Lecture 6: Recall:

How to compute DFT fast?

Goal: Convert image I to $\underbrace{\hat{I}}_{\text{DFT}}$ → Manipulate / adjust $\hat{I}$ (Fourier coefficients)
(Fast ?)
to get a new $\hat{I}^{new}$
↓
Convert $\hat{I}^{new}$ into the spatial domain
(Fast ?)

Fast Fourier Transform

Recall: DFT is separable ⇒ 2D DFT = Two 1D DFT!

$$\hat{I}(m,n) = \underbrace{\frac{1}{N}\sum_{k=0}^{N-1}\underbrace{\left(\frac{1}{N}\sum_{l=0}^{N-1} I(k,l)\, e^{-j2\pi\left(\frac{ln}{N}\right)}\right)}_{\text{1D DFT}} e^{-j2\pi\left(\frac{km}{N}\right)}}_{\text{1D DFT}}$$

Suffices to consider how to compute 1D DFT fast!!

1D DFT is: $\hat{f}(u) = \frac{1}{N}\sum_{x=0}^{N-1} f(x)\, \underbrace{e^{-j2\pi\left(\frac{ux}{N}\right)}}_{\omega_N^{ux}}$ where $\omega_N = e^{-j\frac{2\pi}{N}}$

Assume $N = 2^n = 2M$ $\quad (\therefore M = 2^{n-1})$.

Then: $\hat{f}(u) = \frac{1}{2M}\sum_{x=0}^{2M-1} f(x)\, \omega_{2M}^{ux}$

Separate the summation into odd and even parts:

$$\hat{f}(u) = \frac{1}{2}\left\{\frac{1}{M}\sum_{y=0}^{M-1} \underbrace{f(2y)}_{f_{even}(y)}\, \underbrace{\omega_{2M}^{u(2y)}}_{= \omega_M^{uy}} + \frac{1}{M}\sum_{y=0}^{M-1} \underbrace{f(2y+1)}_{f_{odd}(y)}\, \underbrace{\omega_{2M}^{u(2y+1)}}_{\omega_{2M}^{u(2y)}\,\omega_{2M}^{u}}\right\}$$

$\omega_{2M}^{u(2y)} = \omega_M^{uy}$

Let $f_{even} = (f(0), f(2), \dots, f(2M-2))^T$ — even part of f

$f_{odd} = (f(1), f(3), \dots, f(2M-1))^T$ — odd part of f

Then: $\hat{f}(u) = \frac{1}{2}\left\{\hat{f}_{even}(u) + \hat{f}_{odd}(u)\, \omega_{2M}^{u}\right\}$ for $u = 0, 1, 2, \dots, M-1$

only defined for $u = 0, 1, 2, \dots, M-1$

For $u \geq M$, consider: $\hat{f}(u+M) = \frac{1}{2}\left\{\frac{1}{M}\sum_{y=0}^{X-1} f(2y)\, \omega_M^{uy+My} + \frac{1}{M}\sum_{y=0}^{M-1} f(2y+1)\, \underbrace{\omega_M^{uy+My}}_{\omega_M^{uy}}\, \underbrace{\omega_{2M}^{u+M}}_{-\omega_{2M}^{u}}\right\}$

for $u = 0, 1, 2, \dots, M-1$

$\therefore \quad \hat{f}(u+M) = \frac{1}{2}\left\{ \hat{f}_{even}(u) - \hat{f}_{odd}(u)\, W_{2M}^{u} \right\}$ for $u = 0, 1, 2, \ldots, M-1$

FFT algorithm: Let $N = 2^n$ and $f \in \mathbb{R}^N$

Step 1: Split f into:

$$f_{even} = [f(0), f(2), \ldots, f(2M-2)]^T$$
$$f_{odd} = [f(1), f(3), \ldots, f(2M-1)]^T.$$

Step 2: Compute $\hat{f}_{even} = \underbrace{F_M}_{\text{DFT matrix}} f_{even}$ and $\hat{f}_{odd} = F_M f_{odd}$

$M = 2^{n-1}$

$F_M = \left(W_M^{ux} \right)_{0 \le u, x \le M-1}$

$\| \; M \times M$ matrix!

