18-660: Numerical Methods for Engineering Design and Optimization

Xin Li
Department of ECE
Carnegie Mellon University
Pittsburgh, PA 15213

Electrical & Computer ENGINEERING

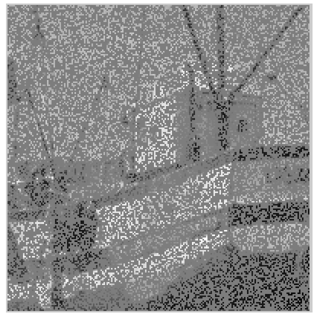
Outline

- **Project 2: Image Recovery**
  - Objective
  - Methods
  - Submission details

Objective

- Apply compressed sensing to recover a full image from a small number of sampled pixels

Original image



Sampled image

Recovered image

Application

- Corrupted image recovery

Corrupted image

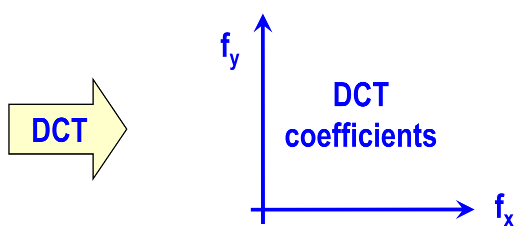
Recover

Recovered image

Outline

- Objective
- Methods
- Submission details

2-D DCT Transform



$$\underbrace{g(x,y)}_{\text{Image pixel}}=\sum_{u=1}^{P}\sum_{v=1}^{Q}\underbrace{\alpha_u\cdot\beta_v\cdot\cos\frac{\pi(2x-1)(u-1)}{2\cdot P}\cdot\cos\frac{\pi(2y-1)(v-1)}{2\cdot Q}}_{\text{Transformation}}\cdot\underbrace{G(u,v)}_{\text{DCT coefficient}}$$

$$\underbrace{\begin{bmatrix}\times\\\times\\\times\\\times\\\times\end{bmatrix}}_{\mathbf{C}}=\underbrace{\begin{bmatrix}\times&\times&\times&\times&\times\\\times&\times&\times&\times&\times\\\times&\times&\times&\times&\times\\\times&\times&\times&\times&\times\\\times&\times&\times&\times&\times\end{bmatrix}}_{\mathbf{T}}\cdot\underbrace{\begin{bmatrix}\times\\\times\\\times\\\times\\\times\end{bmatrix}}_{\boldsymbol{\alpha}}$$

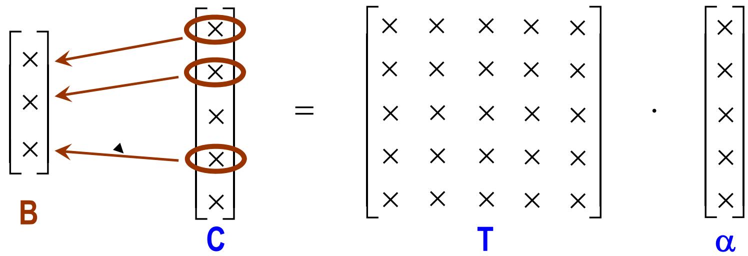
$$x,u\in\{1,2,\cdots,P\}$$

$$y,v\in\{1,2,\cdots,Q\}$$

$$\alpha_u=\begin{cases}\sqrt{1/P} & (u=1)\\ \sqrt{2/P} & (2\le u\le P)\end{cases}$$

$$\beta_v=\begin{cases}\sqrt{1/Q} & (v=1)\\ \sqrt{2/Q} & (2\le v\le Q)\end{cases}$$

Compute DCT Coefficients



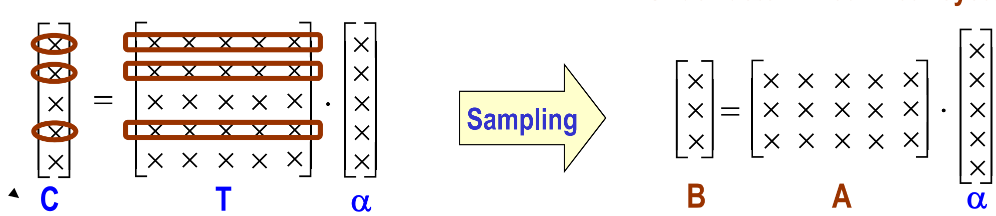
- **Transformation matrix T is known**

- **How to compute DCT coefficients?**
  - If the full image C is given: $\alpha = T^{-1} \cdot C$
  - If only a few samples of C (sampled vector B) is given, *can we get an approximate solution of α?*

