in the linear relationship between the number of objects (spheres) randomly placed in the image and the number of particles used by the APR (*right axis*). Adaption is linear despite the brightness of the objects randomly varying over an order of magnitude (see SuppMat 15.4). Image quality is maintained throughout (*left axis*). Figure 4C shows an example of a medium-quality input image and its APR reconstruction. Figure 4E shows that

the number of particles used by the APR to represent a fixed number of objects becomes independent of image size. Also, if pixel resolution and image size are increased proportionally, the APR approaches a constant number of particles (SFigure 27). Together, these results show that the APR adapts proportional to image content, independent of the number of pixels. hence fulfilling RC2.

Evaluation of the Local Intensity Scale So far, we have not directly assessed the validity of the Local Intensity Scale σ. In order to do this, we need a ground-truth reference. In SuppMat 14.5 we introduce the *perfect APR*, and the *Ideal Local Intensity Scale* σ^{ideal} that can be calculated for synthetic data. This ground-truth representation is then used to benchmark the APR.

The results in STables 1 and 2 show that the local intensity scale we use is effective over wide range of scenarios. However, for crowded images with large contrast variations (two orders of magnitude or more), we find that the Local Intensity Scale over-estimates the dynamic range of dim regions that are close to bright regions. This effect is most pronounced in high-quality images, where alternative formulations of the Local Intensity Scale could provide better results.

Computational cost Due to the adaptivity of the APR, its computational cost depends on image content through the number of particles, and not on the input image size N. For a given input image, we define the *Computational Ratio* (CR) as:

$$\text{CR} = \frac{\text{number of input pixels}}{\text{number of output particles}}. \tag{3}$$

We assess the performance of the APR for synthetic images with numbers of objects roughly corresponding to CR = 5, 20, 100, representing high, medium, and low complexity images (SFigure 28, SuppMat 16.1). The results are given in Table 1. The APR achieved effective CR values of 5.63, 19.7, and 93.9, respectively.

	Computational Ratio (CR)	Raw Image Size (GB)	Compressed APR (GB)	MCR (APR)	MCR (APR-WNL)	MCR (Pixels pbzip2)	Pipeline Time (s)	Pulling Scheme Runtime (s)
CR5	5.63 (0.02)	1.024	0.129 (0.0006)	7.9 (0.04)	15.95 (0.07)	2.03 (0.001)	2.34 (0.086)	0.104 (0.002)
CR20	19.7 (0.13)	1.024	0.036 (.0002)	28.4 (0.19)	54.4 (0.36)	2.072 (0.0001)	2.01 (0.07)	0.04 (0.003)
CR100	93.9 (1.6)	1.024	0.007 (0.0001)	139.9 (2.1)	265.2 (3.4)	2.092 (0.007)	1.87 (0.08)	0.027 (0.005)
Exemplars Mean	51.1 (89.3)	1.869 (1.38)	0.051 (0.053)	129.5 (284)	191.4 (347)	32.306 (87.2)	3.65 (2.19)	0.10 (0.08)
Exemplars Median	22.7	1.258	0.027	36.8	79.8	3.061	2.19	0.066

Table 1: **Summary statistics of the APR benchmarks on synthetic and real-world images**. Results are shown for synthetic images with fixed CR=5,20,100 and for 19 real-world exemplar datasets (see STable 3). For the exemplars, we report the means, standard deviation (brackets), and medians of the values over all exemplar images. For the synthetic fixed-CR benchmarks, the effective CR and the Memory Compression Ratios (MCR) are averaged over image sizes from 200^3 to 1000^3 and the values for absolute runtimes and storage requirements are given for images of size 800^3. For comparison, we also report the MCR using *within-noise-level* (WNL) compression (*27*) of the APR and the size of the losslessly compressed pixel images using *pbzip2*. We also show the time taken to transform the images to the APR on the benchmark machine, and the runtime of the Pulling Scheme alone.

Benchmarks on real data

We present results for a corpus of 19 exemplar volumetric fluorescence microscopy datasets of different content, size, and imaging modality. The datasets are described in STable 3. The APR parameters used are given in STable 4 and discussed in SuppMat 16.2. The exemplar images range in size from 160 MB to 4GB. SFigure 29 shows a cross-section of the APR for exemplar dataset 7 of labeled cell nuclei in a developing Zebrafish. Summary statistics for the exemplar datasets are given in Table 1. SVideo 1 illustrates the adaptation of exemplar dataset 1 by Particle Cell level and compares a piecewise constant reconstruction with the original image.

Memory requirements Calculation of the APR from an image requires approximately 2.7 times (for 16-bit images) the size of the original image in memory. Further, the maximum size is only limited by available main memory (RAM) of the machine and by the ability to globally index the particles using an unsigned 64-bit integer. Our pipeline has been successfully tested on datasets exceeding 100 GB.

