Programming Assignment 3

CS 474

https://github.com/alexander-novo/CS474-PA3

Alexander Novotny
50% Work

Matthew Page
50% Work

Due: November 16, 2020
Submitted: November 13, 2020

Contents

1 Experiment 1

1.1 Theory

1.2 Implementation

1.3 Results and Discussion

2 Experiment 2

2.1 Theory

The 2-Dimensional Fourier Transform has a property called "separability", which shows that it can be computed as nested 1-Dimensional Fourier Transforms. A proof of this fact is given below:

$$\begin{aligned}
\mathcal{F}\{f(x,y)\}(u,v) &= \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)\exp\Big(-i2\pi\Big(\frac{ux}{M}+\frac{vy}{N}\Big)\Big) \\
&= \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)\exp\Big(-i2\pi\frac{ux}{M}\Big)\exp\Big(-i2\pi\frac{vy}{N}\Big) \\
&= \sum_{x=0}^{M-1}\left[\exp\Big(-i2\pi\frac{ux}{M}\Big)\sum_{y=0}^{N-1} f(x,y)\exp\Big(-i2\pi\frac{vy}{N}\Big)\right] \\
&= \sum_{x=0}^{M-1}\exp\Big(-i2\pi\frac{ux}{M}\Big)\mathcal{F}_y\{f(x,y)\}(x,v) \\
&= \mathcal{F}_x\{\mathcal{F}_y\{f(x,y)\}(x,v)\}(u,v),
\end{aligned} \tag{1}$$

where $\mathcal{F}_x$ is the 1-Dimensional Fourier Transform of the x variable.

Another property of the 2-D Fourier Transform is the translation property:

$$\mathcal{F}\Big\{f(x,y)\exp\Big(i2\pi\Big(\frac{u_0x}{M}+\frac{v_0y}{N}\Big)\Big)\Big\}(u,v) = \mathcal{F}\{f(x,y)\}(u-u_0,v-v_0), \tag{2}$$

from which we can see that the Fourier Transform translated half a period in each dimension is

$$\begin{aligned}
\mathcal{F}\{f(x,y)\}(u-M/2,v-N/2) &= \mathcal{F}\left\{f(x,y)\exp\left(i2\pi\left(\frac{(M/2)x}{M}+\frac{(N/2)y}{N}\right)\right)\right\}(u,v) \\
&= \mathcal{F}\{f(x,y)\exp(i\pi(x+y))\}(u,v) \\
&= \mathcal{F}\Big\{f(x,y)\exp(i\pi)^{(x+y)}\Big\}(u,v) \\
&= \mathcal{F}\Big\{f(x,y)(-1)^{(x+y)}\Big\}(u,v).
\end{aligned} \tag{3}$$

2.2 Implementation

Equation (1) (the separability property) shows that we can compute the 2-Dimensional Fourier Transform of an image by simply reusing our 1-Dimensional Fast Fourier Transform algorithm and applying it first to the columns of an image, and then to the rows. A nice benefit to this is that the Fourier

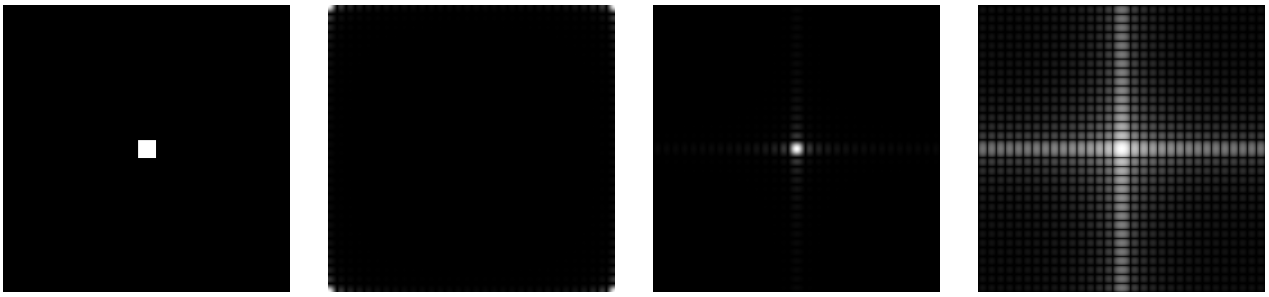
transform of each row (and each column) is independent from each other, so it is trivially parallelizable in shared memory (simply divide the rows evenly among threads and have them calculate each row separately, then do the same for columns - you must have a barrier between calculating columns and rows and it is recommended to use shared memory, as every row calculation depends on every column calculation).

Since the dc term $\mathcal{F}(f(x,y))(0,0)$ and surrounding values dominate the spectrum, they are the most visually interesting values. Unfortunately, in an image, $(0,0)$ is in the corner, so it is difficult to see, and surrounding values are spread across every corner due to the Fourier Transform being periodic in the dimensions of the image (see fig. 1b). It makes sense to shift these more important values to the center of the image, so we use eq. (3) to translate the Fourier Transform by one half period in each direction (half the image) by simply multiplying half the pixels by -1 in a checkerboard pattern before taking the Fourier Transform (see fig. 1c).

