CS 4495 Computer Vision

*Frequency2: Sampling and Aliasing*

Aaron Bobick
School of Interactive Computing
Administrivia

- Project 1 is due tonight. Submit what you have at the deadline.

- Next problem set – stereo – will be out Thursday Sept. 11 and will be due Tuesday evening, Sept 23rd, 11:55pm.
  - It is easier…

- Readings for this week: Still FP Chapter 4
Salvador Dali
“Gala Contemplating the Mediterranean Sea, which at 30 meters becomes the portrait of Abraham Lincoln”, 1976
Fourier: A nice set of basis

Teases away fast vs. slow changes in the image.
We want to understand the frequency $\omega$ of our signal. So, let’s reparametrize the signal by $\omega$ instead of $x$:

$$f(x) \rightarrow \text{Fourier Transform} \rightarrow A\sin(\omega x + \phi) \rightarrow F(\omega)$$

For every $\omega$ from 0 to $\text{inf}$, (actually $–\text{inf}$ to $\text{inf}$), $F(\omega)$ holds the amplitude $A$ and phase $\phi$ of the corresponding sine.

- How can $F$ hold both? Complex number trick!

$$A = \pm \sqrt{R(\omega)^2 + I(\omega)^2}$$

$$F(\omega) = R(\omega) + iI(\omega)$$

$\phi = \tan^{-1}\left(\frac{I(\omega)}{R(\omega)}\right)$

$\text{Even}$ $\qquad \text{Odd}$

And we can go back:

$$F(\omega) \rightarrow \text{Inverse Fourier Transform} \rightarrow f(x)$$
Frequency actually goes from $-\infty$ to $\infty$. Sinusoid example:

**Even (cos)**

**Odd (sin)**

**Magnitude**
2D Fourier Transforms

- The two dimensional version:

\[
F(u, v) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) e^{-i 2\pi (ux + vy)} \, dx \, dy
\]

- And the 2D *Discrete FT*:

\[
F(k_x, k_y) = \frac{1}{N} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x, y) e^{-i \frac{2\pi (k_x x + k_y y)}{N}}
\]

- Works best when you put the origin of \( k \) in the middle.
Linearity of Sum

\[
\begin{array}{c}
\begin{array}{c}
\text{Image 1} \\
\text{Image 2}
\end{array}
\end{array}
\] +
\[
\begin{array}{c}
\begin{array}{c}
\text{Image 3} \\
\text{Image 4}
\end{array}
\end{array}
\] =
\[
\begin{array}{c}
\begin{array}{c}
\text{Result 1} \\
\text{Result 2}
\end{array}
\end{array}
\]
Extension to 2D – Complex plane

Both a Real and Im version
Examples
Fourier Transform and Convolution

Let \( g = f \ast h \)

Then

\[
G(u) = \int_{-\infty}^{\infty} g(x) e^{-i2\pi ux} \, dx
\]

\[
= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(\tau)h(x - \tau)e^{-i2\pi ux} \, d\tau dx
\]

\[
= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \left[ f(\tau)e^{-i2\pi ux} \, d\tau \right] h(x - \tau)e^{-i2\pi u(x-\tau)} \, dx
\]

\[
= \int_{-\infty}^{\infty} \left[ f(\tau)e^{-i2\pi ux} \, d\tau \right] \int_{-\infty}^{\infty} h(x')e^{-i2\pi ux'} \, dx'
\]

\[
= F(u)H(u)
\]

Convolution in spatial domain

\( \iff \) Multiplication in frequency domain
Fourier Transform and Convolution

Spatial Domain \((x)\) \hspace{1cm} Frequency Domain \((u)\)

\[ g = f * h \hspace{1cm} G = FH \]

\[ g = fh \hspace{1cm} G = F \times H \]

So, we can find \(g(x)\) by Fourier transform

\[
\begin{align*}
\text{IFT} & \quad \uparrow & \quad \text{FT} & \quad \downarrow \\
G & = & F & \times H
\end{align*}
\]
Example use: Smoothing/Blurring