Execution time In SuppMat 18 we provide more detailed analysis of the time taken to produce the APR for the exemplar datasets. On our benchmark system, we find linear scaling in N and an average datarate of 507 MBs per second for processing. This rate corresponds to taking 3.9 seconds to form the APR from an input image of size $N = 1000^3$. In STable 3 we provide execution times of the exemplar datasets. The execution times range from 0.37 seconds to 8.14 seconds, with an average of 3.65 seconds. Table 1 summarizes the results. The pipeline can be further accelerated using additional CPU cores showing efficient parallel scaling (Amdahl's Law, parallel fraction = 0.95) on up to 47 cores, achieving a data rates of up to 1400 MB/second (SFigure 31). This enables real-time conversion of images to the APR, as it is faster than the acquisition rate of the microscope (*28, 29*). The computation of the gradient magnitude using smoothing B-splines dominates the execution time, taking up to 59% of the total time (STable 4). In contrast, determining the Implied Resolution Function using the Pulling Scheme on average takes less than 3.5% of the total time (STable 4). The relatively high cost of the pixel filter operations is not a consequence of expensive filters, as all filters are simple and use efficient implementations. Instead, it is a reflection of the low cost of the Pulling Scheme and use of the Equivalence Optimization. Given that the Pulling Scheme is the novel algorithmic contribution here, we provide additional benchmarks in SuppMat 18.2. We confirm the worst-case linear scaling in N (SFigure 32) and find on average sub-linear scaling as the size of the image N and the size of the Local Particle Cell set $\mathcal{L}$ are varied.

We conclude that images can be rapidly converted into the APR with a cost that scales at most linearly with image size N, fulfilling RC3.

Storage requirements For the fixed-CR datasets, we observe an average Memory Compression Ratio (MCR) = (Size of the input image in Bytes)/(Size of the compressed APR in Bytes) of 1.4 times the CR. STable 3 gives the MCR for the exemplar datasets. The median MCR of

the exemplars is 36.8, and the mean is 129.5. This corresponds to an average size of the input images of 1.87 GB and the compressed APR of 51 MB. Table 1 summarizes the results, additionally showing MCR for pixel images stored using lossless pbzip2 compression. In the APR files, on average 89% of the Bytes are used to store the particle intensities. Implying that the overhead introduced by the APR data structures is 11% on average. In the limiting case where the number of particles is equal to the number of input pixels, the particle intensities account for 99.99% of the storage, indicating that the APR adds virtually no overhead in this case. These compression ratios are comparable to custom lossy compression methods designed specifically for storing of fluorescence microscopy images (*27, 30*). Additionally, the APR particle intensities can be further compressed in a lossy manner. As an example, in Table 1 we also give the MCR using the *within-noise-level* (WNL) compression algorithm (*27*) applied to particles at pixel resolution, achieving an additional 1.4. . .4 compression factor. Hence, the APR can be efficiently compressed with a file size proportional to the image content, fulfilling RC2. Unlike compression techniques, however, the APR is an image *representation* that can be leveraged in downstream processing tasks without going back to the original full pixel image.

Image Processing on the APR

We show how the APR reduces the memory and computational cost of downstream image-processing tasks (RC4). Once we have transformed the input image into an APR, the input image is no longer needed. All processing, storage, and visualization can be done directly on the APR.

Image-processing methods are always developed using a certain interpretation of images. Just like pixels, one can also interpret and use the APR in different ways depending on the processing task. These interpretations align with those commonly used in pixel-based processing. Figure 5A-D outlines the four main interpretations of the APR.