As well, the dc term and surrounding values dominates the spectrum by several orders of magnitude, typically. This means that when remapping to $[0, 255]$ for an image, most of the spectrum will be mapped to 0 (or nearby), making it invisible. We can get rid of these orders of magnitude difference by simply applying a logarithmic intensity transformation before converting to an image (see fig. 1d).

2.3 Results and Discussion

Figure 1 shows the steps detailed above



(a) A 512×512 pixel image with a centered 32×32 pixel rectangle.

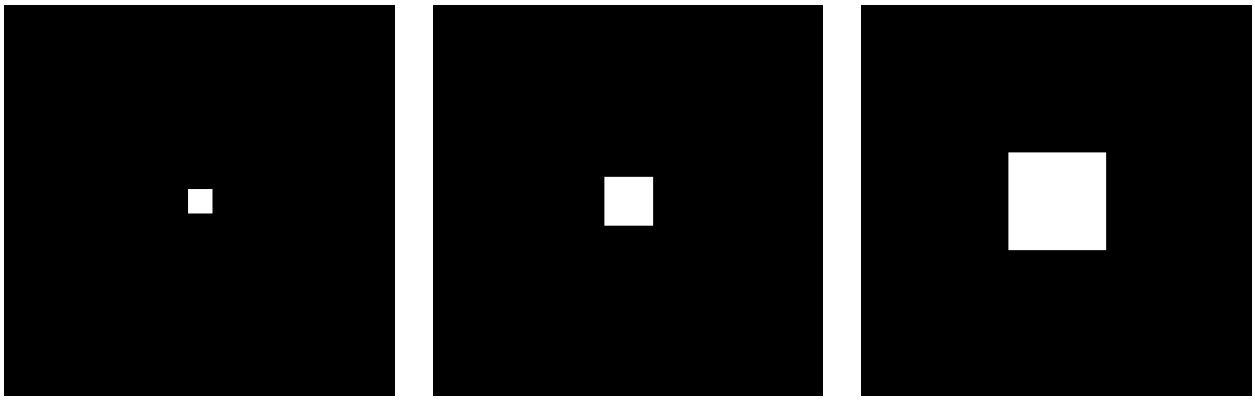
(b) The spectrum of the image.

(c) The spectrum shifted so that the dc term is centered on the image.

(d) A logarithmic intensity transformation is applied for better visualization.

Figure 1: A rectangle and its spectrum.

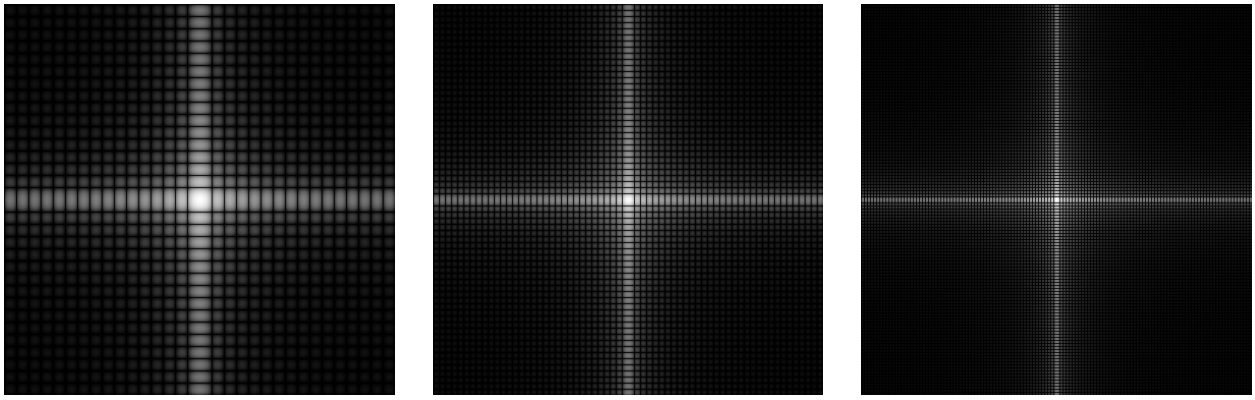
Figure 2 compares the spectrums of rectangles of different sizes after applying the above transformations. We can see that each one is the product of sinc functions (the expected outcome, as we went over in class) and as the rectangles get larger, the the sinc sin functions become higher frequency. This, combined with the symmetry property of the Fourier Transform, indicates that as the frequency of a sinc function gets higher, its Fourier Transform takes on values further from 0, which makes sense.



(a) A 512×512 pixel image with a centered 32×32 pixel rectangle.

(b) A 512×512 pixel image with a centered 64×64 pixel rectangle.

(c) A 512×512 pixel image with a centered 128×128 pixel rectangle.



(d) The spectrum of above.

(e) The spectrum of above.

(f) The spectrum of above.

Figure 2: A comparison of spectrums of rectangles of different sizes.

3 Experiment 3

3.1 Theory

3.2 Implementation

3.3 Results and Discussion