Using Sparse Matrices to Speed Up Poisson Image Editing

Yang Zhuoyu

Works before

- Implement Poisson Image Editing in Python and the result looks good.

Direct

Opaque

Transparent

Problems

- Since we have already got good result, now it's time for us to speed it up. The most time-consuming part in my code is to solve Ax=b, where A is quite a sparse matrix.

$$4f_p - f_{q1} - f_{q2} - f_{q3} - f_{q4} = 4g_p - g_{q_1} - g_{q_2} - g_{q_3} - g_{q_4}$$

Problems

- At First, I just use a full matrix A and use np.linalg.lstsq to solve Ax=b.
- To calculate the value of 26431 pixels, it takes 32 minutes.

Direct

Opaque

Transparent

Sparse Matrices in Scipy

- bsr_matrix: Block Sparse Row matrix
- coo_matrix: COOrdinate format matrix
- csc_matrix: Compressed Sparse Column matrix
- csr_matrix: Compressed Sparse Row matrix
- dia_matrix: Sparse matrix with DIAgonal storage
- dok_matrix: Dictionary Of Keys based sparse matrix.
- lil_matrix: Row-based linked list sparse matrix

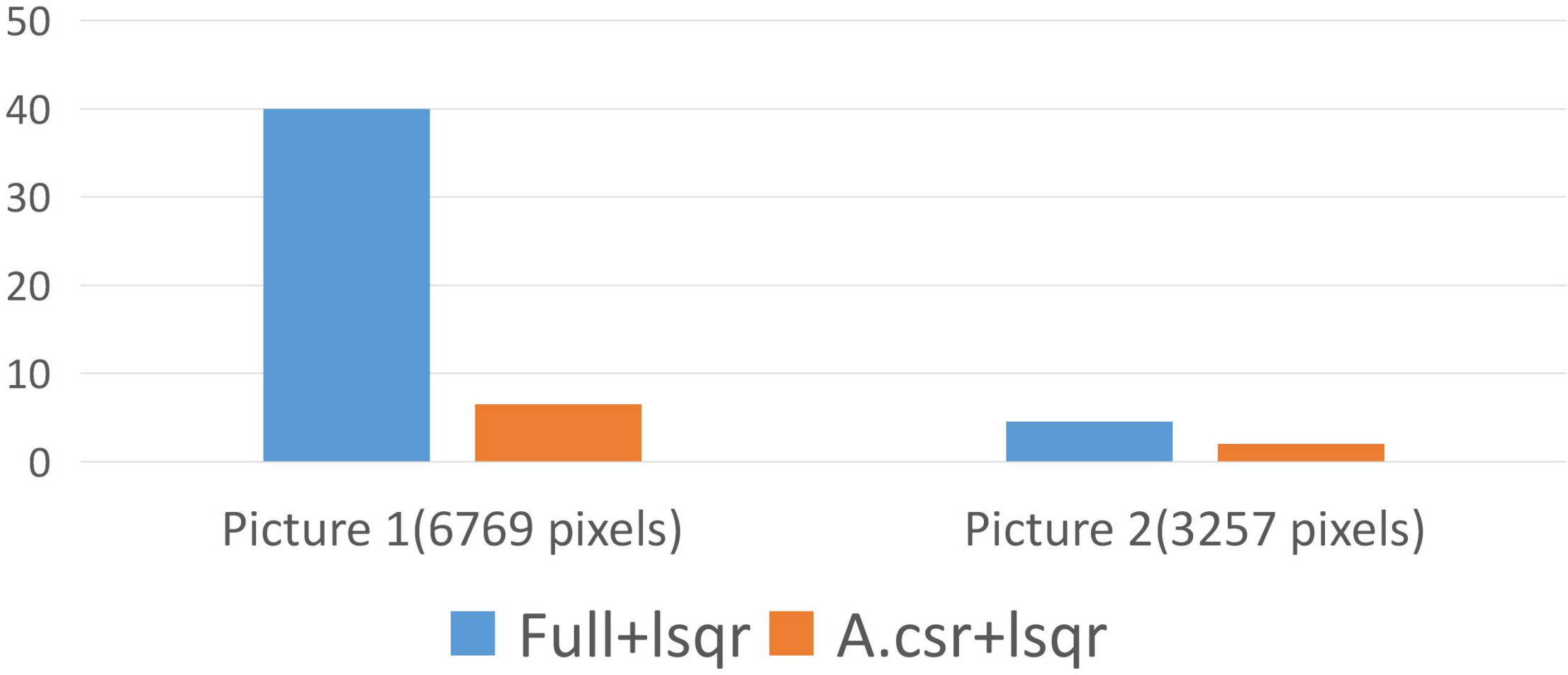
Solving Linear Equation in Scipy

- spsolve
- bicg - BIConjugate Gradient iteration
- bicgstab - BIConjugate Gradient STABilized iteration
- cg - Conjugate Gradient iteration
- cgs - Conjugate Gradient Squared iteration
- gmres - Generalized Minimal RESidual iteration
- lgmres - LGMRES algorithm
- ……