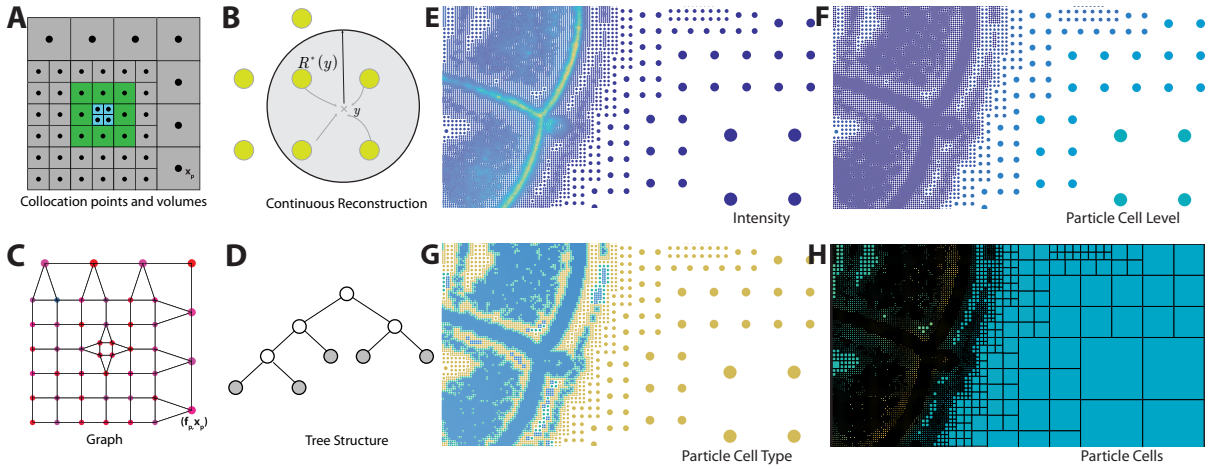

Figure 5: **Interpretations of the APR for image processing**. **A.** The APR can be interpreted as a spatial partition defined by the Particle Cells in $\mathcal{V}$, or by the set of particles $\mathcal{P}$ with positions x_p. This interpretation relates to the concept of super-pixels (*13*). **B.** The APR can be interpreted as a continuous function approximation where the intensity value can be reconstructed at each location y, also between particles and pixels, relating to smooth particle function approximations (*31*).**C.** The APR can be interpreted as a graph, where the particles are nodes and edges link neighboring particles (SuppMat 20). This relates the APR to graphical models often used on pixel images (*32*). **D.** The APR can be interpreted as a pruned binary tree (quad-tree in 2D, oct-tree in 3D) with links between parent and child Particle Cells. This relates the APR to wavelet decompositions (*33*), image pyramids (*26*), and tree-based methods (*34*). **E-H.** While particles store local fluorescence intensity, just like pixels (**E**), they also provide additional information that is not available on the pixels. This includes the Particle Cell level containing information about the local level of detail in the image (**F**), the Particle Cell type encoding the structure of the image (**G**), and the Particle Cells naturally decomposing the image domain in a content-adaptive way (**H**).

Performance metrics

The APR can accelerate existing algorithms in two ways: First, by decreasing the total processing time through reducing the number of operations that have to be executed. Second, by reducing the amount of memory required to run the algorithm. The relative importance of the two, and the degree of reduction, depends on the specific algorithm and its implementation. We use quantitative metrics to evaluate the improvements for different algorithms and input images.

The first evaluation metric relates to the computational performance of the algorithm. For a given algorithm and implementation, we define the speed-up (SU) as:

$$\text{SU} = \frac{\text{Processing time of the algorithm on pixels}}{\text{Processing time of the algorithm on APR}}. \tag{4}$$

It is insightful to relate the SU to the CR by SU = CR * (Pixel-Particle Speed Ratio) (PP), where PP = (Time to compute the operation on one pixel)/(Time to compute the operation on one particle). The value of PP depends on many factors, including memory access patterns, data structures, hardware, and the absolute size of the data in memory. Consequently, even for a given algorithm running on defined hardware, the PP is a function of the input image size N. Therefore, for tasks where PP$<$1, as in most low-level vision tasks, there is a minimum value of CR for which the algorithm is faster on the APR than on pixels.

The second evaluation metric relates to memory usage. We define the *Memory Reduction Ratio* (MRR) = (Memory used for pixel algorithm)/(Memory used for APR algorithm). Expressed using the CR, we define: MRR=CR*(Pixel-Particle Memory Cost) (MPP). Where MPP=(Memory required per pixel)/(Memory required per particle). For an algorithm run on a pixel image, the Memory Cost (MC) in Bytes usually scales linearly with the number of pixels N and algorithm variables, as MC=(Number of variables)*(Data type in Bytes)*N. The APR additionally requires storing particle locations and neighbor-access data structures. The memory cost of the APR MC = (Number of variables)*(Data type in Bytes)*N_p + (Cost of data

structure per particle)*N_p, where N_p is the number of particles, and the cost of the data structure per particle depends on N. We find an estimated average of 8 bits per particles overhead for the SA data structure SuppMat 17. Therefore, as the number of algorithm variables increases, the overhead of the APR is amortized so that the MPP approaches 1, and the MRR approaches the CR.

	Speed Up (SU)	Time APR (s)	Time Pixels (s)	PP	Memory Pixels (GB)	Memory APR (GB)	MRR
Linear Neighbor Iteration							
CR5	0.55 (0.02)	1.86 (0.07)	1.02 (0.0002)	0.097 (0.003)	3.072	0.599 (2.7)	5.12 (0.02)
CR20	1.9 (0.09)	0.54 (0.03)	1.02 (0.002)	0.096 (0.004)	3.072	0.181 (1.0)	16.9 (0.09)
CR100	7.1 (0.5)	0.14 (0.009)	1.02 (0.007)	0.076 (0.005)	3.072	0.053 (0.0005)	60.2 (2.3)
Exemplars Mean	4.06 (5.7)	0.86 (0.5)	1.83 (1.3)	0.094 (0.01)	5.61 (4.2)	0.278 (0.28)	37.5 (56)
Random Neighbor Access							
CR5	0.71 (0.03)	15.4 (0.2)	11.0 (0.4)	0.126 (0.005)	3.072	0.599 (2.7)	5.12 (0.02)
CR20	3.52 (0.3)	3.23 (0.05)	11.4 (0.8)	0.178 (0.01)	3.072	0.181 (1.0)	16.9 (0.09)
CR100	24.8 (0.8)	0.44 (0.01)	11.01 (0.3)	0.26 (0.007)	3.072	0.053 (0.0005)	60.2 (2.3)
Exemplars Mean	11.57 (23.6)	7.29 (10.1)	21.4 (16)	0.17 (0.05)	5.61 (4.2)	0.278 (0.28)	37.5 (56)
Image Filtering							
CR5	7.36 (1.2)	1.36 (0.009)	8.02 (0.2)	1.26 (0.2)	4.10	0.93 (0.003)	4.38 (0.04)
CR20	14.82 (3.7)	0.76 (0.01)	8.07 (0.3)	0.77 (0.2)	4.10	0.30 (0.002)	12.82 (0.9)
CR100	31.10 (13)	0.57 (0.003)	7.96 (0.3)	0.35 (0.15)	4.10	0.09 (0.0002)	36.85 (6.7)
Exemplars Mean	12.27 (3.0)	1.24 (0.93)	14.13 (9.8)	0.51 (0.33)	7.48 (5.5)	0.36 (0.28)	24.49 (19)
Image Segmentation							
CR5	5.10 (0.7)	1.87 (0.02)	8.86 (0.09)	0.86 (0.04)	$\approx$68.5*	12.57 (0.08)	5.51 (0.14)
CR20	18.30 (2.6)	0.48 (0.003)	8.83 (0.08)	0.95 (0.07)	$\approx$68.5*	3.75 (0.02)	18.18 (0.3)
CR100	85.3 (12)	0.10 (0.001)	8.78 (0.09)	0.97 (0.06)	$\approx$68.5*	0.80 (0.003)	84.09 (2.9)
Exemplars Mean	N/A	6.99 (5.9)	N/A	N/A	$\approx$385* (286)	13.54 (11.7)	39.72 (40)