Compute DCT Coefficients

- **Sampled image leads to an underdetermined linear system:** **B = A ·α**

Under-determined linear system



- **Compute DCT coefficients α by** *solving an underdetermined linear system*: **B = A ·α**

- **Once DCT coefficients are computed, recover the full image by C = T·α**

Image Recovery

- **Apply compressed sensing to determine approximate DCT coefficients from sampled pixels**
- **Apply inverse DCT transform to recover the full image**

Compressed Sensing

$$\underbrace{\begin{bmatrix} \times \\ \times \\ \times \end{bmatrix}}_{\mathbf{B}} = \underbrace{\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{bmatrix}}_{\mathbf{A}} \cdot \underbrace{\begin{bmatrix} \mathbf{0} \\ \times \\ \times \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}}_{\alpha \text{ (Sparse)}}$$

- **Solve the underdetermined linear system: B = A · α**
- **If α is sparse:**
  - Find the sparse solution α by L_0-norm regularization

$$\min_{\alpha} \quad \|A\alpha - B\|_2^2$$
$$\text{S.T.} \quad \|\alpha\|_0 \le \lambda$$

Implementation Issues

- α is expected to be sparse
- How to solve the optimization with L_0-norm regularization?
- How to determine λ?

Construct Sparse DCT Coefficients

- DCT coefficients of a large image are often not sparse
- Solution: break a large image into small blocks
  - Each small block corresponds to few non-zero DCT coefficients
  - You can try different block size *K* in your own experiments
  - 8x8 block is suggested for the small test image
  - 16x16 block is suggested for the large test image

Solve L_0-norm Regularization

- **Use Orthogonal Matching Pursuit (OMP) to solve:**

$$\min_{\alpha} \quad \|A\alpha - B\|_2^2 \qquad \text{S.T.} \quad \|\alpha\|_0 \le \lambda$$

  - Step 1: Set F = B, $\Omega = \{\ \}$ and p = 1
  - Step 2: Calculate inner product values $\theta_i = \langle A_i, F \rangle$
  - Step 3: Identify the index s for which $|\theta_s|$ takes the largest value
  - Step 4: Update Ω by $\Omega = \Omega \cup \{s\}$
  - Step 5: Approximate F by the linear combination of $\{A_i;\ i \in \Omega\}$

$$\min_{\alpha_i, i\in\Omega} \left\| \sum_{i\in\Omega} \alpha_i \cdot A_i - B \right\|_2^2$$

  - Step 6: Update F

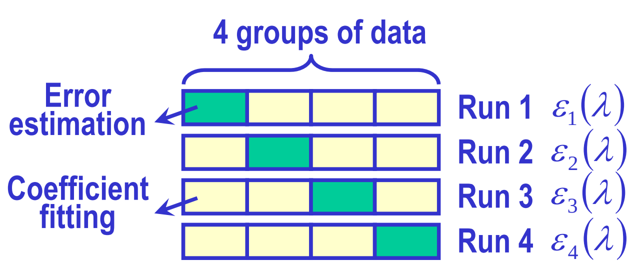
$$F = B - \sum_{i\in\Omega} \alpha_i \cdot A_i$$

  - Step 7: If p < λ, p = p+1 and go to Step 2. Otherwise, stop.

$$\alpha_i = 0 \quad (i \notin \Omega)$$

Determine λ by Cross Validation

- **For each λ in a given list**
  - Calculate DCT coefficients from the training set
  - Estimate approximation "error" from the testing set
    - Use mean square error as a measurement of the "error"
  - Example: 4-fold cross validation



$$Error(\lambda) = [\varepsilon_1(\lambda) + \varepsilon_2(\lambda) + \varepsilon_3(\lambda) + \varepsilon_4(\lambda)]/4$$

- **Select λ with the minimum error**

Cross Validation with Random Subsets

- **N-fold cross validation**
  - Distribute all data into N folds such that the numbers of samples in all folds are equal
  - Take one fold as the test set at each iteration
  - Training-and-test process is repeated for N times
- **Cross validation with random subsets**
  - At each iteration, randomly draw ***m*** samples to form the test set and use all other samples as the training set
    - Use ***m = floor(S/6)*** in this project, where ***S*** is the total number of samples
  - Repeat the training-and-test process for M times
    - Use ***M = 20*** in this project
- **Apply cross validation with random subsets in this project**
  - When the data set is small, using this method with large M is more accurate than N-fold cross validation
  - It is easy to implement if the data set cannot be divided equally into N folds