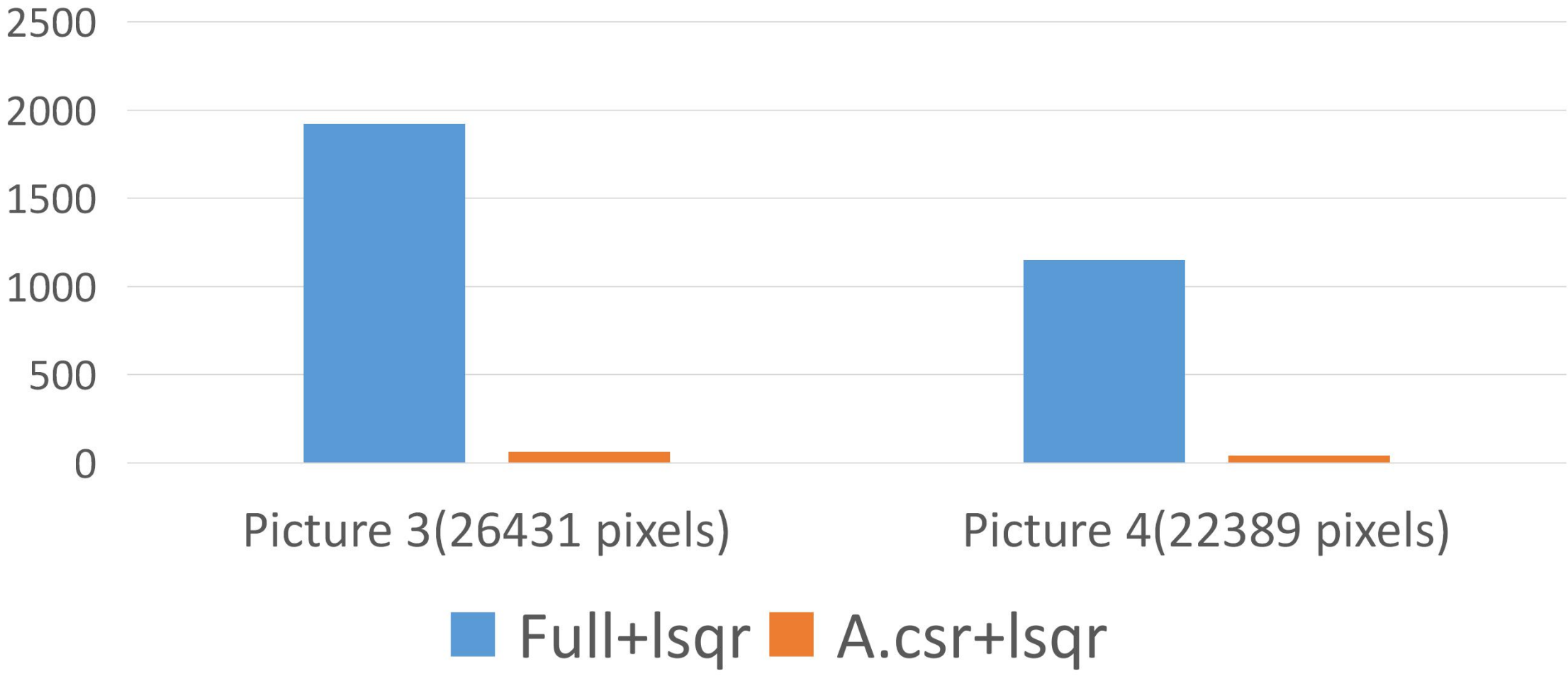
Experiment 1

- Fisrt, use csr_matrix in scipy.sparse to compress the full matrix A.
- Then use lsqr from scipy.sparse.linalg to solve Ax = b.
- Show the result compared with full matrix A and lsqr.
- Picture1 has 6769 pixels to calculate