Table 2: **Summary statistics of the image-processing benchmarks on synthetic and real-world images**. For the exemplars, we report the means (standard deviation in brackets) of the values over all exemplar images. For the synthetic fixed-CR datasets, the speed-up (SU), Pixel-Particle Speed Ratio (PP), and Memory Reduction Ratio (MRR) are averaged over image sizes from 200^3 to 1000^3; absolute timings and memory requirements are given for images of size 800^3. Graph-cut segmentation on pixels was not possible for 800^3 images as the memory requirement exceeded the 64 GB available on the benchmark machine. The corresponding entries in the table (marked with *) are extrapolations from benchmarks run on smaller images and the SU, PP, and pixel timing for the exemplars could not be determined in this case (*N/A*). See SuppMat 21 for a detailed descriptions of the benchmarks.

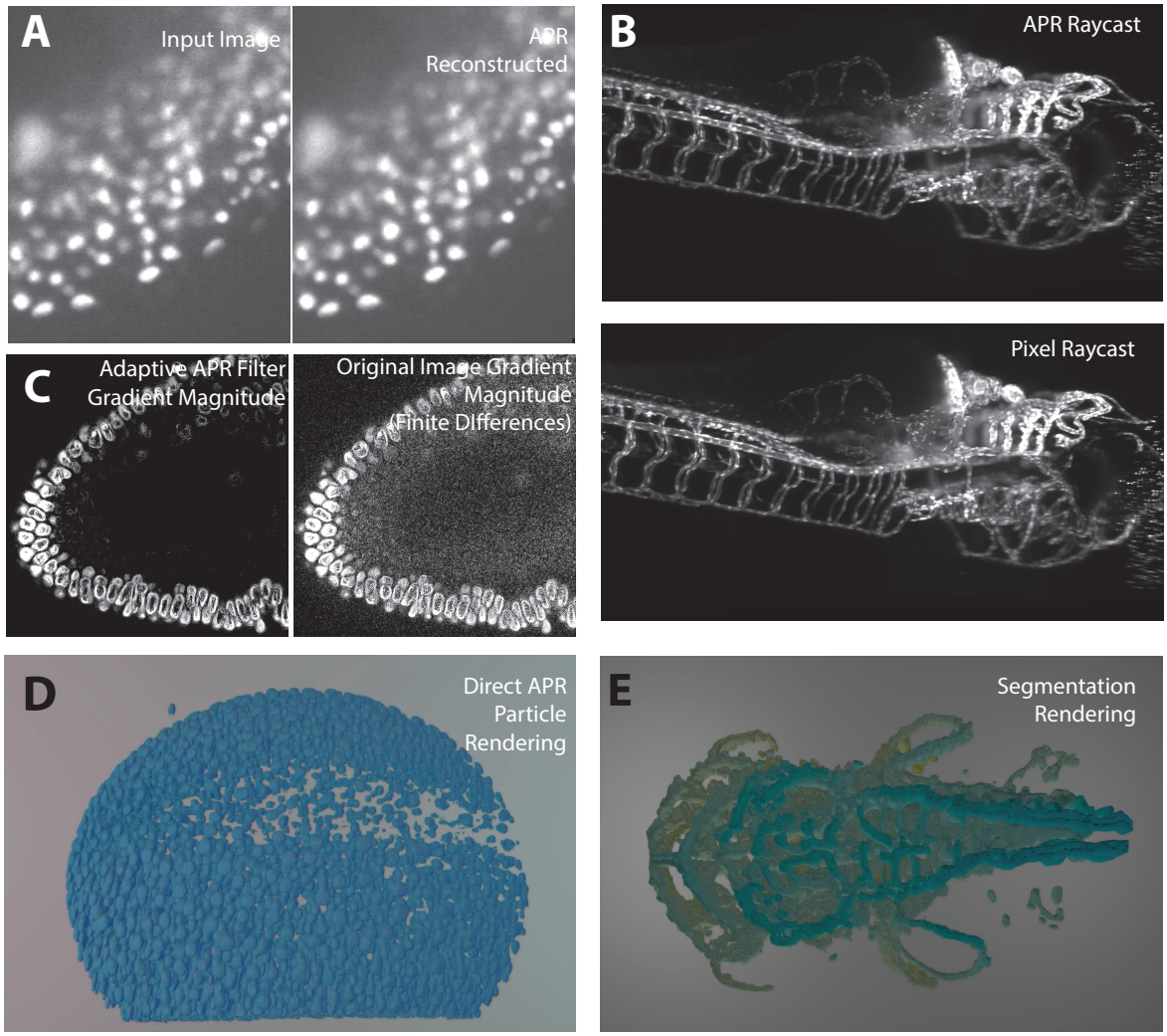

Figure 6: **Image processing using the APR**. **A.** Comparison of an example image (*left*, exemplar dataset 7) with its piecewise constant APR reconstruction (*right*), showing that they are visually indistinguishable. **B.** Comparison of the maximum-intensity projection of a direct 3D APR ray-cast (*top*) with the maximum projection of the pixels (*bottom*) for exemplar dataset 17 (full image see SFigure 40), showing that they are visually indistinguishable. **C.** Comparison of the intensity-gradient magnitude estimated using the Adaptive APR Filter (*left*, SuppMat 21.4) and central finite differences over the pixels (*right*) for exemplar dataset 6 (Tomancak Lab, MPI-CBG). The result computed on the APR has a higher signal-to-noise ratio because the filter adapts to image contents and does not amplify noise as finite differences do. **D.** Direct 3D particle rendering of Zebrafish nuclei (exemplar dataset 7) using a custom, *scenery*-based (*35*) renderer. Even without image segmentation, the nuclei are visible as dense clusters of particles. **E.** APR Volume rendering of a 3D image-segmentation result, colored by depth, computed using graph-cut segmentation directly on the APR, as described in SuppMat 21.3.3 (exemplar dataset 13, cf. **B**). Image segmentation can exploit the additional information provided by the APR to obtain higher-quality results at a lower computational cost. Segmenting this image on the APR took 5.5 seconds, and was not possible on the original pixel image using our benchmark machine. (**A,B,D,E** courtesy of Huisken Lab, MPI-CBG & Morgridge Institute for Research.)

Image Processing Performance Benchmarks

We analyze two low-level and one high-level image-processing task. These are neighbor access and filtering as low-level tasks, and image segmentation as high-level task. The low-level tasks represent a lower bound on the benefits of the APR due to their simple operations and access patterns, which are best suited for processing on pixels. The segmentation task in contrast provides a representative practical example of microscopy image analysis.

For these three benchmarks, we provide results for the computational and memory metrics for three fixed-CR datasets with input images from $N = 200^3$ up to $N = 1000^3$, and for all real-world exemplar datasets. The results of all benchmarks are summarized in Table 2. SuppMat 21 describes the benchmark protocols.