Median Filter

- **Median filtering is to replace each pixel in an image by the median of its neighborhood**
- **Median filtering algorithm where filter size is m×n:**
  - Sort all pixel values in an m×n block, centered at (x,y), to find the median
  - Replace the pixel value f(x,y) by the median

123	125	126	130	140
122	124	126	127	135
118	120	150	125	134
119	115	119	123	133
111	116	110	120	130

Neighbourhood values:

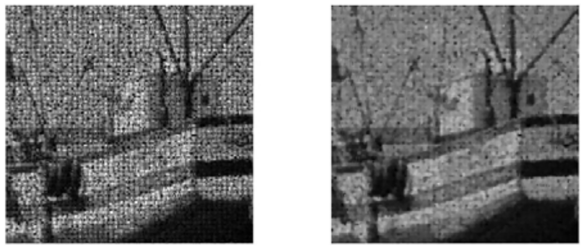
115, 119, 120, 123, 124,
125, 126, 127, 150

Median value: 124

Median Filter

- **Apply median filter (MF) to improve the quality of recovered images**
  - You can use MATLAB function medfilt2
  - Set filter size 3×3

- **Compare the error of the recovered image w/ MF and w/o MF**



Recovered images (4×4 block)
(Left: w/o median filter, Right: w/ median filter)

Summary of Image Recovery

- **Read in an image**
- **Break image into blocks where block size is K×K**
- **For each block:**
  - Randomly sample a few pixels where sample size is *S*
    - *No repetition* for each pixel
  - Compute DCT coefficients from the samples
    - Use OMP algorithm
    - λ Is determined by cross validation using random subsets
  - Apply inverse DCT transform to recover the block
- **Combine all recovered blocks into a full image**
  - Apply median filter to improve image quality

Critical Functions

- **The following three functions are critical for your implementation and grading:**
  - Data sampling
  - OMP solver
  - Optimal λ selection via cross validation

Quality Measurement

- **Mean square error between recovered image and original image is used to measure the quality of recovery**
  - $$\frac{1}{W \times H} \sum_{\substack{1 \le x \le W \\ 1 \le y \le H}} \left[\hat{g}(x, y) - g(x, y)\right]^2$$
  - W : image width

    H : image height
  - $\hat{g}(x, y)$: pixels of recovered image

    $g(x, y)$: pixels of original image

Outline

- Objective
- Methods
- **Submission details**

Project Files

All files for this project can be found from the distributed package

- **A report template**
  - Proj2.doc
- **Two test images**
  - fishing_boat.bmp
  - lena.bmp
- **Three MATLAB functions**
  - imgRead.m
  - imgShow.m
  - imgRecover.m

MATLAB Functions

- **imgRead.m**
  - Load test image: *e.g. A = imgRead('lena.bmp')*
  - Input: input file name
  - Output: *H*-by-*W* matrix
    - *A(i,j)*: image pixel at i-th row and j-th column
    - *W*: image width
    - *H*: image height
- **imgShow.m**
  - display image: *e.g. imgShow(B);*
  - Input: *H*-by-*W* matrix
    - *B(i,j)*: image pixel at i-th row and j-th column
    - *W*: image width
    - *H*: image height

MATLAB Functions

- **imgRecover.m**
  - Should be implemented by you
  - Syntax: *imgOut = imgRecover(imgIn, blkSize, numSample)*
  - Input:
    - imgIn: *H*-by-*W* input matrix
    - blkSize: block size, i.e., *K* on *Slide* 12
    - numSample: number of samples per block, i.e., *S* on *Slide* 15
  - Output: *H*-by-*W* matrix
    - Recovered image without median filtering

Project Submission

- **You should zip the MATLAB code (.m) and figures (.fig) into a single file and submit it to the course web site**
  - Your code must work on Linux cluster without any modification
  - Follow instructions specified in the WORD template

- **You should also submit a PDF report (at most 4 pages) to the course web site**
  - Follow instructions specified in the WORD template

Grading Criteria

- **75% for MATLAB code and results**
- **25% for project report**