- Picture2 has 3257 pixels to calculate
- Picture3 has 26431 pixels to calculate
- Picture4 has 22389 pixels to calculate

Time for different methods(time unit: second)



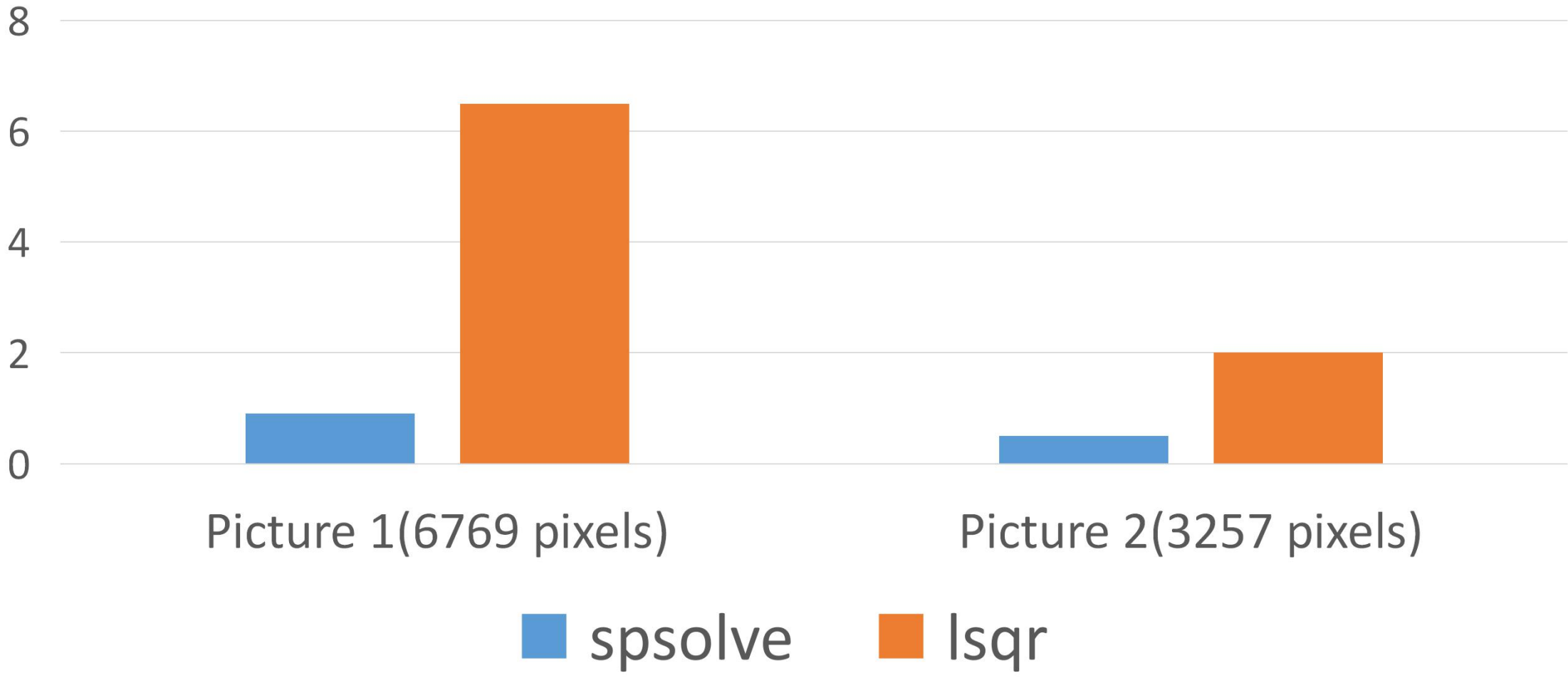
Time for different methods(time unit: second)