Neighbor Access For each pixel or particle, the task involves averaging the intensities of all face-connected neighbors (see SuppMat 21.1 for details). In the APR, neighbors are defined by the particle graph, as shown in Figure 5C and described in SuppMat 20. We benchmark two forms of neighbor access: *Linear iteration* loops over all neighbors in sequential order. *Random access* visits neighbors in random order, irrespective of how they are stored in memory.

For linear iteration, the APR shows low speed-ups. It is even slower than pixel operations for images with CR=5 and for four of the exemplar datasets (Table 2, group 1). This is because linear iteration is optimally suited to pixel images. However, the APR provides consistently higher speed-ups for random neighbor access, especially for high CRs. This is likely due to the smaller overall size of the APR improving cache efficiency.

The total memory cost of the APR reflects the CR of the dataset. This provides significant memory cost reductions across all benchmark datasets for both the linear and random neighbor access patterns.

Image filtering We consider the task of filtering the image with a Gaussian blur kernel (see SuppMat 21.2). We exploit the separability of the kernel and perform three consecutive filtering steps using 1D filters in each direction. On the APR, this requires locally evaluating the function reconstruction. For simplicity, we use the piecewise constant reconstruction method (see SuppMat 10). The benchmark results are shown in Table 2, group 3. Directly filtering the APR consistently outperforms the pixel-based pipeline, both in terms of memory cost and execution time.

In SuppMat 21.2.2 we analyze the results in detail and find that the APR is most appropriate if the filtering result looks similar to the original image, such that the same set of content-adapted particles is also suitable to represent the filtered image. SFigure 35 illustrates this, showing how for a small blur the APR filter has higher PSNR than the pixel filter. For larger blurs this is reversed, because the specific APR adapted to the input image is no longer suitable to represent the filtered image. Care must be taken when designing algorithms, as not all approaches are equally suited to the APR.

Image segmentation We perform binary image segmentation using graph cuts (see SuppMat 21.3). We use the method and implementation of Ref. (*32*) to compute the optimal foreground/background segmentation for both APR and pixel images. When computing the cut energies, we directly exploit the additional information provided by the particle cell level, type, and local min-max range. To allow direct comparison with the pixel-based segmentation, we interpolate all energies calculated on the APR to pixels and then determine the cuts over the pixel image using the same energies. For both APR and pixel images, a face-connected neighborhood graph is used. Given the energy calculations are identical, we benchmark the execution time and memory cost of the graph-cut solver. The results are shown in Table 2, group 4. For the APR we find speed-ups directly reflecting the CR. Due to high memory requirements of

the graph-cut solver, pixel images can only be segmented for sizes $N \leq 550^3$ on our benchmark machine with 64 GB RAM. Using the APR, also larger images can be segmented without problems, illustrating the benefits of the reduced memory cost of the APR.

