1. DFT+음악파일 필터링
2. #include<iostream>
3. #include<fstream>
4. #include"complex.h"
5. using namespace std;
6. #define PI 3.141592
7. #define WORD unsigned short
8. #define DWORD unsigned int
9. int main()
10. {
11. ifstream Infile;//읽어올 것
12. Infile.open("MixA.wav", ios::binary);
13. char\* header = new char[44];
14. Infile.read((char\*)header, 44);//문법적인 사항으로 기억하자
15. cout << "RIFF " << header[0] << header[1] << header[2] << header[3] << endl;
16. cout << "filesize " << \*(DWORD\*)(header + 4) << endl;//wav 파일의 특성, 실제는 +8해줘야 함
17. cout << "WAVE " << header[8] << header[9] << header[10] << header[11] << endl;
18. cout << "cksize " << \*(DWORD\*)(header + 16) << endl;
19. cout << "channels " << \*(WORD\*)(header + 22) << endl;//중간에 불필요한 부분 삭제돼있음(1이면 모노 2면 스테레오)
20. cout << "fs " << \*(DWORD\*)(header + 24) << endl;//샘플링 주파수 8000
21. cout << "bytes/s " << \*(DWORD\*)(header + 28) << endl;
22. cout << "bits/sam " << \*(WORD\*)(header + 34) << endl;//샘플당 바이트, 16비트=2바이트->short형 선언이 유리
23. cout << "data " << header[36] << header[37] << header[38] << header[39] << endl;
24. cout << "cksize " << \*(DWORD\*)(header + 40) << endl; //sub-chunk 2size
26. short\* data = new short[16000]; //샘플의 수만큼 크기를 할당하고 하나당 2바이트이므로 short형으로 선언
27. Infile.read((char\*)data, 16000\*sizeof(short)); //char\*는 문법적인 내용 반드시 저렇게 써야함, 콤마뒤에는 전체용량
28. //음악파일의 모든 정보 "data"에 저장.
29. ofstream OFk, OFm, OFph;
30. OFk.open("k.txt");
31. OFm.open("mag.txt");
32. OFph.open("phase.txt");
33. complex\* x = new complex[16000];
34. for (int n=0;n<16000;n++)
35. x[n] = complex(data[n], 0);
36. complex\* X = new complex[16000];
37. for (int k = 0; k < 16000; k++)
38. {
39. for (int n = 0; n < 16000; n++)
40. {
41. X[k] += complex(data[n],0)\*complex(cos((-2. \* PI\*k\*n) / (double)16000), sin((-2. \* PI\*k\*n) / (double)16000));
42. }
43. }
44. for (int k = 0; k < 16000; k++)
45. {
46. OFk << (double)8000/16000\*k << endl;//인덱스에서 주파수로 변경
47. OFm << X[k].mag() << endl;
48. OFph << X[k].phase() << endl;
49. }
50. OFk.close();
51. OFm.close();
52. OFph.close();
53. for (int k = 2000; k < 16000; k++)//필터를 통해 특정 주파수의 크기를 0으로 만듬
54. {
55. X[k] = complex(0, 0);
56. }
57. for (int n = 0; n < 16000; n++)
58. {
59. for (int k = 0; k < 16000; k++)
60. {
61. x[n] += X[k] \* complex(cos((2. \* PI\*k\*n) / (double)16000), sin((2. \* PI\*k\*n) / (double)16000));
62. }
63. x[n] = x[n] / 16000;
64. }
65. short\*data\_ = new short[16000];
66. for (int i = 0; i < 16000; i++)
67. {
68. data\_[i] = x[i].re;
69. }
70. ofstream TEST("NEWA.wav", ios::binary);
71. TEST.write((char\*)header, 44);
72. TEST.write((char\*)data\_, 32000);
73. return 0;
74. }
75. 2차원 DFT+이미지파일 필터링
76. #include<iostream>
77. #include<fstream>
78. #include"complex.h"
79. using namespace std;
80. #define PI 3.141592
81. #define W 64
82. #define H 64
83. #define Size 12342
84. #define HSize 54
85. int main()
86. {
87. ifstream InF;
88. InF.open("twin\_noise\_64.bmp", ios::binary);
89. char\*header = new char[HSize];
90. InF.read((char\*)header, 54);
92. unsigned char\*\* A;
93. A = new unsigned char\*[64];
94. for (int i = 0; i < 64; i++)
95. {
96. A[i] = new unsigned char[64];
97. }
98. unsigned char\*\*B;
99. unsigned char\*\*G;
100. unsigned char\*\*R;
101. unsigned char\*\*BGR;
102. complex \*\*dft;
103. B = new unsigned char\*[64];
104. G = new unsigned char\*[64];
105. R = new unsigned char\*[64];
106. BGR = new unsigned char\*[3 \* 64];