Step 3: For $u = 0, 1, 2, \ldots, M-1$, compute

$$\hat{f}(u) = \frac{1}{2}\left[\hat{f}_{even}(u) + \hat{f}_{odd}(u)\, W_{2M}^{u} \right]$$
$$\hat{f}(u+M) = \frac{1}{2}\left[\hat{f}_{even}(u) - \hat{f}_{odd}(u)\, W_{2M}^{u} \right]$$

$\therefore$ Reduce the matrix multiplication by $\frac{1}{2}$!

For step 2, we can apply the splitting idea again to compute $\hat{f}_{even}$ and $\hat{f}_{odd}$!

Computational cost of FFT:

Let C_M be the computational cost of $F_M \vec{x}$. Then: $C_1 = 1$!!

Clearly, $C_N = 2C_M + 3M$ (2 matrix multiplication by F_M, M multiplication, M additions and M subtractions)

$$\therefore \quad C_{2^n} = 2C_{2^{n-1}} + 3\underset{}{\overset{2^{n-1}}{\overset{\shortparallel}{M}}} \Rightarrow 2^{-n} C_{2^n} = 2^{-(n-1)} C_{2^{n-1}} + 3/2.$$

$$= 2^{-(n-2)} C_{2^{n-2}} + 2(3/2)$$

$$= \vdots$$

$$= \underset{\overset{\shortparallel}{1}}{C_1} + n(3/2)$$

$$\therefore \quad C_{2^n} = 2^n + n2^n(3/2)$$

We conclude that the computational cost C_N is bounded by $KN(\log_2 N)$ (or $O(N\log_2 N)$)

e.g. If $N = 2^{10}$, then $N^2 = 2^{20}$ (Computational cost for conventional matrix multiplication)

For FFT, $N\log_2 N = 2^{10} \cdot 10 < 2^{14} \Rightarrow 2^6$ times faster !!

Mathematics of JPEG

Consider a $N \times N$ image f. Extend f to a $2M \times 2N$ image $\tilde{f}$, whose indices are taken from $[-M, M-1]$ and $[-N, N-1]$.

Define $f(k,l)$ for $-M \leq k \leq M-1$ and $-N \leq l \leq N-1$ such that

$$f(-k-1,-l-1) = f(k,l) \quad \Big\} \text{ Reflection about } (-1/2,-1/2)$$

$$\left.\begin{aligned} f(-k-1,l) &= f(k,l) \\ f(k,l-1) &= f(k,l) \end{aligned}\right\} \text{ Reflection about the axis } k=-1/2 \text{ and } l=-1/2$$

Example:

$l = -1/2$

$l=-3$	$l=-2$	$l=-1$	$l=0$	$l=1$	$l=2$	
9	8	7	7	8	9	$k=-3$
6	5	4	4	5	6	$k=-2$
3	2	1	1	2	3	$k=-1$
3	2	1	1	2	3	$k=0$
6	5	4	4	5	6	$k=1$
9	8	7	7	8	9	$k=2$

$k = -1/2$

$f(-1,1)$
$\parallel$
$f(0,1)$

Reflection about the axis $k = -1/2$.

$f(-2,-2)$
$\parallel$
$f(1,1)$

Reflection about $(-1/2,-1/2)$.

Make the extension as a reflection about $(0,0)$, the axis $k=0$ and the axis $l=0$.

