CP1 - Image Deblurring using Least Squares Approximation, QR Factorization, and Householder Reflectors

Reese Karo, William Mahnke, Raaghav Thatte

Overview

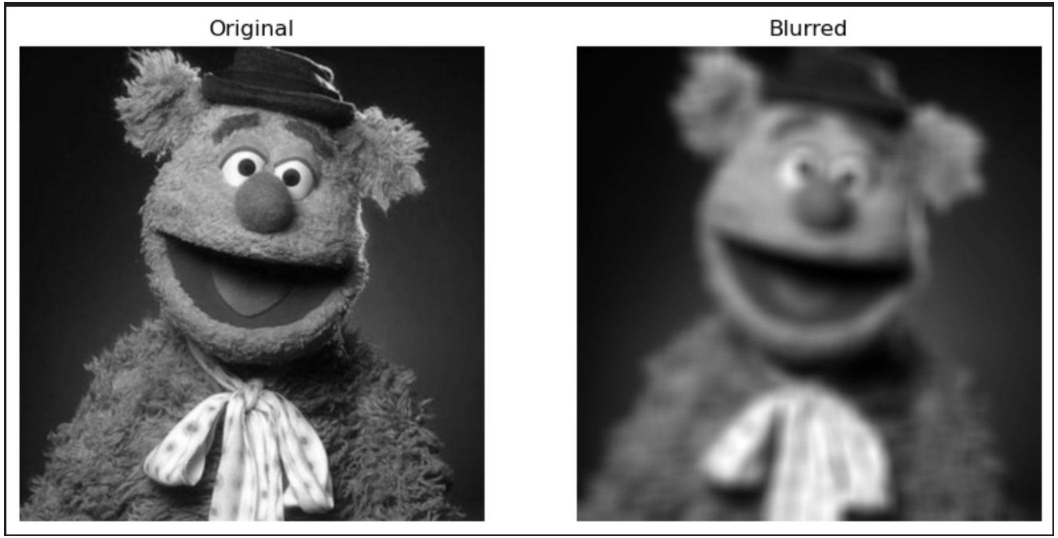
- **Goal**: Blur an image using OpenCv, and Numpy, and then **transform** the image back to the original using various methods of least squares approximation

Concepts Used

- **Least Squares Approximation**
- **QR Factorization**
- **Householder Reflectors**

Method: Blurring the Image

- Check that least squares is applicable
  - Fozzie Bear image is a 678 x 664 array → m > n, so we can use least squares
- Convert image to black and white
- Apply OpenCV's cv.blur in python (here, blurring is set to (25, 25))



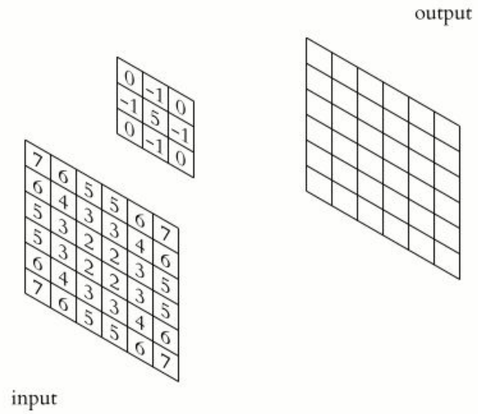
Blurring Effect (Smoothing):

- The blurring effect comes from convolving our image matrix with a kernel K, which is know as a normalized box filter. K is defined as:

$$K = \frac{1}{x * y} \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 1 \end{bmatrix}$$

- Matrix convolution of K and A is defined as:

$$K * A(x, y) = \sum_{i=-x}^{x} \sum_{j=-y}^{y} K(i, j) A(x - i, y - j)$$



Method: Applying Least Squares

- **Least Squares Approximation:** Given an m x n matrix A and vector b of length m, we aim to solve the normal equation $A^T A\bar{x} = A^T b$

Our "target" vector is the original image, which is an **m x n array of values** as opposed to a **vector of length m**.

In order to properly utilize least squares, we will **iterate through the columns of the original image matrix** to create each individual column of our solution matrix.