107. dft = new complex\*[64];
108. for (int i = 0; i < 64; i++)
109. {
110. B[i] = new unsigned char[64];
111. G[i] = new unsigned char[64];
112. R[i] = new unsigned char[64];
113. BGR[i] = new unsigned char[3 \* 64];
114. dft[i] = new complex[64];
115. }//dft값 초기화
116. for (int i = 0; i < 64; i++)
117. {
118. for (int j = 0; j < 64; j++)
119. {
120. dft[i][j] = 0;
121. }
122. }
123. for (int i = 0; i < 64; i++)
124. {
125. InF.read((char\*)BGR[i], 3 \* 64 \* sizeof(char));
126. }
127. for (int i = 0; i < 64; i++)
128. {
129. for (int j = 0, jj=0; j < 64; j++,jj+=3)
130. {
131. B[i][j] = BGR[i][jj];
132. G[i][j] = BGR[i][jj+1];
133. R[i][j] = BGR[i][jj+2];
134. }
135. }
136. for (int v = 0; v < 64; v++) //DFT
137. {
138. for (int u = 0; u < 64; u++)
139. {
140. for (int y = 0; y < 64; y++)
141. {
142. for (int x = 0; x < 64; x++)
143. {
144. dft[v][u] += complex(R[y][x], 0)\*complex(-2.\*PI\*(((double)u\*x / W) + ((double)v\*y / H)));
145. }
146. }
147. }
148. }
149. //노이즈값 제거
150. for (int i = 10; i < 30; i++)
151. {
152. for (int j = 0; j < 64 ; j++)
153. {
154. if (j > 10 && j < 30)
155. dft[i][j] = complex(0, 0);
156. }
157. }
158. for (int i = 30; i< 50; i++)
159. {
160. for (int j = 0; j < 64; j++)
161. {
162. if (j > 30 && j < 50)
163. dft[i][j] = complex(0, 0);
164. }
165. }
166. ofstream please;//텍스트로 결과출력
167. please.open("pease.txt");
168. for (int i = 0; i < 64; i++)
169. {
170. for(int j = 0; j < 64;j++)
171. {
172. please << dft[i][j].mag() <<"\t";
173. }
174. please << endl;
175. }
176. complex\*\* idft;
177. idft = new complex\*[64];
178. for (int i = 0; i < 64; i++)
179. {
180. idft[i] = new complex[64];
181. }
182. for (int v = 0; v < 64; v++)//IDFT
183. {
184. for (int u = 0; u < 64; u++)
185. {
186. for (int y = 0; y < 64; y++)
187. {
188. for (int x = 0; x < 64; x++)
189. {
190. idft[v][u] += dft[y][x] \* complex(2.\*PI\*(((double)u\*x / W) + ((double)v\*y / H)));
191. }
192. }
193. idft[v][u] = idft[v][u] / (64 \* 64);
194. R[v][u] = idft[v][u].re;
195. }
196. }
197. for (int i = 0; i < W; i++)
198. {
199. for (int j = 0, jj = 0; j < H; j++, jj += 3)
200. {
201. BGR[i][jj] = R[i][j];
202. BGR[i][jj + 1] = R[i][j];
203. BGR[i][jj + 2] = R[i][j];
204. }
205. }
206. ofstream TEST("endgame.bmp", ios::binary);
207. TEST.write((char\*)header, 54);
208. for (int i = 0; i < H; i++) {
209. TEST.write((char\*)BGR[i], 3 \* W);
210. }
211. return 0;
212. }
213. 2차원 DCT+이미지파일 필터링
214. #include<iostream>
215. #include<fstream>
216. #define PI 3.141592
217. using namespace std;
218. void DCT\_2D(unsigned char\*\*data, double\*\*dct);
219. void IDCT\_2D(double\*\*dct, unsigned char\*\*result\_data);
220. int main()
221. {
222. ifstream InF;//파일 열음
223. InF.open("twin\_.bmp", ios::binary);
224. char\*header = new char[54];
225. InF.read((char\*)header, 54);
226. unsigned char\*\* A, \*\*AA, \*\*AAA;
227. A = new unsigned char\*[256];
228. AA = new unsigned char\*[256];
229. AAA = new unsigned char\*[256];
230. for (int i = 0; i < 256; i++)
231. {
232. A[i] = new unsigned char[256];
233. AA[i] = new unsigned char[256];
234. AAA[i] = new unsigned char[256];
235. }
236. unsigned char\*\*B;
237. unsigned char\*\*G;
238. unsigned char\*\*R;
239. unsigned char\*\*BGR;
240. double \*\*dct1, \*\*dct2, \*\*dct3;
241. B = new unsigned char\*[256];
242. G = new unsigned char\*[256];
243. R = new unsigned char\*[256];
244. BGR = new unsigned char\*[3 \* 256];