Done by shifting the image by $(\frac{1}{2}, \frac{1}{2})$

After shifting

$$
\begin{array}{cccccc|l}
9 & 8 & 7 & 7 & 8 & 9 & \frac{1}{2}+(-3) \\
6 & 5 & 4 & 4 & 5 & 6 & \frac{1}{2}+(-2) \\
3 & 2 & 1 & 1 & 2 & 3 & \frac{1}{2}+(-1) \\
3 & 2 & 1 & 1 & 2 & 3 & \frac{1}{2}+0 \\
6 & 5 & 4 & 4 & 5 & 6 & \frac{1}{2}+1 \\
9 & 8 & 7 & 7 & 8 & 9 & \frac{1}{2}+2 \\
\hline
\frac{1}{2}+-3 & \frac{1}{2}+-2 & \frac{1}{2}+-1 & \frac{1}{2}+0 & \frac{1}{2}+1 & \frac{1}{2}+2 & \\
\end{array}
$$

k

l

Now, we compute the DFT of (shifted) $\tilde{f}$:

$$
\begin{aligned}
F(m,n) &= \frac{1}{(2M)(2N)} \sum_{k=-M}^{M-1} \sum_{l=-N}^{N-1} f(k,l) e^{-j\frac{2\pi}{2M}m(k+\frac{1}{2})} e^{-j\frac{2\pi}{2N}n(l+\frac{1}{2})} \\
&= \frac{1}{4MN} \sum_{k=-M}^{M-1} \sum_{l=-N}^{N-1} f(k,l) e^{-j(\frac{\pi}{M}m(k+\frac{1}{2})+\frac{\pi}{N}n(l+\frac{1}{2}))} \\
&= \frac{1}{4MN} (\underbrace{\sum_{k=-M}^{-1} \sum_{l=-N}^{-1}}_{A_1} + \underbrace{\sum_{k=-M}^{-1} \sum_{l=0}^{N-1}}_{A_2} + \underbrace{\sum_{k=0}^{M-1} \sum_{l=-N}^{-1}}_{A_3} + \underbrace{\sum_{k=0}^{M-1} \sum_{l=0}^{N-1}}_{A_4}) \\
&\quad f(k,l) e^{-j(\frac{\pi}{M}m(k+\frac{1}{2})+\frac{\pi}{N}n(l+\frac{1}{2}))}
\end{aligned}
$$

After some messy simplication, we can get:

$$
A_1 + A_2 + A_3 + A_4 = \frac{1}{MN} \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} f(k,l) \cos\left[\frac{m\pi}{M}\left(k+\frac{1}{2}\right)\right] \cos\left[\frac{n\pi}{N}\left(l+\frac{1}{2}\right)\right]
$$

Definition: (Even symmetric discrete cosine transform [EDCT])

Let f be a $M \times N$ image, whose indices are taken as $0 \leq k \leq M-1$ and $0 \leq l \leq N-1$. The **even symmetric discrete cosine transform (EDCT)** of f is given by:

$$\hat{f}_{ec}(m,n) = \frac{1}{MN} \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} f(k,l) \cos\left[\frac{m\pi}{M}\left(k+\frac{1}{2}\right)\right] \cos\left[\frac{n\pi}{N}\left(l+\frac{1}{2}\right)\right]$$

with $0 \leq m \leq M-1, 0 \leq n \leq N-1$

Remark:
- Smart idea to get a decomposition consisting only of cosine function (by reflection and shifting!)
- Can be formulated in matrix form
- Again, it is a separable image transformation.

- The inverse of EDCT can be explicitly computed. More specifically, the **inverse EDCT** is defined as:

$$f(k,l) = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} C(m)C(n)\hat{f}_{ec}(m,n)\cos\frac{\pi m(2k+1)}{2M}\cos\frac{\pi n(2l+1)}{2N} \qquad (**)$$

where $C(0)=1, C(m)=C(n)=2$ for $m,n\neq 0$

Also involving cosine functions only!

- Formula (**) can be expressed as matrix multiplication:

$$f = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} \hat{f}_{ec}(m,n)\vec{T}_m\vec{T'}_n^T$$

elementary images under EDCT!

where: $\vec{T}_m = \begin{pmatrix} T_m(0) \\ T_m(1) \\ \vdots \\ T_m(M-1) \end{pmatrix}, \vec{T'}_n = \begin{pmatrix} T'_n(0) \\ T'_n(1) \\ \vdots \\ T'_n(N-1) \end{pmatrix}$ with $T_m(k) = C(m)\cos\frac{\pi m(2k+1)}{2M}$

This is what JPEG does!!

and $T'_n(k) = C(n)\cos\frac{\pi n(2k+1)}{2N}$.