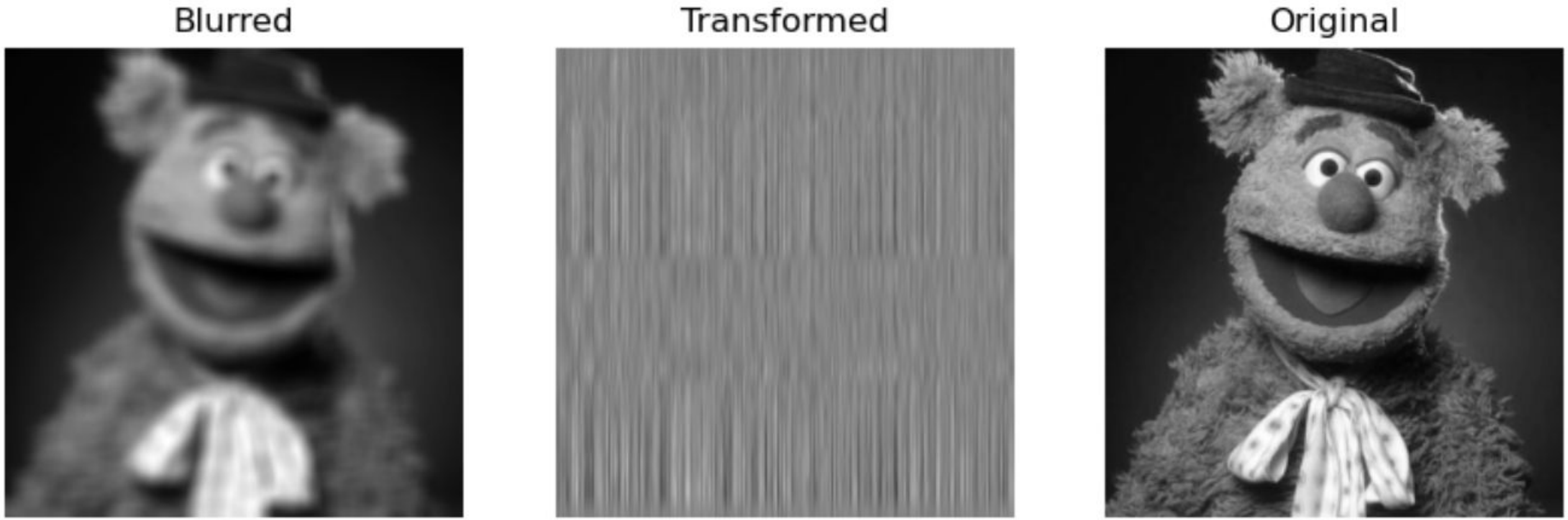
Our solution matrix is an n x n matrix.

Method: Applying Least Squares

(via Numpy's Least Squares function)

The resulting image is nearly identical, with very minor issues in shading.

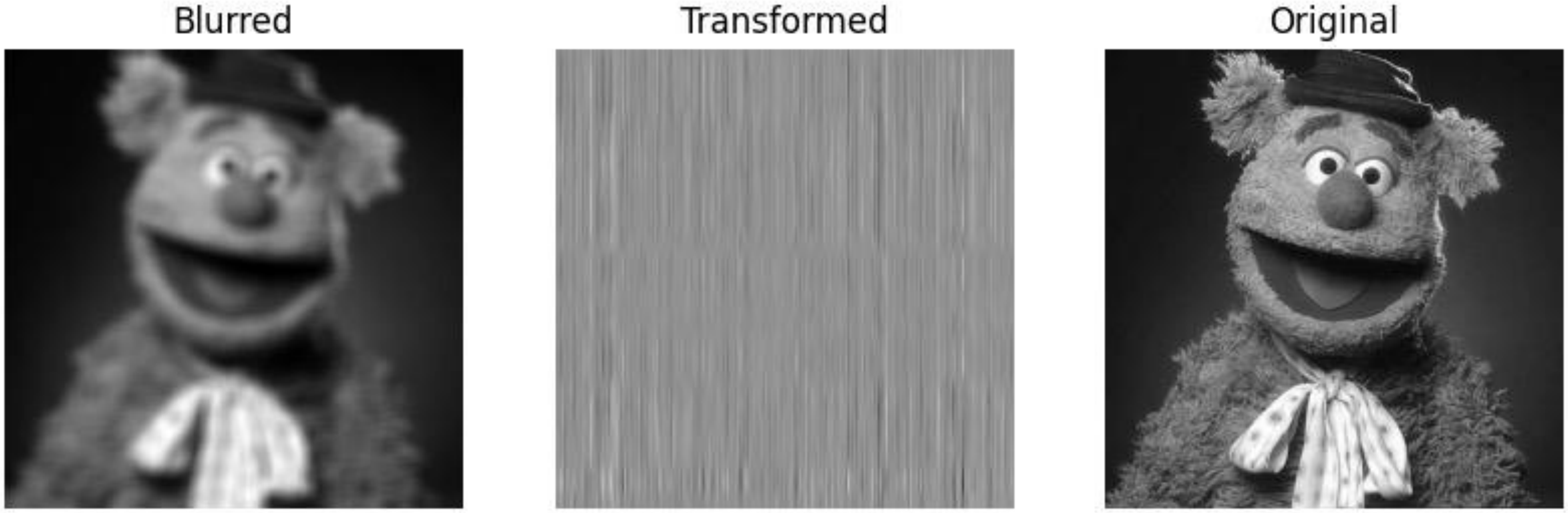
Method: Normal Equation



(via our own hard code)

Normal equation with the original blurring at (25, 25)