- We want a smoothed function of $f(x)$

$$g(x) = f(x) \ast h(x)$$

- Let us use a Gaussian kernel

$$h(x) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left[-\frac{1}{2} \frac{x^2}{\sigma^2}\right]$$

- Convolution in space is multiplication in freq:

$$G(u) = F(u)H(u)$$

$H(u)$ **attenuates** high frequencies in $F(u)$ (Low-pass Filter)!
2D convolution theorem example

\[ f(x,y) \]

\[ h(x,y) \]

\[ g(x,y) \]

\[ |F(s_x,s_y)| \]

( or \[ |F(u,v)| \] )

\[ |H(s_x,s_y)| \]

\[ |G(s_x,s_y)| \]
Low and High Pass filtering

Ringing
Fourier Transform smoothing pairs

Spatial domain
\[ f(x) \]

Frequency domain
\[ F(s) = \int_{-\infty}^{\infty} f(x) e^{-i2\pi sx} dx \]
## Properties of Fourier Transform

<table>
<thead>
<tr>
<th></th>
<th>Spatial Domain ((x))</th>
<th>Frequency Domain ((u))</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Linearity</strong></td>
<td>(c_1 f(x) + c_2 g(x))</td>
<td>(c_1 F(u) + c_2 G(u))</td>
</tr>
<tr>
<td><strong>Scaling</strong></td>
<td>(f(ax))</td>
<td>(\frac{1}{</td>
</tr>
<tr>
<td><strong>Shifting</strong></td>
<td>(f(x - x_0))</td>
<td>(e^{-i2\pi u x_0} F(u))</td>
</tr>
<tr>
<td><strong>Symmetry</strong></td>
<td>(F(x))</td>
<td>(f(-u))</td>
</tr>
<tr>
<td><strong>Conjugation</strong></td>
<td>(f^*(x))</td>
<td>(F^*(-u))</td>
</tr>
<tr>
<td><strong>Convolution</strong></td>
<td>(f(x) * g(x))</td>
<td>(F(u)G(u))</td>
</tr>
<tr>
<td><strong>Differentiation</strong></td>
<td>(\frac{d^n f(x)}{dx^n})</td>
<td>((i2\pi u)^n F(u))</td>
</tr>
</tbody>
</table>

**Shrink**: Multiply by \(\frac{1}{|a|}\)

**Stretch**: Multiply by \(|a|\)

**Differentiate**: Multiply by \((i2\pi u)^n\)
## Fourier Pairs (from Szeliski)

<table>
<thead>
<tr>
<th>Name</th>
<th>Signal</th>
<th>Transform</th>
</tr>
</thead>
<tbody>
<tr>
<td>impulse</td>
<td>$\delta(x)$</td>
<td>$1$</td>
</tr>
<tr>
<td>shifted impulse</td>
<td>$\delta(x-u)$</td>
<td>$e^{-j\omega u}$</td>
</tr>
<tr>
<td>box filter</td>
<td>$\text{box}(x/a)$</td>
<td>$a\text{sinc}(a\omega)$</td>
</tr>
<tr>
<td>tent</td>
<td>$\text{tent}(x/a)$</td>
<td>$a\text{sinc}^2(a\omega)$</td>
</tr>
<tr>
<td>Gaussian</td>
<td>$G(x; \sigma)$</td>
<td>$\frac{\sqrt{2\pi}}{\sigma}G(\omega; \sigma^{-1})$</td>
</tr>
<tr>
<td>Laplacian of Gaussian</td>
<td>$(\frac{x^2}{\sigma^4} - \frac{1}{\sigma^2})G(x; \sigma)$</td>
<td>$-\frac{\sqrt{2\pi}}{\sigma}\omega^2G(\omega; \sigma^{-1})$</td>
</tr>
<tr>
<td>Gabor</td>
<td>$\cos(\omega_0 x)G(x; \sigma)$</td>
<td>$\frac{\sqrt{2\pi}}{\sigma}G(\omega \pm \omega_0; \sigma^{-1})$</td>
</tr>
<tr>
<td>unsharp mask</td>
<td>$(1 + \gamma)\delta(x) - \gamma G(x; \sigma)$</td>
<td>$\frac{(1 + \gamma)}{\sqrt{2\pi}}\frac{1}{\sigma}G(\omega; \sigma^{-1})$</td>
</tr>
<tr>
<td>windowed sinc</td>
<td>$r\cos(x/(aW))$ \hspace{1cm} $\text{sinc}(x/a)$</td>
<td>(see Figure 3.29)</td>
</tr>
</tbody>
</table>
Fourier Transform Sampling Pairs