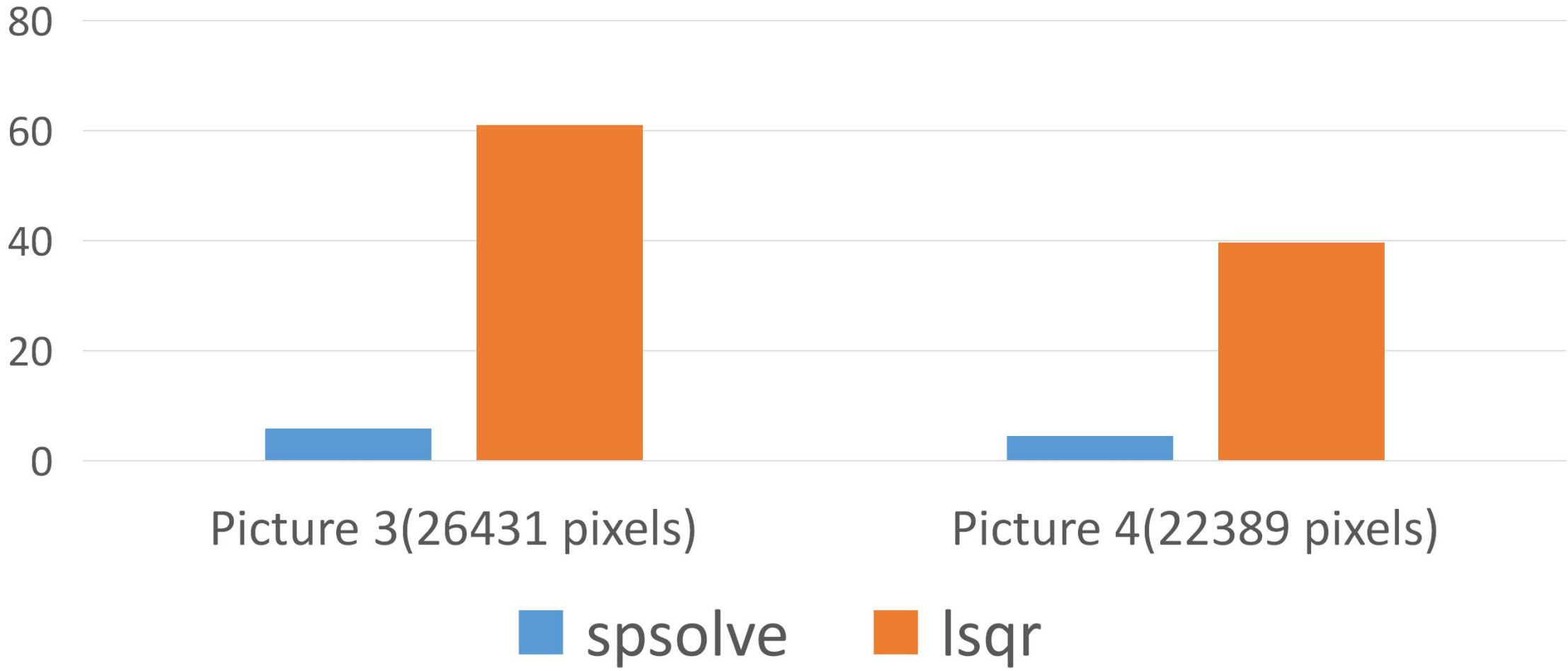
However…

- After we use compressed sparse row matrix to compress the matrix A, the performance of Poisson image editing improves a lot.
- However, with compressed matrix A, can we try to use other solving method instead of the least-squares solution? Maybe much more improvement.

