# FlashDeconv: Fast Linear Algebra for Scalable Hybrid Deconvolution

## Introduction

This is a fully evolved methodological framework. Based on reflections on Cell2location (accuracy), RCTD (platform effects), and CARD (spatial correlation), this new version addresses the shortcomings of **non-Gaussian distribution handling**, **platform effect correction**, and **rare cell type preservation**, while maintaining the extreme speed brought by **randomized sketching**.

## 1 Methods: FlashDeconv

### 1.1 Overview and Statistical Framework

The fundamental challenge in spatial transcriptomics (ST) deconvolution is solving the cell type abundance matrix $\beta \in \mathbb{R}^{N \times K}$ given the spatial gene expression matrix $Y \in \mathbb{R}^{N \times G}$ and the single-cell reference signature matrix $X \in \mathbb{R}^{K \times G}$, such that $Y \approx \beta X$. Current Bayesian methods (e.g., Cell2location) model the raw counts using Negative Binomial distributions, which provides high accuracy but incurs prohibitive computational costs for million-scale datasets. Conversely, fast linear regression methods (e.g., NNLS) often ignore the heteroscedastic noise of count data and platform-specific effects.

**FlashDeconv** introduces a four-stage framework that combines:

1. **Gene Selection** to identify informative genes (highly variable genes + cell-type markers).
2. **Data Preprocessing** with flexible normalization options (log-CPM, Pearson residuals, or raw).
3. **Structure-Preserving Matrix Sketching** to compress genomic dimensions ($G \approx 20,000$) into a biologically informative low-dimensional subspace ($d \approx 500$), reducing memory usage by orders of magnitude.
4. **Spatial Graph Regularized Optimization** to explicitly model spatial autocorrelation among neighboring spots, solved via a high-performance Coordinate Descent algorithm.

### 1.2 Data Preprocessing

Raw ST data follows a Poisson-Gamma mixture distribution where variance depends on the mean. Direct application of Euclidean-based sketching (Johnson-Lindenstrauss lemma) on raw counts is statistically invalid. FlashDeconv provides three preprocessing strategies, selectable via the `preprocess` parameter:

#### 1.2.1 Method 1: Log-CPM Normalization (Default)

The default and recommended preprocessing (`preprocess="log_cpm"`) normalizes counts to Counts Per Million (CPM) and applies log1p transformation:

$$\tilde{Y}_{ig} = \log(1 + 10^4 \cdot \frac{Y_{ig}}{N_i}), \quad \tilde{X}_{kg} = \log(1 + 10^4 \cdot \frac{X_{kg}}{N_k^{ref}}) \tag{1}$$

where $N_i = \sum_g Y_{ig}$ is the total count for spot $i$ and $N_k^{ref} = \sum_g X_{kg}$ is the total count for cell type $k$. This normalization:

- Removes sequencing depth effects
- Stabilizes variance via log transformation
- Preserves non-negativity for downstream NNLS

### 1.2.2 Method 2: Uncentered Pearson Residuals

For variance-stabilizing transformation (`preprocess="pearson"`), we use **uncentered** Pearson residuals that divide by the expected standard deviation:

$$\tilde{Y}_{ig} = \frac{Y_{ig}}{\sigma_g^Y}, \quad \tilde{X}_{kg} = \frac{X_{kg}}{\sigma_g^X} \tag{2}$$

where the gene-wise standard deviations are computed using a Negative Binomial variance model:

$$\sigma_g^Y = \sqrt{\bar{\mu}_g^Y + (\bar{\mu}_g^Y)^2/\theta}, \quad \sigma_g^X = \sqrt{\bar{\mu}_g^X + (\bar{\mu}_g^X)^2/\theta} \tag{3}$$

Here $\bar{\mu}_g^Y = \text{mean}_i(Y_{ig})$ and $\bar{\mu}_g^X = \text{mean}_k(X_{kg})$ are the gene-wise means, and $\theta = 100$ is a fixed overdispersion parameter.

**Key Design Choice:** Unlike standard Pearson residuals $(Y - \mu)/\sigma$, we use uncentered residuals $Y/\sigma$ to preserve non-negativity. This is essential because downstream solving uses non-negative least squares (NNLS), which requires non-negative inputs.

### 1.2.3 Method 3: Raw Counts

For pre-normalized data or synthetic experiments (`preprocess="raw"`), the raw counts can be used directly:

$$\tilde{Y} = Y, \quad \tilde{X} = X \tag{4}$$

**Recommendation:** Use `log_cpm` (default) for most real datasets. Use `pearson` for data with high technical noise. Use `raw` only for pre-normalized or synthetic data.