FT of an “impulse train” is an impulse train.
Sampling and Aliasing
Sampling and Reconstruction
Sampled representations

- How to store and compute with continuous functions?
- Common scheme for representation: samples
  - write down the function’s values at many points
Reconstruction

• Making samples back into a continuous function
  • for output (need realizable method)
  • for analysis or processing (need mathematical method)
  • amounts to “guessing” what the function did in between
1D Example: Audio

low frequencies

high frequencies
Sampling in digital audio

- Recording: sound to analog to samples to disc
- Playback: disc to samples to analog to sound again
  - how can we be sure we are filling in the gaps correctly?
Sampling and Reconstruction

- Simple example: a sign wave
Undersampling

- What if we “missed” things between the samples?
- Simple example: undersampling a sine wave
  - unsurprising result: information is lost
Undersampling

- What if we “missed” things between the samples?
- Simple example: undersampling a sine wave
  - unsurprising result: information is lost
  - surprising result: indistinguishable from lower frequency
Undersampling

- What if we “missed” things between the samples?
- Simple example: undersampling a sine wave
  - unsurprising result: information is lost
  - surprising result: indistinguishable from lower frequency
  - also was always indistinguishable from higher frequencies
  - *aliasing*: signals “traveling in disguise” as other frequencies
Aliasing in video

Imagine a spoked wheel moving to the right (rotating clockwise). Mark wheel with dot so we can see what’s happening.

If camera shutter is only open for a fraction of a frame time (frame time = 1/30 sec. for video, 1/24 sec. for film):

Without dot, wheel appears to be rotating slowly backwards! (counterclockwise)
Aliasing in images

Disintegrating textures
What’s happening?

Input signal:

Plot as image:

x = 0:.05:5; imagesc(sin((2.^x).*x))

Alias!
Not enough samples
Antialiasing

• What can we do about aliasing?

• Sample more often
  • Join the Mega-Pixel craze of the photo industry
  • But this can’t go on forever

• Make the signal less “wiggly”
  • Get rid of some high frequencies
  • Will loose information
  • But it’s better than aliasing
Preventing aliasing

- Introduce lowpass filters:
  - remove high frequencies leaving only safe, low frequencies
  - choose lowest frequency in reconstruction (disambiguate)
(Anti)Aliasing in the Frequency Domain
Impulse Train

Define a *comb* function (impulse train) in 1D as follows:

\[
comb_M[x] = \sum_{k=-\infty}^{\infty} \delta[x - kM]
\]

where \(M\) is an integer.
Impulse Train in 1D

\[ \text{comb}_2(x) \]

\[ \frac{1}{2} \text{comb}_{\frac{1}{2}}(u) \]

Remember:

Scaling \( f(ax) \)

\[ \frac{1}{|a|} F \left( u \frac{1}{a} \right) \]
Impulse Train in 2D (bed of nails)

\[ \text{comb}_{M,N}(x, y) = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} \delta(x - kM, y - lN) \]

- Fourier Transform of an impulse train is also an impulse train:

\[ \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} \delta(x - kM, y - lN) \iff \frac{1}{MN} \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} \delta\left(u - \frac{k}{M}, v - \frac{l}{N}\right) \]

\[ \text{comb}_{M,N}(x, y) \quad \text{comb}_{\frac{1}{M}, \frac{1}{N}}(u, v) \]

As the comb samples get further apart, the spectrum samples get closer together!

