Good afternoon. My name is Yao Xueyang. Thank you for coming today.Today I’m gonna present part of my master work on topic ‘Discrete 2D-Fourier transform in polar coordinates’. The outline is as shown: first, introduction on Fourier transform. Then the definition of the discrete 2D-Fourier transform in polar coordinates, followed by test of accuracy of the transform and finally the conclusion.

Ok, let’s start with the introduction of the Fourier transform. Fourier transform has been proved to be invaluable in many disciplines. Here is its definition. The forward transform will convert the function from time domain to frequency domain. The inverse transform does the other way.

Here’s a little example. On the left hand side is a signal in time domain and the right hand side is its Fourier transform. This signal sounds like this. From the Fourier transform we can see there are three spikes. These three spikes denote that the original signal is actually a combination of three sinusoidal signals with different frequencies. As you can see here. So, the Fourier transform can help find the frequency content of a signal. And we can use sin signals with different frequencies as building block to construct a signal.

Then we come to 2D-Fourier transform in Cartesian coordinates. As you can see, the 2D Fourier transform consists 2 integrals. It is widely used in image processing. Here’s an example. On the left, as you can see, there are some strips with dark and light. For example this one, if we define the intensity of light as the amplitude, say light as 1 and dark as -1, it is actually a sin signal in this direction and constant in this direction. The 2D-Fourier Transform of this image is these two points. These two points denote that the sin signal is along this direction and distance from the point to the origin denotes the frequency. Similarly, these two points are the Fourier transform of this image. It is sin signal in this direction and since the frequency is apparently bigger than that one, the point gets farther to origin. These strips are the building blocks of an image.

In previous example, we define the intensity of light as the amplitude of an image. But if we define certain values as the amplitude, we will get plots like this. The left hand side is the original function and the right hand side the 2D-Fourier Transform. And they are in Cartesian coordinates. But as you can see, the function is angular symmetric, which means if we define the function in polar coordinates, it will be just a function in radial direction because the function in angular direction is constant. Therefore, for functions that are best described in polar coordinates, a Fourier transform in polar coordinates is needed.

Here comes our definition of discrete 2D-Fourier Transform in polar coordinates. I know it looks a little messy. I won’t go too deep in the math. Basically, it transforms a matrix fpk to another matrix Fql. And you can do it backwards. It doesn’t sound very interesting. But, it gets a little interesting if this discrete transform can be used to approximate the continuous Fourier transform. So, can we use it to approximate the continuous Fourier transform? Here is the definition of the discrete transform to approximate the continuous transform. Different from previous definition, we have scaling factors and specific sample points. I know they look a little weird, I will talk about it later. And this big function is little heavy, I will make easier. Basically, we have function little f and its continuous FT, we sample them at some specific points, name it fpk. Than we perform the discrete transform and get Fql. The question is does this Fql equals to the sampled F.

To answer this question, we need to test the accuracy of the discrete transform. And we use dynamic error to measure the accuracy. Here’s the definition of dynamic error. Basically, the dynamic error just shows the difference between the continuous 2D-Fourier transform and the discrete counterpart part. Since the unit is decibel , so bigger negative value means smaller error.

Here’s the first test function, Gaussian. The original function and it continuous FT are as shown. Here is the test result. On the left, it is the sampled continuous Forward transform and the discrete transform. On the right, it is the error distribution. It could be observed that the error gets bigger at the center. The maximum error is -8dB and the average of the error is -64 dB.

Here’s the test result of the inverse transform. The maximum error is -12dB and the average of the error is -98 dB. Similarly, the error gets bigger at the center.

The second test function is square donut. It is called this way because it looks like a donut but have square edges. The plots show the original function and its Fourier Transform. Here’s the test result of the forward transform. The maximum error is --8dB and the average of the error is -34 dB. This result is not as good as the Gaussian. It is as expected because of discontinuity. Also, we can see at the error gets bigger at the center. Seems like we have trouble with the center area. This is the inverse transform, the error max is 1.5 dB and the error average is -73db.

The third test function is Four-term sinusoid and Modified exponential. The original function and FT transform are as shown. Here’s the test result of the Forward transform, error max is -6.6db and error average is -29db. Here’s the test result of the Forward transform, error max is 0.6db and error average is -69db The result gets even worse. Still, bigger error at the origin.

Here’s a summary of the test results. From the table, the Gaussian function shows the best result because Gaussian function is smooth and continuous. The worst case is the Fou-term sin and Mexp. From the tests, we can see that the discrete transform shows good approximation because worst case still has an Error average of -30 which is acceptable. We can see that for all the cases the dynamic error gets bigger at the center.

Another interesting fact is about how the sample size will effect on the Dynamic error. Intuitively the more you sample, the more information captured , the less error you should get. However, in this picture, N1 is the sample size in radial direction and N2 is the sample size in angular direction, it could observed that increasing N1 will decrease the error but increasing N2 will increase the error.

It is found that this bizarre phenomena is caused by the sample points. The left is sample grid in Cartesian coordinates and the right is sample grid in polar coordinates. The Cartesian sample grid is equally spaced and starts from the origin, While the polar grid is not equally space and cannot get to the origin. This explains why we have bigger error at the center, because we cannot sample there. Further work shows increasing N1 will make the sample points closer to the origin but increasing N2 will make this gap bigger.

So we get the guidelines of choosing sample size.

After verify that the discrete transform can be used to approximate the continuous transform, the next problem is that the algorithm is pretty complicated so it costs a lot of time to do the transform. So in order to make it useful in real life, we try to make some improvements on reducing the computing time.

First, we go back to the definition. Instead of thinking of the transform with a big kernel, we get a new interpretation of the transform as a DFT, DHT and IDFT in sequence. This is very interesting, because this way, we can use the fast Fourier transform (fft) algorithm and save a lot of time.

Another way to reduce computing time is by pre-calculating the Bessel zeros. Because it was found that a lot of computational time was used to calculate the Bessel zeros for every different test case, even though the Bessel zeros are the same in every case. So pre-calculating the Bessel zeros and save it in a file could save a lot of computational time.

This table shows the computing time for three cases, it could be seen that, by interpreting the transform as 3 operations and pre-calculating the Bessel zeros, the computational time is reduced significantly.