Experiment 2 Use scipy.sparse.linalg.spsolve



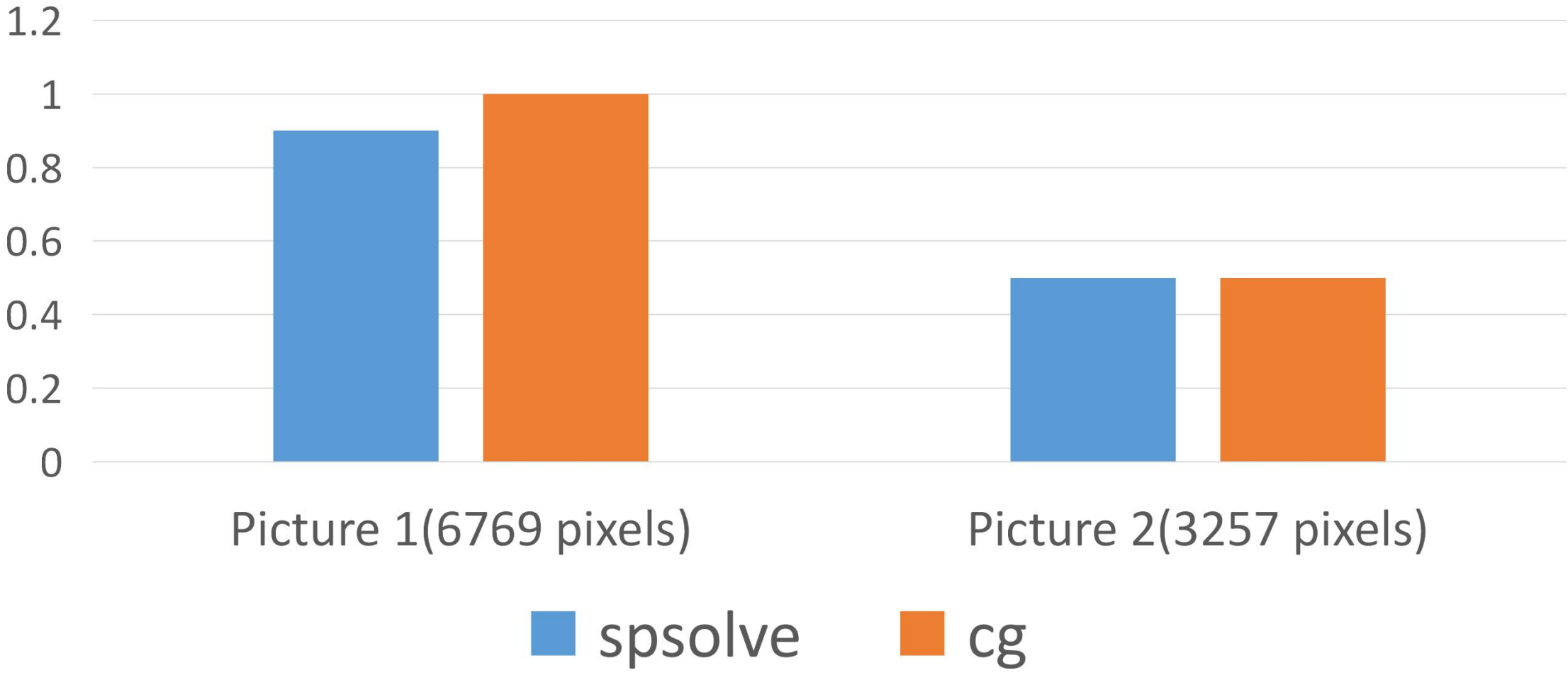
Experiment 2 Use scipy.sparse.linalg.spsolve