B.K. Gunturk
Impulse Train

\[ \text{comb}_2[n] \]

Remember:

\[ f(ax) \]

Scaling

\[ \frac{1}{|a|} \mathcal{F}\left( \frac{u}{a} \right) \]

B.K. Gunturk
Sampling low frequency signal

\[ f(x) \]

\[ F(u) \]

\[ \text{comb}_M(x) \]

\[ \text{comb}\frac{1}{M}(u) \]

Multiply:

\[ f(x)\text{comb}_M(x) \]

Convolve:

\[ F(u)\ast\text{comb}\frac{1}{M}(u) \]
Sampling low frequency signal

\[ f(x) \]

\[ F(u) \]

\[ f(x) \text{comb}_M(x) \]

\[ F(u) * \text{comb}_{\frac{1}{M}}(u) \]

No "problem" if \( \frac{1}{M} > 2W \)
Sampling low frequency signal

If there is no overlap, the original signal can be recovered from its samples by low-pass filtering.
Sampling high frequency signal

Overlap: The high frequency energy is folded over into low frequency. It is “aliasing” as lower frequency energy. And you cannot fix it once it has happened.
Sampling high frequency signal

\[ f(x) \leftrightarrow f(x) * h(x) \]

\[ \left[ f(x) * h(x) \right] \text{comb}_M(x) \leftrightarrow \frac{1}{M} \]

Anti-aliasing filter

\[ F(u) \]

B.K. Gunturk
Sampling high frequency signal

- Without anti-aliasing filter:

  \[ f(x)\text{comb}_M(x) \]

- With anti-aliasing filter:

  \[ [f(x) * h(x)]\text{comb}_M(x) \]
Aliasing in Images
Image half-sizing

This image is too big to fit on the screen. How can we reduce it?

How to generate a half-sized version?
**Image sub-sampling**

Throw away every other row and column to create a 1/2 size image - called *image sub-sampling*.
Image sub-sampling

1/2

1/4 (2x zoom)

1/8 (4x zoom)

Aliasing! What do we do?
Gaussian (lowpass) pre-filtering

Solution: filter the image, \textit{then} subsample

- Filter size should double for each $\frac{1}{2}$ size reduction. Why?
Subsampling with Gaussian pre-filtering

Gaussian 1/2  G 1/4  G 1/8
Compare with...

1/2

1/4 (2x zoom)

1/8 (4x zoom)
Campbell-Robson contrast sensitivity curve

The higher the frequency the less sensitive human visual system is...
Lossy Image Compression (JPEG)

Block-based Discrete Cosine Transform (DCT) on 8x8
Using DCT in JPEG

• The first coefficient $B(0,0)$ is the DC component, the average intensity
• The top-left coeffs represent low frequencies, the bottom right – high frequencies
Image compression using DCT

- DCT enables image compression by concentrating most image information in the low frequencies
- Lose unimportant image info (high frequencies) by cutting $B(u,v)$ at bottom right
- The decoder computes the inverse DCT – IDCT

• Quantization Table

<table>
<thead>
<tr>
<th></th>
<th>3</th>
<th>5</th>
<th>7</th>
<th>9</th>
<th>11</th>
<th>13</th>
<th>15</th>
<th>17</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>5</td>
<td>7</td>
<td>9</td>
<td>11</td>
<td>13</td>
<td>15</td>
<td>17</td>
<td>19</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>9</td>
<td>11</td>
<td>13</td>
<td>15</td>
<td>17</td>
<td>19</td>
<td>21</td>
</tr>
<tr>
<td>7</td>
<td>9</td>
<td>11</td>
<td>13</td>
<td>15</td>
<td>17</td>
<td>19</td>
<td>21</td>
<td>23</td>
</tr>
<tr>
<td>9</td>
<td>11</td>
<td>13</td>
<td>15</td>
<td>17</td>
<td>19</td>
<td>21</td>
<td>23</td>
<td>25</td>
</tr>
<tr>
<td>11</td>
<td>13</td>
<td>15</td>
<td>17</td>
<td>19</td>
<td>21</td>
<td>23</td>
<td>25</td>
<td>27</td>
</tr>
<tr>
<td>13</td>
<td>15</td>
<td>17</td>
<td>19</td>
<td>21</td>
<td>23</td>
<td>25</td>
<td>27</td>
<td>29</td>
</tr>
<tr>
<td>15</td>
<td>17</td>
<td>19</td>
<td>21</td>
<td>23</td>
<td>25</td>
<td>27</td>
<td>29</td>
<td>31</td>
</tr>
</tbody>
</table>
JPEG compression comparison