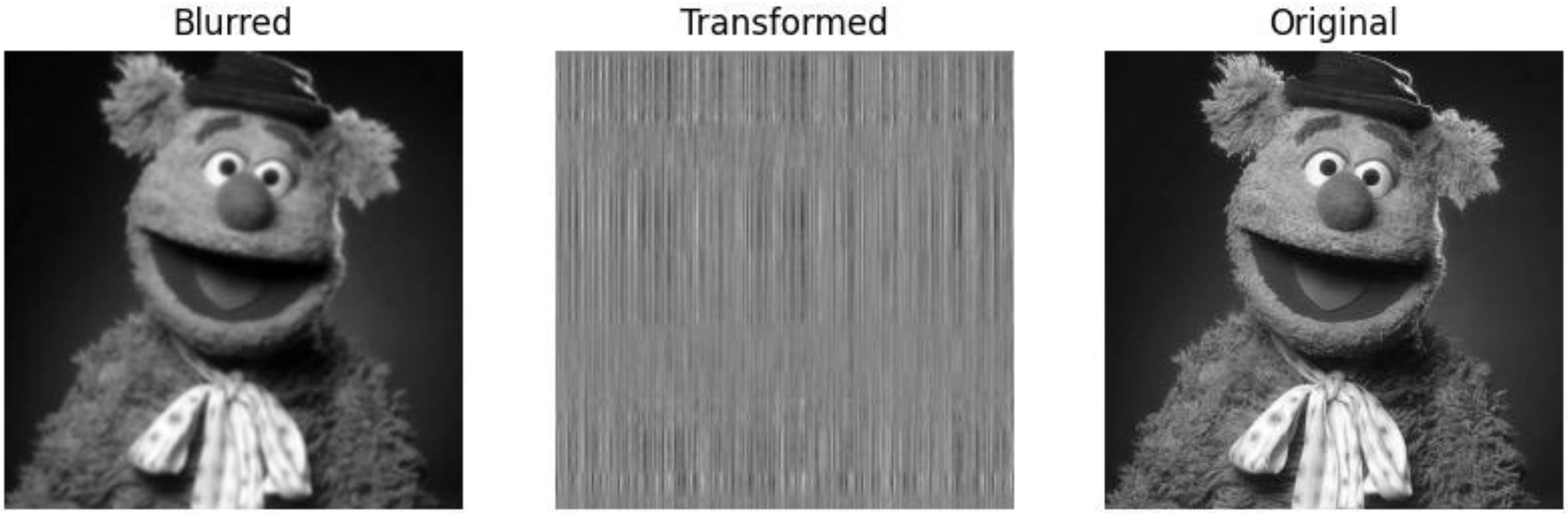
Method: Normal Equation



(via our own hard code)

The blurring in this case was set to (15, 15).

Method: Normal Equation



We then set the blurring on the original image to (5,5).

Clearly, the matrix is ill-conditioned, and the normal equation is NOT a suitable equation for this task.

Method: QR Factorization

Modified Gram-Schmidt:

Given $A_j, j = 1, \ldots, n$ be linearly independent vectors.

- **for** $j = 1, 2, \ldots, n$
  - $y = A_j$
  - **for** $i = 1, 2, \ldots, j-1$
    - $r_{ij} = q_i^T y$
    - $y = y - r_{ij} q_i$
  - $r_{jj} = \|y\|_2$
  - $q_j = y / r_{jj}$

Gram-Schmidt:

Given A_j $(j = 1, \cdots, n)$ that are linearly independent

- **for** $j = 1, 2, \ldots, n$
  - $y = A_j$
  - **for** $i = 1, 2, \ldots, j-1$
    - $r_{ij} = q_i^T A_j$
    - $y = y - r_{ij} q_i$
  - $r_{jj} = \|y\|_2$
  - $q_j = y / r_{jj}$

- QR factorization gives an orthogonal matrix Q and upper-triangular matrix R
- Provides a more numerically stable way to compute systems of equations

Method: QR Factorization

- Again, in order to create a least squares solution matrix, we need to iterate among the column vectors of our original matrix to create the column vectors of our solution matrix.

- It is worth noting that runtime of both Reduced and Full QR factorizations were **17 and 20 minutes, respectively**. Numpy's qr package was used to speed up the calculations.

Method: Reduced QR Factorization

Method: Full QR Factorization

Method: Householder Reflectors

Define the Householder Reflector $H_1 = I - \frac{v^T v}{v v^t}$, where $v = w - x, \;\; w = \pm \left(\|x_1\|_2, 0, \ldots, 0 \right)$

Recursively multiply the Householder transformation $A = H_1 H_2 ... H_k = QR$

Because we again needed to iterate among the column vectors of the original image matrix to obtain our solution matrix, and the already existing nested loop in the Householder function, **the final runtime for the Householder method was 162 minutes.**

Method: Householder Reflectors

The resulting image is nearly identical to the original.

Takeaways

- Numpy’s least squares function and the QR factorization proved to be the most efficient methods.
- The Householder method proved to be very accurate but had an extensively long runtime.
- The normal equation failed even with reduced blurriness, implying that the image matrix is ill-conditioned.

What next?

- Single-valued decomposition (SVD)
- Neural networks (NN)
- Repeated QR and Householder Reflectors

Thank you!