Something similar can be developed:

Definition: (Odd symmetric discrete cosine transform [ODCT])

Let f be a $M \times N$ image, whose indices are taken as $0 \leq k \leq M-1$ and $0 \leq l \leq N-1$. The **odd symmetric discrete cosine transform (ODCT)** of f is given by:

$$\hat{f}_{oc}(m,n) = \frac{1}{(2M-1)(2N-1)} \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} C(k)C(l)f(k,l) \cos\frac{2\pi mk}{2M-1} \cos\frac{2\pi nl}{2N-1}$$

where $C(0) = 1$ and $C(k) = C(l) = 2$ for $k, l \neq 0$, $0 \leq m \leq M-1, 0 \leq n \leq N-1$.

The **inverse ODCT** is given by:

$$f(k,l) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} C(m)C(n)\hat{f}_{oc}(m,n) \cos\frac{2\pi mk}{2M-1} \cos\frac{2\pi nl}{2N-1}$$

where $C(0) = 1, C(m) = C(n) = 2$ if $m, n \neq 0$

Understanding convolution:

Recall: Discrete convolution:

$$\underset{g * I(n,m)}{\overset{\shortparallel}{v(n,m)}} = \underbrace{\sum_{n'=0}^{N-1} \sum_{m'=0}^{N-1} g(n-n', m-m')\, I(n', m')}_{\text{Linear combination of pixel values of } I}$$

In particular, if $g(k,\ell)$ is only non-zero around $(0,0)$, then, $g * I(n,m)$ is a linear combination of pixel value of I around (n,m)!!

Why is DFT useful in imaging:

DFT of convolution:

Recall: $$g * w(n, m) = \sum_{n'=0}^{N-1} \sum_{m'=0}^{N-1} g(n-n', m-m')\, w(n', m')$$

$\left(g, m \in M_{N \times M}(\mathbb{R}) \right)$

Then, the DFT of $g * w(p, q) = MN\, \mathrm{DFT}(g)(p, q)\, \mathrm{DFT}(w)(p, q)$

$\therefore$ DFT of convolution can be reduced to simple multiplication!

Recall: Shift-invariant image transformation = 2D convolution.

$\therefore$ Easy computation/manipulation of shift-invariant transf. after DFT!!

Proof:

DFT of $g*w$ at (p,q)

$$= \frac{1}{NM}\sum_{n=0}^{N-1}\sum_{m=0}^{M-1} g*w(n,m)\, e^{-j2\pi\left(\frac{pn}{N}+\frac{qm}{M}\right)}$$

$$= \frac{1}{NM}\sum_{n'=0}^{N-1}\sum_{m'=0}^{M-1}\sum_{n=0}^{N-1}\sum_{m=0}^{M-1} g(n-n', m-m')\, w(n',m')\, e^{-j2\pi\left(\frac{pn}{N}+\frac{qm}{M}\right)}$$

Change of variables:

$n \to n'' = n - n'$

$m \to m'' = m - m'$

$$= \underbrace{\frac{1}{NM}\sum_{n'=0}^{N-1}\sum_{m'=0}^{M-1} w(n',m')\, e^{-j2\pi\left(\frac{pn'}{N}+\frac{qm'}{M}\right)}}_{\hat{w}(p,q)} \underbrace{\sum_{n''=-n'}^{N-1-n'}\sum_{m''=-m'}^{M-1-m'} g(n'',m'')\, e^{-j2\pi\left(\frac{pn''}{N}+\frac{qm''}{M}\right)}}_{T(p,q)}$$

$$\sum_{n''=-n'}^{N-1-n'}\sum_{m''=-m'}^{M-1-m'} = \sum_{n''=0}^{N-1}\sum_{m''=0}^{M-1}$$