**高等影像處理**

**作業#(5)**

姓名：梁仔弘

|  |  |
| --- | --- |
| 1.1 | |
| **Figures** | |
| 1\_1\_dftRect.raw | 1\_1\_dftRectRot.raw |
| 1\_1\_dftCircle.raw | 1\_1\_dftLines.raw |
| **Discussion** | |
| 除了Line那張圖，其他圖片在計算傅立葉轉換的時候都可以用一個高點是255，低點是0的方波來看待。而方波在經過傅立葉轉換以後便會變成Sinc波，這就造成了每一張頻譜圖都會有一些類似漣漪的現象。 而對於Line的傅立葉頻譜，我們可以先看最明顯穿過中心的白線，因為這些穿過中心的白線，理應根據原圖的白線數量與方向做出相對應的轉換。而我們的原圖中，四條白線都朝向不同方向，因此轉換過後的頻譜也應該有四條朝向不同方向的白線。 如果今天原圖是四條白線，但兩兩方向相同，如正方形，那轉換過後的頻譜也只會有兩條白線。 | |

|  |  |  |
| --- | --- | --- |
| 1.2 | | |
| **Figures** | | |
| Self-made DFT | OpenCV DFT | |
|  |  | 1\_2\_dftRect.raw |
|  |  | 1\_2\_dftRectRot.raw |
|  |  | 1\_2\_dftCircle.raw |
|  |  | 1\_2\_dftLines.raw |
| Discussion | | |
| 就結果圖來看，我認為除了顏色對比度以外，外觀基本上是一模一樣的，而細微的對比度也幾乎可以忽略不計。  最有感覺的就是執行時間了，我自己的DFT大概要跑3分鐘，而OpenCV的DFT按下去就完成了，這速度差距真的很大。不過這應該是因為OpenCV的算法是FFT+蝴蝶排序，所以才能這麼快吧。而我自己的版本則是要執行四個for迴圈，時間複雜度是，雖然做兩次一維DFT可以把時間複雜度降到，但還是遠比OpenCV慢。 | | |

|  |  |
| --- | --- |
| 1.3 | |
| **Figures** | |
| 1\_3\_dftRect.raw | 1\_3\_dftRectRot.raw |
| 1\_3\_dftCircle.raw | 1\_3\_dftLines.raw |
|  | |
| Discussion | |
| 我的IDFT跑出來的結果有點奇怪，除了Line那一張圖以外，其他的圖都或多或少有些變形，比如說第一張，下面有一些雜訊，第二張則是更明顯，有奇怪的菱形邊框，第三張的圓則是感覺沒有原圖那麼的圓。其實我不確定為什麼會這樣，因為我認為我的程式並沒有問題，但我認為最大的可能應該是因為取樣頻率不夠吧。不過MSE真的是一個不錯的指標，錯得越誇張的圖MSE明顯就越大。 | |

|  |  |  |
| --- | --- | --- |
| 1.4 | | |
| **Figures** | | |
| Self-made IDFT | OpenCV IDFT | |
|  |  | 1\_4\_dftRect.raw |
|  |  | 1\_4\_dftRectRot.raw |
|  |  | 1\_4\_dftCircle.raw |
|  |  | 1\_4\_dftLines.raw |
| Discussion | | |
| 執行速度依然是OpenCV屌打我，而且出來的圖片還比我的更漂亮，MSE應該也不用算了，肯定比我的好。除了Line以外，我跟OpenCV輸出的基本上都不太一樣，因為OpenCV是原圖，沒有任何誤差。 | | |