We validate the APR segmentations, in SuppMat 21.3.2 by comparing both the APR and pixel-based segmentations to ground truth using the Dice coefficient (*36*). SFigure 36 provides an illustrative comparison. Across datasets, we find that the Dice coefficients are not statistically significantly different (p-value: 0.92, Welch's t-test).

We provide a representative example in SVideo 2 and show a 3D rendering of a segmentation in Figure 6E.

Novel Algorithms

The APR provides additional information about the image that is not contained in pixel representations. This information can be exploited in image-processing algorithms, as illustrated in the segmentation example above. In addition, it can also be used to design entirely novel, APR-specific algorithms, as demonstrated in the following example.

Adaptive APR filter We define a discrete filter over neighboring particles in the APR particle graph. Since the distance between neighboring particles varies across the image depending on image content, this amounts to spatially adaptive filtering with the filter size automatically adjusting to the content of the image. On the APR, this only requires linear neighbor iteration. In contrast, an adaptive pixel implementation would be significantly more involved. SuppMat 21.4 describes the adaptive APR filter in detail. SFigure 37 shows synthetic results for an adaptive blurring filter, and SFigure 38 for a filter that adaptively estimates the intensity gradient magnitude. In both examples, the adaptive APR-filtered results have significantly higher PSNR than results from corresponding non-adaptive pixel filters. This is significant, as shown in Figure 6C

and SFigure 39 for a partial and full slice from an exemplar image. Across exemplars, the adaptive APR filter shows superior robustness to noise. The computational and memory costs are identical to those of the linear neighbor iteration benchmark above.

Visualization

Images represented using the APR can directly be visualized without going back to pixels. The APR image can be visualized using both traditional and novel visualization methods. We provide examples of visualization methods and refer to SuppMat 21.5 for details.

Visualization by slice First, visualization can be done using a slice-by-slice function reconstruction, never having to reconstruct the entire image. Figure 6A and SVideo 1 show examples of the APR reconstruction in comparison with the pixel image. The piecewise constant reconstruction used here is computationally efficient and works well for near-isotropic images. However, as shown in SFigure 24, piecewise constant reconstructions may show blocking artifacts in low-content areas of the image, which is avoided when using higher-order smooth function reconstructions at a higher computational cost.

Visualization by ray-casting A second method, allowing for direct 3D visualization of an APR, is ray-casting as described in SuppMat 21.5.2. Figure 6B (full image in SFigure 40) and SVideo 3 show a perspective maximum-intensity projection in comparison with the same ray-cast of the original pixel image. The resulting visualizations are largely indistinguishable SFigure 41 shows a contrast-adjusted version to highlight the differences. APR ray-casting only requires storing and computing on the APR, therefore reducing memory and computational costs proportionally to the CR of the image. Thus, APR ray-casting is useful for visualizing large images, such as those exemplars over 2 GB, which cannot be rendered at full resolution by state-of-the-art pixel-based software (*37*).

Visualization by particle rendering Lastly, we can directly visualize the particles of the APR as glyphs (see SuppMat 21.5.3). This can be done both in 2D (Figure 3) and in 3D (Figure 5). Figure 6D and SVideo 4&5 show examples of particle renderings in 3D using open-source rendering toolkit *scenery* (*35*). These direct visualization techniques natively allow visualizations that decouple the observing of the structure of the image content from the information displayed using coloring and size of particles.

Image Processing Summary

Across all benchmarks and exemplar datasets other than the worst-case example of linear neighbor access, processing directly on the APR resulted in smaller execution times and memory costs. In most cases, the reductions are directly proportional to the computational ratio (CR), hence fulfilling RC4. Moreover, in the examples of visualization and segmentation, the memory cost reduction of the APR enabled processing of data sets that would not otherwise have been possible on our benchmark machine. The APR has a range of interpretations that align with those of pixel images, allowing direct application of established image-analysis frameworks to the APR.

In addition, we highlight that the APR may simplify processing tasks by providing additional information about the structure of the image through the Particle Cell level and type. This structural information can be leveraged in existing algorithms, as shown for segmentation, or it can be used to design novel algorithms, such as the adaptive APR filter and APR ray-casting visualization.

Discussion and Conclusion

We have introduced a novel content-adaptive image representation for fluorescence microscopy, the Adaptive Particle Representation (APR). The APR is inspired by how the human visual

system effectively avoids the data and processing bottlenecks that plague modern fluorescence microscopy, particularly for 3D imaging. The APR combines aspects of previous adaptive-resolution methods, including wavelets, super-pixels, and equidistribution principles in a way that fulfills all representation criteria set out in the introduction. The APR is computationally efficient, suited for real-time applications at acquisition speed, and easy to implement.

We presented the ideas and concepts of the APR in 1D for ease of illustration, with all naturally extending to higher dimensions. The APR resamples an image by adapting a set of Particle Cells $\mathcal{V}$ and a set of particles $\mathcal{P}$ to the content of an image, taking into account the Local Intensity Scale σ similar to gain control in the human visual system. The main theoretical and algorithmic contribution that made this possible with a computational cost that scales linearly with image contents is the Pulling Scheme. The Pulling Scheme guarantees sub-optimal image representations within user-specified relative intensity deviations.