## 1.3 Structure-Preserving Randomized Sketching

Solving the regression problem in the original $G$-dimensional space ($G \approx 20,000$) is computationally wasteful due to the high collinearity of gene expression. We propose **Structure-Preserving Sketching**, an improvement over standard random projections.

We define a sketching matrix $\Omega \in \mathbb{R}^{G \times d}$, where $d \ll G$ (default $d = 512$). Instead of a dense Gaussian matrix (which is slow to generate and multiply), we use a **Sparse CountSketch Matrix** with an **Importance Sampling** twist:

1. **Feature Selection:** We select a subset of Informative Genes $\mathcal{G}_{info}$ (Union of Highly Variable Genes and Cell-type Specific Markers) to prevent noise accumulation.
2. **Weighted Sketching:** For each column $j \in \{1 \dots d\}$ of $\Omega$, we assign exactly one non-zero value $\pm 1$ for each gene $g$, but the probability of assignment is weighted by the gene's **Leverage Score** (biological importance). This ensures that marker genes for rare cell types are preserved with high probability.

### 1.3.1 CountSketch Matrix Construction (Detailed Algorithm)

The CountSketch matrix $\Omega \in \mathbb{R}^{G \times d}$ is constructed as follows:

**Input:** Number of genes $G$, sketch dimension $d$, leverage scores $\ell \in \mathbb{R}^G$

**Algorithm:**

1. Normalize leverage scores: $p_g = \ell_g / \sum_{g'} \ell_{g'}$ (probability distribution)
2. For each gene $g \in \{1, \dots, G\}$:

   (a) Hash assignment: $h(g) \sim \text{Uniform}(\{1, \ldots, d\})$
   (b) Sign assignment: $s(g) \sim \text{Uniform}(\{-1, +1\})$
   (c) Importance weight: $w_g = \sqrt{p_g \cdot G}$ (scaled by leverage)
   (d) Clip weight: $w_g = \text{clip}(w_g, 0.1, 10.0)$ (numerical stability)

3. Construct sparse matrix: $\Omega[g, h(g)] = s(g) \cdot w_g$

4. Column normalization: For each column $j$:

$$\Omega[:, j] \leftarrow \Omega[:, j] \cdot \frac{\sqrt{G/d}}{\|\Omega[:, j]\|_2} \tag{5}$$

**Output:** Sparse matrix $\Omega$ of shape $(G, d)$ with exactly $G$ non-zero entries

The key innovation is the **importance weighting** $w_g = \sqrt{p_g \cdot G}$: genes with higher leverage scores contribute more strongly to the sketch, preserving discriminative information for rare cell types.

### 1.3.2 Leverage Score Computation

Leverage scores quantify the importance of each gene for distinguishing cell types. We compute them via SVD of the reference matrix:

$$X^T = U \Sigma V^T \tag{6}$$

where $X \in \mathbb{R}^{K \times G}$ is the (centered) reference signature matrix.

The leverage score for gene $g$ is:

$$\ell_g = \sum_{j=1}^{r} \frac{\sigma_j^2}{\sigma_j^2 + \epsilon} \cdot U_{gj}^2 \tag{7}$$

where $r = \min(K, G)$ is the rank, $\sigma_j$ are singular values, and $\epsilon = 10^{-6}$ prevents division by zero. This weighted sum emphasizes genes that lie in the principal subspace of the cell type signatures.

**Intuition:** Genes with high leverage scores are those that:

- Have high variance across cell types (captured by large $\sigma_j$)
- Are not redundant with other genes (unique directions in $U$)
- Are essential for distinguishing rare cell types (marker genes)

### 1.3.3 Integration: CountSketch + Importance Sampling

The combination works as follows:

| **Component** | **Standard CountSketch** | **Our Method** |
|---|---|---|
| Hash function | Uniform random | Uniform random |
| Sign function | $\pm 1$ uniform | $\pm 1$ uniform |
| Entry value | $\pm 1$ | $\pm w_g$ (leverage-weighted) |
| Expectation | $\mathbb{E}[\Omega^T \Omega] = I$ | $\mathbb{E}[\Omega^T \Omega] \approx I$ (weighted) |

**Theoretical Guarantee (Leverage Score Sampling Framework):**

Following the leverage score sampling framework (Drineas et al., 2006; Mahoney, 2011), the weighted sketch preserves the column space of $X$ with high probability. Specifically, for the regression problem $\min_\beta \|Y - \beta X\|_F^2$, solving in the sketched space yields a $(1+\epsilon)$-approximation:

$$\|\tilde{Y} - \hat{\beta}\tilde{X}\|_F^2 \leq (1+\epsilon) \min_\beta \|\tilde{Y} - \beta\tilde{X}\|_F^2 \tag{8}$$

with probability at least $1 - \delta$, where $d = O(K \log(K)/\epsilon^2)$ suffices.