89k

12k
Maybe the end?
Or not!!!

• A teaser on pyramids…
Image Pyramids

Idea: Represent NxN image as a “pyramid” of 1x1, 2x2, 4x4,..., 2^k x 2^k images (assuming N=2^k)

Known as a Gaussian Pyramid [Burt and Adelson, 1983]

- In computer graphics, a *mip map* [Williams, 1983]
- A precursor to *wavelet transform*
Band-pass filtering

Gaussian Pyramid (low-pass images)

These are “bandpass” images (almost).
Why almost bandpass?

- Why are the subtracted sequential Gaussians ("difference of Gaussians") "almost" bandpass?

![Graph showing Gaussians in frequency](image)

- Gaussian: $G(x; \sigma) \iff \frac{\sqrt{2\pi}}{\sigma} G(\omega; \sigma^{-1})$
- Laplacian of Gaussian: $(\frac{x^2}{\sigma^4} - \frac{1}{\sigma^2})G(x; \sigma) \iff -\frac{\sqrt{2\pi}}{\sigma} \omega^2 G(\omega; \sigma^{-1})$
Laplacian Pyramid

Original image

• How can we reconstruct (collapse) this pyramid into the original image?
Computing the Laplacian Pyramid

Don’t worry about these details – YET! (PS4?)

Need $G_k$ to reconstruct
What can you do with band limited imaged?
Apples and Oranges in bandpass

Fine $L_0$

$L_2$

Coarse $L_4$

Reconstructed
What can you do with band limited imaged?
Really the end...
Fourier Transform: Properties

- **Linearity**  \[ af(x, y) + bg(x, y) \iff aF(u, v) + bG(u, v) \]

- **Shifting**  \[ f(x - x_0, y - y_0) \iff e^{-j2\pi(ux_0 + vy_0)} F(u, v) \]

- **Modulation**  \[ e^{j2\pi(u_0x + v_0y)} f(x, y) \iff F(u - u_0, v - v_0) \]

- **Convolution**  \[ f(x, y) \ast g(x, y) \iff F(u, v)G(u, v) \]

- **Multiplication**  \[ f(x, y)g(x, y) \iff F(u, v) \ast G(u, v) \]

- **Separable functions**  \[ f(x, y) = f(x)f(y) \iff F(u, v) = F(u)F(v) \]
Fourier Transform: Properties

- Separability

\[
F(u, v) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y)e^{-j2\pi(ux+vy)} \, dx \, dy
\]

\[
= \int_{-\infty}^{\infty} \left[ \int_{-\infty}^{\infty} f(x, y)e^{-j2\pi ux} \, dx \right] e^{-j2\pi vy} \, dy
\]

\[
= \int_{-\infty}^{\infty} F(u, y)e^{-j2\pi vy} \, dy
\]

2D Fourier Transform can be implemented as a sequence of 1D Fourier Transform operations.
Fourier Transform: Properties

- Energy conservation

\[
\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} |f(x, y)|^2 \, dx \, dy = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} |F(u, v)|^2 \, du \, dv
\]
Fourier Transform: 2D Discrete Signals

- Fourier Transform of a 2D discrete signal is defined as

\[
F(u,v) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} f[m,n]e^{-j2\pi(um+vn)}
\]

where \(-1/2 \leq u, v < 1/2\)

- Inverse Fourier Transform

\[
f[m,n] = \int_{-1/2}^{1/2} \int_{-1/2}^{1/2} F(u,v)e^{j2\pi(um+vn)}dudv
\]
Fourier Transform: Properties

