The Background:
LSST Deblender

NMF Model

\[ \text{Image} = \sum_{k} \text{SED}_k \times \text{Morphology}_k + \text{noise} \]

\[ \text{model} = S\Gamma M \]

- $\Gamma$ describes PSF convolution and translation term
- $k$ is the total number of components
- For example:

\[ \Gamma = T_Y P T_X \]

- The solution is to minimize

\[ \| D - S\Gamma M \|^2_2 + g(S, M) \]

See Moolekamp and Melchior 2017 (in Press) and Melchior et al. 2017 (in Prep) for details about the algorithms
Basic Algorithm

- Initialize the model with the SED of a single pixel morphology for each object (the peak)
- For galaxies: initialize a separate disk component that is slightly bluer and larger
- In each step:
  - Update the morphology of each object
  - Update the SED of each object
  - Update the position of each object
  - Repeat until convergence
LSST Deblender

Data

Model

SEDs

Peak 0

Peak 1

Peak 2
LSST Deblender

Constraints
A Matrix (SEDs):
  • Normalization to unity (for each peak)

S Matrix (Intensities):
  • Symmetry
  • Monotonicity
  • Sparsity
  • Single pixel (convolved with PSF)
  • Color
  • Smoothness (TV)

SDSS/Current Deblender
Stars
Constraints

180° Symmetry

- Linear Operator: $L_sS = 0$
- Useful for separating objects
Constraints

Monotonicity

- Forces flux to be non-increasing from the peak
- Implementation as a linear operator (radial gradient $\geq 0$) with mixed results
- For now works better as a projection onto the space of decreasing radial flux
Constraints

Sparsity
Usually even Symmetry and Monotonicity aren’t enough
Constraints

Sparsity
Sparsity prevents low level (but still monotonic) creeping of a model
Translations and Convolutions

**Transformation Operators**

\[
\Gamma = T_Y P T_X
\]

\[
T_X = \begin{pmatrix}
1 - dx & dx & 0 & \ldots & 0 \\
0 & \ddots & \ddots & \ddots & \vdots \\
\vdots & \ddots & \ddots & \ddots & \vdots \\
0 & \ldots & 0 & 1 - dx & \vdots \\
\end{pmatrix}
\]

\[
T_Y = \begin{pmatrix}
1 - dx & 0 & \ldots & 0 & dx & 0 & \ldots & 0 \\
0 & \ddots & \ddots & \ddots & \vdots & \vdots & \ddots & \vdots \\
\vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \ddots & \vdots \\
0 & \ldots & 0 & 1 - dx & \vdots & \vdots & \ddots & \vdots \\
\end{pmatrix}
\]

\[
dx = 0.75 \quad \text{dy} = 0.25\]
Translations and Convolutions

Example with inexact positions
Fitting positions

- Calculate the difference images in x and y for each peak, in each band
- Simultaneously fit the peak positions
- Roughly:
  
  Peak Model + $\alpha$ Diff $\text{Im}_x$ + $\beta$ Diff $\text{Im}_y$
Translations and Convolutions

Example with inexact positions
Translations and Convolutions

Same example with position fitting
Translations and Convolutions

Same example with position fitting
Results

Simulated Data
More Complicated Blends

Simulated Data
Results: HSC Data
Results: HSC Data (2 Components Each)
Results: HSC Data (2 Components when necessary)
Results: HSC Data (single component)
Future Plans
Simultaneously Fit entire CCD

- Simultaneously fit the entire CCD by subdividing the image for each object
  - Current runtime $O(K \times N^2)$
  - Future runtime $O(K \times N_k^2)$, $N_k \ll N$

- Model stars and galaxies with different constraints

- Attempt to implement ideas from crowded field codes (kind of):
  - Simultaneously fit *all* point sources in the image
  - Subtract the detected objects
  - Detect new peaks in the image
  - Repeat all of the above
  - Might be too computationally expensive in practice
LSST Deblender

Algorithmic Questions

How good will our PSF’s be in crowded Fields?

How will we handle multiple epochs?

- Baseline: produce a coadd for each epoch and ignore temporal variation
- Running on all epochs simultaneously is essentially forced photometry (if we assume morphology is constant across epochs)
- Proper motions can help us model moving objects (also star-galaxy separator?)

What is the optimal set of constraints?

- How do we decide to model each object?
- What constraints do we use for different object types?
Conclusion

- On simulated data and a small subset of HSC data, we see significant improvements in color recovery and stability over the SDSS deblender.
- Within the next week we are beginning to test the deblender on a much larger subset of HSC data to identify the strengths and weaknesses of the algorithm.
- We want to know from you all what measurements you would like to see in future tests.
- Optimization is needed to improve the speed of the algorithm (the HSC field shown in the examples takes ~2 min to process with multiple components).
- This is still a work in progress, if you are interested in special use cases, get involved.