The importance weighting ensures that the effective approximation error is **lower for high-leverage genes** (cell type markers), preserving rare cell type signals even after dimension reduction. Unlike standard Johnson-Lindenstrauss projections, this leverage-based approach provides **non-uniform preservation** tailored to the regression structure.

We then project the data into the low-dimensional "sketch space":

$$Y_{sketch} = \tilde{Y}\Omega \quad (N \times d) \tag{9}$$

$$X_{sketch} = \tilde{X}\Omega \quad (K \times d) \tag{10}$$

**Theoretical Guarantee:** According to the Johnson-Lindenstrauss lemma and recent extensions for oblivious subspace embeddings, solving the regression in this $d$-dimensional sketch space yields a solution $\hat{\beta}$ that is an $(1+\epsilon)$-approximation of the optimal solution in the original space, with high probability.

## 1.4 Spatial Graph Laplacian Regularization

To incorporate spatial information (inspired by CARD) without the computational burden of inverting dense covariance matrices, we introduce a **Graph Laplacian Regularizer**.

We construct a spatial neighbor graph adjacency matrix $A$, where $A_{ij} = 1$ if spot $i$ and $j$ are physical neighbors, else 0. The Laplacian matrix is $L = D - A$, where $D$ is the degree matrix.

Our final objective function in the sketch space is:

$$\mathcal{L}(\beta) = \underbrace{\frac{1}{2}\|Y_{sketch} - \beta X_{sketch}\|_F^2}_{\text{Fidelity in Sketch Domain}} + \underbrace{\lambda \cdot \mathrm{Tr}(\beta^T L \beta)}_{\text{Spatial Smoothing}} + \underbrace{\rho\|\beta\|_1}_{\text{Sparsity}} \tag{11}$$

Subject to: $\beta \geq 0$.

The term $\mathrm{Tr}(\beta^T L\beta) = \sum_{(i,j)\in E} \|\beta_i - \beta_j\|^2$ forces neighboring spots to have similar cell type compositions, effectively denoising "drop-out" events in low-coverage spots.

## 1.5 Fast Optimization: Block Coordinate Descent

While the objective function is convex, standard gradient descent is slow for large $N$. We derive a **Block Coordinate Descent (BCD)** algorithm that solves row-by-row (spot-by-spot) but leverages shared pre-computed matrices.

For each spot $i$, the update rule for cell type $k$ ($\beta_{ik}$) has a **closed-form solution** combining soft-thresholding (for L1) and projection to non-negative (for constraints).

Let $G = X_{sketch}X_{sketch}^T$ (the Gram matrix, precomputed once). The update is:

$$r_{ik} = (Y_{sketch})_i (X_{sketch})_k^T - \sum_{j\neq k} \beta_{ij} G_{jk} + \lambda \sum_{n\in\mathcal{N}(i)} \beta_{nk} \tag{12}$$

$$\beta_{ik} \leftarrow \left[\frac{S_\rho(r_{ik})}{G_{kk} + \lambda|\mathcal{N}(i)|}\right]_+ \tag{13}$$

where $S_\rho(x) = \mathrm{sign}(x) \cdot \max(0, |x| - \rho)$ is the **soft-thresholding operator** for L1 regularization, and $[\cdot]_+ = \max(0, \cdot)$ enforces non-negativity.

**Convergence Criteria:**

- **Maximum iterations**: 100 (default)
- **Tolerance**: rel_change $= \frac{\max|\beta^{(t)} - \beta^{(t-1)}|}{\max|\beta^{(t-1)}| + 10^{-10}} < 10^{-4}$
- Empirically, convergence is achieved in **10-30 iterations** for most datasets.

**Computational Edge:**

- The term $(X_{sketch})^T X_{sketch}$ is a tiny $K \times K$ matrix computed only once.
- The neighbor sum $\sum_{n\in\mathcal{N}(i)} \beta_{nk}$ exploits the sparsity of the spatial graph.
- The algorithm converges in very few iterations and is fully parallelizable across spots.