We verified accuracy and performance of the APR using synthetic benchmark images. The analysis showed that all theoretical results hold in practice, and that the number of particles used by the APR scales with image content while maintaining image quality (RC1). Further, we showed that although image noise places a limit on representation accuracy, there exists an optimal range for the relative error threshold E. In this range, the reconstruction error for noisy images is always well within the imaging noise level (RC1). Moreover, we found that the number of particles is independent of the original image size, with computational and memory costs of the APR proportional to the information content of the image (RC2). We showed how pixel images can rapidly be transformed to the APR, and efficiently stored both in memory and in files (RC3). We have demonstrated that the APR benefits both in terms of execution time and memory requirements can be leveraged for a range of image-processing tasks without ever returning to a pixel image (RC4). Finally, we showed how the adaptive sampling and structure of the APR inspires the development of novel, content-adaptive image-processing algorithms.

Taken together, the APR meets all four Representation Criteria (RC) set out in the introduction. We believe that the gains of the APR will in many cases be sufficient to alleviate the current processing bottlenecks. In particular, image-processing pipelines using the APR would be well suited for high-throughput experiments and real-time processing, e.g., in smart microscopes (*9, 38*). However, the APR is sub-optimal with respect to the number of particles used. This sub-optimality results from the conservative limiting assumptions required to derive the efficient Pulling Scheme. It is easily seen by the fact that the APR particle properties could be represented by a Haar wavelet transform (*33*) with non-zero coefficients whose number is either equal to, or less than, the number of particles in the APR while allowing exact reconstruction of the APR particle properties.

The use of adaptive representations of images (*39–41*) and its motivation by the human visual system (*13,42*) are not new. The APR shares several principles and ideas with established adaptive representations. The Resolution Function $R(y)$ of the APR, e.g., is related to the oracle adaptive regression method (*43*) and the derivation and form of the Resolution Bound are related to ideas originally introduced in equidistribution methods for splines (*18, 44, 45*), which also inspired the work here (*20*). The Reconstruction Condition for a constant Local Intensity Scale relates to infinity norm adaptation (*46*) for wavelet thresholding in adaptive surface representations. Further, the use of a powers-of-two decomposition of the domain is central to many adaptive-resolution methods (*26, 33, 34, 47*) and its use here was particularly inspired by Ref. (*48*). Despite its similarity to existing methods, the APR uniquely fulfills all representation criteria and extends many of the previous concepts. Core novelties include the spatially varying Local Intensity Scale, the fact that the APR works with a wider class of reconstruction methods, provides theoretical bounds on arbitrary derivatives of the represented function, and enables integration of additional spatial constraints with no changes to the Particle Cell formulation or the Pulling Scheme algorithm.

Outlook

The APR has the potential to completely replace pixel-based image-processing pipelines for the next generation of fluorescence microscopes. We envision that the APR is immediately formed, possibly after image enhancement (*49*), on the acquisition computer or even on the camera itself. Following this, all data transfer, storage, visualization, and processing can be done using the APR, providing memory and computational gains across all tasks. Although, there are also contexts when the exact pixel noise distribution of the original image conveys information, and therefore the APR is not appropriate. In addition, the realization of such pipelines requires further algorithm and software development including integration with current microscope systems, image databases (*50*), and image-processing tools (*51*).

Here, we presented a particular realization of an APR pipeline. We foresee alternative pipelines, e.g., using deep learning approaches (*52*) to provide improved estimation of the Local Intensity Scale, the image intensity gradient, and the smooth image reconstruction. Just as in space, the APR can also be used to adaptively sample time. Such temporal adaptation can lead to a multiplicative reduction in memory and computational costs compared to those presented here, allowing even faster APR computations. Further, the APR can be extended to allow for anisotropic adaptation using rectangular particle cells and anisotropic particle distributions within each cell.

Given the wide success of adaptive representations in scientific computing, the unique features of the APR could be useful also in non-imaging applications. This includes applications to time-series data, where the APR could provide an adaptive regression method (*43*), and to surface representation in computer graphics (*46*). Further, the APR could be used in numerical simulations for efficient mesh generation or as an adaptive mesh-free collocation method for numerically solving partial differential equations (*20, 53–55*).