|  |  |  |
| --- | --- | --- |
| 2.1 | | |
| **Figures** | | |
| Ideal Low-Pass Filter | | |
|  | Filtered Spectrum | Result of Inverse |
| d=5 | 2\_1\_cat\_LP\_spec\_d5.raw | 2\_1\_cat\_LP\_d5.raw |
| d=20 | 2\_1\_cat\_LP\_spec\_d20.raw | 2\_1\_cat\_LP\_d20.raw |
| d=50 | 2\_1\_cat\_LP\_spec\_d50.raw | 2\_1\_cat\_LP\_d50.raw |
| Ideal High-Pass Filter | | |
| d=5 | 2\_1\_cat\_HP\_spec\_d5.raw | 2\_1\_cat\_HP\_d5.raw |
| d=20 | 2\_1\_cat\_HP\_spec\_d20.raw | 2\_1\_cat\_HP\_d20.raw |
| d=50 | 2\_1\_cat\_HP\_d50.raw | 2\_1\_cat\_HP\_d50.raw |
| **Discussion** | | |
| 根據頻譜分析的基本原理，頻譜中心部位代表著低頻，低頻代表著輪廓等較高階的表徵；頻譜越外圍代表高頻訊號，而高頻代表著細節、材質等。  **理想低通濾波器：**此濾波器將高頻完全截掉，只留下低頻，那就只剩下輪廓等高階特徵被保留，因此可以看到，隨著D0越大，能看到的細節也就越多。  **理想高通濾波器：**此濾波器將低頻完全截掉，只留下高頻，那就只剩下細節等一眼看過去的特徵被保留，因此可以看到，隨著D0越大，原本的輪廓就更加模糊。到D0=50的時候，就只剩下了一些毛毛躁躁的小細節。 | | |

|  |  |  |
| --- | --- | --- |
| 2.2 | | |
| **Figures** | | |
| Gaussian Low-Pass Filter | | |
|  | Filtered Spectrum | Result of Inverse |
| d=5 | 2\_2\_cat\_GLP\_spec\_d5.raw | 2\_2\_cat\_GLP\_d5.raw |
| d=20 | 2\_2\_cat\_GLP\_spec\_d20.raw | 2\_2\_cat\_GLP\_d20.raw |
| d=50 | 2\_2\_cat\_GLP\_spec\_d50.raw | 2\_2\_cat\_GLP\_d50.raw |
| Gaussian High-Pass Filter | | |
| d=5 | 2\_2\_cat\_GHP\_spec\_d5.raw | 2\_2\_cat\_GHP\_spec\_d5.raw |
| d=20 | 2\_2\_cat\_GHP\_spec\_d20.raw | 2\_2\_cat\_GHP\_d20.raw |
| d=50 | 2\_2\_cat\_GHP\_spec\_d50.raw | 2\_2\_cat\_GHP\_d50.raw |
| **Discussion** | | |
| 高低通濾波器的原理在2.1已經提過，在此不再贅述。這邊主要討論漣漪現象。  根據2.1理想濾波器的結果，雖然我們有達到我們要的效果，不過轉換回來的圖中，卻出現了一些俗稱「水波紋」的現象。這是因為理想濾波器的遮罩的邊緣是從0直接跳到255(或反過來)，這樣劇烈的變動對於傅立葉轉換來說相當於方波，而方波就會造成Sinc波出現，這就導致了漣漪現象的產生。  而高斯濾波器就是為了避免這樣的情況而被使用，因為其遮罩是非常平滑的，所以可以看到本題的結果很好地消除了不必要的漣漪，當高通濾波器的D0調大時，也能清楚地顯示貓貓的細節。 | | |

|  |  |  |
| --- | --- | --- |
| 3.1 | | |
| **Figures** | | |
| k=15 | 3\_1\_watermarked\_pancake\_spec\_normal\_k.raw | 3\_1\_watermarked\_pancake\_normal\_k.raw |
| k=200 | 3\_1\_watermarked\_pancake\_spec\_large\_k.raw | 3\_1\_watermarked\_pancake\_large\_k.raw |
| **Discussion** | | |
| 隨著k越大，浮水印的字樣也會越清楚。而我們並不需要讓他看得太清楚，所以只要能夠剛好看到就好，因此我的k選擇k=15。而k值越大，所轉換回來的圖片也會越暗，因此必須選擇最小的k，同時保證浮水印不會太不明顯。 | | |

|  |  |
| --- | --- |
| 3.2 | |
| **Figures** | |
| 3\_2\_blurred\_watermarked\_pancake.raw | 3\_2\_blurred\_watermarked\_pancake\_spectrum.raw |
| **Discussion** | |
| 我的推論：因為對原圖做高斯模糊會造成細節的部份丟失，而細節的部份，在對應的傅立葉頻譜則是高頻的地區，而這就造成了我們打上的浮水印，跟著高斯模糊一起被糊掉了。 | |