## 1.6 Complexity Analysis

| **Component** | **Cell2location (VI)** | **CARD (CAR)** | **FlashDeconv (Ours)** |
|---|---|---|---|
| Data Representation | Dense $N \times G$ | Dense $N \times G$ | Sketch $N \times d$ |
| Spatial Model | N/A (or slow MRF) | Dense Matrix Inversion | Sparse Laplacian |
| Time Complexity | $O(Iter \cdot N \cdot G \cdot K)$ | $O(N^3)$ (approx) | $O(Iter \cdot N \cdot d \cdot K)$ |
| Memory Usage | High (GBs to TBs) | High ($N \times N$ matrix) | **Ultra-Low** ($O(Nd)$) |
| 1M Spots Time | $>$ 24 Hours (GPU) | Out of Memory | $\sim$ **10 Minutes (CPU)** |

Table 1: Computational complexity comparison of different deconvolution methods.

## 1.7 Implementation Details for Reproducibility

- **Software:** Implemented in Python using `Numba` for JIT compilation of the BCD solver.
- **Hardware Independence:** Unlike Tangram or Cell2location, FlashDeconv requires **no GPU** and runs efficiently on standard laptops (e.g., MacBook Air with 8GB RAM).

**Default Hyperparameters:**

| **Parameter** | **Default** | **Valid Range** | **Description** |
|---|---|---|---|
| $d$ (sketch_dim) | 512 | [128, 2048] | Sketch dimension |
| $\lambda$ (lambda_spatial) | 5000 or "auto" | $[0, \infty)$ | Spatial smoothing; 0 disables |
| $\rho$ (rho_sparsity) | 0.01 | [0, 0.1] | L1 regularization for sparsity |
| n_hvg | 2000 | [500, 5000] | Number of highly variable genes |
| n_markers_per_type | 50 | [20, 200] | Marker genes per cell type |
| k_neighbors | 6 | [4, 20] | Neighbors for spatial graph |
| preprocess | "log_cpm" | {log_cpm, pearson, raw} | Preprocessing method |
| max_iter | 100 | – | Maximum BCD iterations |
| tol | $10^{-4}$ | – | Convergence tolerance |

**Automatic $\lambda$ Tuning:**
When `lambda_spatial="auto"`, we estimate $\lambda$ by balancing the data fidelity term and the spatial regularization term. The key insight is that $\lambda$ must be scaled relative to the Gram matrix $X_{sketch}^T X_{sketch}$ to have meaningful effect on the optimization.

In the BCD update, the denominator is:

$$\text{denom} = (X_{sketch}^T X_{sketch})_{kk} + \lambda \cdot n_{neighbors} \tag{14}$$

For spatial regularization to contribute meaningfully, we set:

$$\lambda = \alpha \cdot \frac{\text{mean}(\text{diag}(X_{sketch}^T X_{sketch}))}{\bar{d}} \tag{15}$$

where:

- $\alpha = 0.005$ (default) controls the relative strength of spatial regularization
- $\bar{d}$ is the average number of neighbors per spot

This ensures that the spatial term contributes approximately $\alpha/(1+\alpha) \approx 0.5\%$ to the Hessian diagonal. Users can adjust the effective regularization strength by setting a fixed `lambda_spatial` value (recommended range: 1000-10000 for Visium data).

**Numerical Stability Clipping:**

| Quantity | Clip Range | Rationale |
|---|---|---|
| Sketch weights | [0.1, 10] | Balanced importance sampling |
| Leverage scores | Normalized to sum=1 | Proper probability distribution |

## Key Design Decisions

1. **Flexible Preprocessing:** Unlike Cell2location/RCTD which require specific data formats, FlashDeconv supports multiple preprocessing modes (`log_cpm`, `pearson`, `raw`) to accommodate different data types and user preferences.

2. **Spatial Regularization via Graph Laplacian:** Inspired by CARD's spatial modeling, but using sparse Graph Laplacian instead of dense CAR matrix inversion. This provides $O(N \cdot K)$ complexity instead of $O(N^3)$.

3. **Structure-Preserving Sketching:** Leverage-weighted CountSketch ensures rare cell type markers are preserved during dimension reduction, addressing the "random projection loses rare cells" concern.

4. **CPU-only, Memory-efficient:** No GPU required. The algorithm runs efficiently on standard laptops (8GB RAM), making it accessible for Stereo-seq/Xenium users with million-scale datasets.

5. **Parameter Sensitivity:** The `lambda_spatial` parameter may require tuning for different data types:

   - For sequencing-based platforms (10x Visium): default 5000 works well
   - For imaging-based platforms (seqFISH+, MERFISH): consider `lambda_spatial=0` to disable spatial regularization when sample sizes are small