- **Periodicity**: Fourier Transform of a discrete signal is periodic with period 1.

\[
F(u + k, v + l) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} f[m, n] e^{-j2\pi((u+k)m+(v+l)n)}
\]

\[
= \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} f[m, n] e^{-j2\pi(um+vn)} e^{-j2\pi km} e^{-j2\pi ln}
\]

\[
= \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} f[m, n] e^{-j2\pi(um+vn)}
\]

\[
= F(u, v)
\]
Fourier Transform: Properties

- Linearity, shifting, modulation, convolution, multiplication, separability, energy conservation properties also exist for the 2D Fourier Transform of discrete signals.
Fourier Transform: Properties

- **Linearity**  \[ af[m,n] + bg[m,n] \Leftrightarrow aF(u,v) + bG(u,v) \]
- **Shifting**  \[ f[m-m_0, n-n_0] \Leftrightarrow e^{-j2\pi(um_0+vn_0)} F(u,v) \]
- **Modulation**  \[ e^{j2\pi(um_0+vn_0)} f[m,n] \Leftrightarrow F(u-u_0, v-v_0) \]
- **Convolution**  \[ f[m,n] \ast g[m,n] \Leftrightarrow F(u,v)G(u,v) \]
- **Multiplication**  \[ f[m,n]g[m,n] \Leftrightarrow F(u,v) \ast G(u,v) \]
- **Separable functions**  \[ f[m,n] = f[m]f[n] \Leftrightarrow F(u,v) = F(u)F(v) \]
- **Energy conservation**  \[ \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} \left| f[m,n] \right|^2 = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \left| F(u,v) \right|^2 \, du \, dv \]
Fourier Transform: Properties

- Define *Kronecker delta function*

\[
\delta[m, n] = \begin{cases} 
1, & \text{for } m = 0 \text{ and } n = 0 \\
0, & \text{otherwise}
\end{cases}
\]

- Fourier Transform of the Kronecker delta function

\[
F(u, v) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} \left[ \delta[m, n]e^{-j2\pi(um+vn)} \right] = e^{-j2\pi(u0+v0)} = 1
\]
Fourier Transform: Properties

- Fourier Transform of 1

\[
f(m, n) = 1 \iff F(u, v) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} \left[ 1 - j 2\pi(um + vn) \right] = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} \delta(u - k, v - l)
\]

**To prove:** Take the inverse Fourier Transform of the Dirac delta function and use the fact that the Fourier Transform has to be periodic with period 1.
Sampling Theorem

Continuous signal:

\[ f(x) \]

Shah function (Impulse train):

\[ s(x) = \sum_{n=-\infty}^{\infty} \delta(x - nx_0) \]

Sampled function:

\[ f_s(x) = f(x)s(x) = f(x) \sum_{n=-\infty}^{\infty} \delta(x - nx_0) \]
Sampling Theorem

Sampled function:

\[ f_s(x) = f(x)s(x) = f(x)\sum_{n=-\infty}^{\infty} \delta(x-nx_0) \]

\[ F_S(u) = F(u) * S(u) = F(u) * \frac{1}{x_0} \sum_{n=-\infty}^{\infty} \delta\left( u - \frac{n}{x_0} \right) \]

Only if \( u_{\text{max}} \leq \frac{1}{2x_0} \)
Nyquist Theorem

If \( u_{\text{max}} > \frac{1}{2x_0} \)

When can we recover \( F(u) \) from \( F_S(u) \) ?

Only if \( u_{\text{max}} \leq \frac{1}{2x_0} \) (Nyquist Frequency)

We can use

\[
C(u) = \begin{cases} 
  x_0 & |u| < \frac{1}{2x_0} \\
  0 & \text{otherwise}
\end{cases}
\]

Then \( F(u) = F_S(u)C(u) \) and \( f(x) = \text{IFT}[F(u)] \)

Sampling frequency must be greater than \( 2u_{\text{max}} \)