References and Notes

1. J. Huisken, J. Swoger, F. Del Bene, J. Wittbrodt, E. H. Stelzer, *Science* **305**, 1007 (2004).

2. P. J. Keller, A. D. Schmidt, J. Wittbrodt, E. H. Stelzer, *Science* **322**, 1065 (2008).

3. B.-C. Chen, *et al.*, *Science* **346**, 1257998 (2014).

4. D. C. Prasher, V. K. Eckenrode, W. W. Ward, F. G. Prendergast, M. J. Cormier, *Gene* **111**, 229 (1992).

5. M. Jinek, *et al.*, *Science* **337**, 816 (2012).

6. A. C. Oates, N. Gorfinkiel, M. Gonzalez-Gaitan, C.-P. Heisenberg, *Nature Reviews Genetics* **10**, 517 (2009).

7. E. G. Reynaud, J. Peychl, J. Huisken, P. Tomancak, *Nature methods* **12**, 30 (2015).

8. M. Weber, J. Huisken, *Current opinion in genetics & development* **21**, 566 (2011).

9. N. Scherf, J. Huisken, *Nature biotechnology* **33**, 815 (2015).

10. P. Reinagel, A. M. Zador, *Network: Computation in Neural Systems* **10**, 341 (1999).

11. S. M. Smirnakis, M. J. Berry, D. K. Warland, W. Bialek, M. Meister, *Nature* **386**, 69 (1997).

12. K. Koch, *et al.*, *Current Biology* **16**, 1428 (2006).

13. R. Achanta, *et al.*, *IEEE transactions on pattern analysis and machine intelligence* **34**, 2274 (2012).

14. F. Amat, E. W. Myers, P. J. Keller, *Bioinformatics* **29**, 373 (2012).

15. S. G. Mallat, *IEEE transactions on pattern analysis and machine intelligence* **11**, 674 (1989).

16. I. Daubechies, *Communications on pure and applied mathematics* **41**, 909 (1988).

17. A. Harten, *Journal of Computational Physics* **115**, 319 (1994).

18. C. de Boor, *Spline functions and approximation theory* (Springer, 1973), pp. 57–72.

19. V. Pereyra, E. Sewell, *Numerische Mathematik* **23**, 261 (1974).

20. S. Reboux, B. Schrader, I. F. Sbalzarini, *Journal of Computational Physics* **231**, 3623 (2012).

21. B. Schmid, *et al.*, *Nature communications* **4** (2013).

22. I. Heemskerk, S. J. Streichan, *Nature methods* **12**, 1139 (2015).

23. The HDF Group, Hierarchical Data Format, version 5 (1997-2017). Http://www.hdfgroup.org/HDF5/.

24. F. Alted, Blosc, an extremely fast, multi-threaded, meta-compressor library (2017).

25. M. Unser, A. Aldroubi, M. Eden, *IEEE transactions on signal processing* **41**, 834 (1993).

26. E. H. Adelson, C. H. Anderson, J. R. Bergen, P. J. Burt, J. M. Ogden, *RCA engineer* **29**, 33 (1984).

27. B. Balazs, J. Deschamps, M. Albert, J. Ries, L. Hufnagel, *bioRxiv* p. 164624 (2017).

28. B. Schmid, J. Huisken, *Bioinformatics* **31**, 3398 (2015).

29. Y. Afshar, I. F. Sbalzarini, *PloS one* **11**, e0152528 (2016).

30. F. Amat, *et al.*, *Nature protocols* **10**, 1679 (2015).

31. J. J. Monaghan, *SIAM Journal on Scientific and Statistical Computing* **3**, 422 (1982).

32. Y. Boykov, V. Kolmogorov, *IEEE transactions on pattern analysis and machine intelligence* **26**, 1124 (2004).

33. A. Haar, *Mathematische Annalen* **69**, 331 (1910).

34. D. Meagher, *Computer graphics and image processing* **19**, 129 (1982).

35. scenerygraphics/scenery: scenery 0.2.3-1, https://doi.org/10.5281/zenodo.1111824 (2017).

36. L. R. Dice, *Ecology* **26**, 297 (1945).

37. L. A. Royer, *et al.*, *Nature methods* **12**, 480 (2015).

38. L. A. Royer, *et al.*, *Nature biotechnology* **34**, 1267 (2016).

39. L. Demaret, A. Iske, *Curve and Surface Fitting: Saint-Malo* **2003**, 107 (2002).

40. Y. Wang, O. Lee, A. Vetro, *IEEE Transactions on circuits and systems for video technology* **6**, 647 (1996).

41. Y. Yang, M. N. Wernick, J. G. Brankov, *IEEE transactions on image processing* **12**, 866 (2003).

42. A. Witkin, *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'84*. (IEEE, 1984), vol. 9, pp. 150–153.

43. D. L. Donoho, I. M. Johnstone, *biometrika* pp. 425–455 (1994).

44. C. De Boor, *Conference on the numerical solution of differential equations* (Springer, 1974), pp. 12–20.

45. H. G. Burchard, *Applicable Analysis* **3**, 309 (1974).

46. R. A. DeVore, B. Jawerth, B. J. Lucier, *Computer Aided Geometric Design* **9**, 219 (1992).

47. R. Zhao, T. Tao, M. Gabriel, G. G. Belford, *Proc. SPIE* (2002), vol. 4925, p. 180.

48. O. Awile, F. Büyükkeçeci, S. Reboux, I. F. Sbalzarini, *Computer Physics Communications* **183**, 1073 (2012).

49. M. Weigert, *et al.*, *bioRxiv* (2017).

50. I. G. Goldberg, *et al.*, *Genome biology* **6**, R47 (2005).

51. J. Schindelin, *et al.*, *Nature methods* **9**, 676 (2012).

52. I. Goodfellow, Y. Bengio, A. Courville, *Deep learning* (MIT press, 2016).

53. O. V. Vasilyev, C. Bowman, *Journal of Computational Physics* **165**, 660 (2000).

54. B. Schrader, S. Reboux, I. F. Sbalzarini, *Journal of Computational Physics* **229**, 4159 (2010).

55. D. Rossinelli, *et al.*, *Journal of Computational Physics* **288**, 1 (2015).