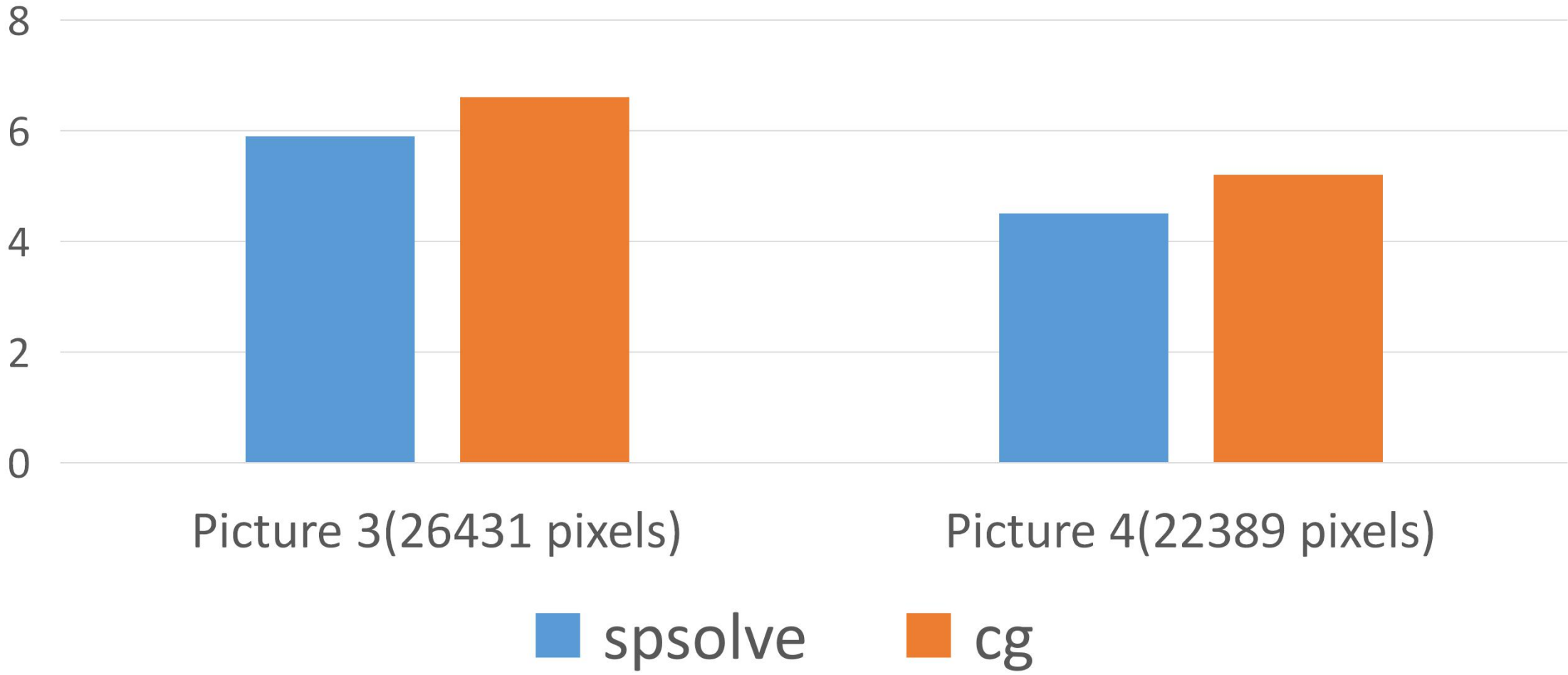
Comparision

- Spsolve performs much better than lsqr.
- But why? What happened inside Spsolve?
- Let's compare the performance of spsolve with Conjugae Gradient(cg).

Experiment 3 Compare spsolve and cg



Experiment 3 Compare spsolve and cg



Conjugate Gradient

- The performance of Conjugate Grandient quite close with spsolve.
- Inside spsolve, it may choose cg to solve this Ax = b.
- In this Poisson Image Editing, the matrix A is a symmetric matrix. If A is symmetric, LSQR should not be used! Alternatives are the symmetric conjugate-gradient method (cg) and/or SYMMLQ. SYMMLQ is an implementation of symmetric cg that applies to any symmetric A and will converge more rapidly than LSQR. If A is positive definite, there are other implementations of symmetric cg that require slightly less work per iteration than SYMMLQ (but will take the same number of iterations).

Note

- 1.Scipy has bug on import lsqr. Therefore, use

```
from scipy.sparse.linalg import lsqr
channel0 = lsqr(A, b[:,0])[0]
```

- Instead of

```
from scipy import sparse
channel0 = sparse.linalg.lsqr(A, b[:,0])[0]
```

Source code on my GitLab

Thanks

- Thanks for the kind instruction of Matsushita Sensei and the inspiring help from Iwata Senpai.