245. dct1 = new double\*[256];
246. dct2 = new double\*[256];
247. dct3 = new double\*[256];
248. for (int i = 0; i < 256; i++)
249. {
250. B[i] = new unsigned char[256];
251. G[i] = new unsigned char[256];
252. R[i] = new unsigned char[256];
253. BGR[i] = new unsigned char[3 \* 256];
254. dct1[i] = new double[256];
255. dct2[i] = new double[256];
256. dct3[i] = new double[256];
257. }//여기까지는 인덱스, 칸 세팅
258. for (int i = 0; i < 256; i++)
259. {
260. InF.read((char\*)BGR[i], 3 \* 256 \* sizeof(char));
261. }
262. for (int i = 0; i < 256; i++)
263. {
264. for (int j = 0, jj = 0; j < 256; j++, jj += 3)
265. {
266. B[i][j] = BGR[i][jj];
267. G[i][j] = BGR[i][jj + 1];
268. R[i][j] = BGR[i][jj + 2];
269. }
270. }
271. DCT\_2D(R, dct1);
272. DCT\_2D(R, dct2);
273. DCT\_2D(R, dct3);
274. //dct 필터링 시작
276. //dct1번
277. int N = 8;
278. int mcrNb = 32;
279. int u, v;
280. for (int mcr\_i = 0; mcr\_i < mcrNb; mcr\_i++)
281. {
282. for (int mcr\_j = 0; mcr\_j < mcrNb; mcr\_j++)
283. {
284. for (int k = 0; k < N; k++)
285. {
286. u = mcr\_i \* N + k;
287. for (int l = 0; l < N; l++)
288. {
289. v = mcr\_j \* N + l;
290. if (k > 4 || l > 4)
291. {
292. dct1[u][v] = 0;
293. }
294. }
295. }
296. }
297. }
298. //2번
299. for (int mcr\_i = 0; mcr\_i < mcrNb; mcr\_i++)
300. {
301. for (int mcr\_j = 0; mcr\_j < mcrNb; mcr\_j++)
302. {
303. for (int k = 0; k < N; k++)
304. {
305. u = mcr\_i \* N + k;
306. for (int l = 0; l < N; l++)
307. {
308. v = mcr\_j \* N + l;
309. if (k > 1 && l > 1)
310. {
311. dct2[u][v] = 0;
312. }
313. }
314. }
315. }
316. }
317. //3번
318. for (int mcr\_i = 0; mcr\_i < mcrNb; mcr\_i++)
319. {
320. for (int mcr\_j = 0; mcr\_j < mcrNb; mcr\_j++)
321. {
322. for (int k = 0; k < N; k++)
323. {
324. u = mcr\_i \* N + k;
325. for (int l = 0; l < N; l++)
326. {
327. v = mcr\_j \* N + l;
328. if (k < 3 && l < 3)
329. {
330. dct3[u][v] = 0;
331. }
332. }
333. }
334. }
335. }
337. IDCT\_2D(dct1, A);
338. IDCT\_2D(dct2, AA);
339. IDCT\_2D(dct3, AAA);
340. ofstream first("fst.bmp", ios::binary);//1번출력
341. first.write((char\*)header, 54);
342. for (int i = 0; i < 256; i++)
343. {
344. for (int j = 0, jj = 0; j < 256; j++, jj += 3)
345. {
346. BGR[i][jj] = A[i][j];
347. BGR[i][jj + 1] = A[i][j];
348. BGR[i][jj + 2] = A[i][j];
349. }
350. }
351. for (int i = 0; i < 256; i++)
352. {
353. first.write((char\*)BGR[i], 3 \* 256 \* sizeof(char));
354. }
355. ofstream second("scd.bmp", ios::binary);//2번출력
356. second.write((char\*)header, 54);
357. for (int i = 0; i < 256; i++)
358. {
359. for (int j = 0, jj = 0; j < 256; j++, jj += 3)
360. {
361. BGR[i][jj] = AA[i][j];
362. BGR[i][jj + 1] = AA[i][j];
363. BGR[i][jj + 2] = AA[i][j];
364. }
365. }
366. for (int i = 0; i < 256; i++)
367. {
368. second.write((char\*)BGR[i], 3 \* 256 \* sizeof(char));
369. }
370. ofstream third("thd.bmp", ios::binary);//3번출력
371. third.write((char\*)header, 54);
372. for (int i = 0; i < 256; i++)
373. {
374. for (int j = 0, jj = 0; j < 256; j++, jj += 3)
375. {
376. BGR[i][jj] = AAA[i][j];
377. BGR[i][jj + 1] = AAA[i][j];
378. BGR[i][jj + 2] = AAA[i][j];
379. }
380. }
381. for (int i = 0; i < 256; i++)
382. {
383. third.write((char\*)BGR[i], 3 \* 256 \* sizeof(char));
384. }
385. system("pause");
386. return 0;
387. }
388. void DCT\_2D(unsigned char\*\*data, double\*\*dct)
389. {
390. int N = 8;
391. int mcrNb = 32;
392. double sum = 0;
393. int u, v, y, x;
394. for (int mcr\_i = 0; mcr\_i < mcrNb; mcr\_i++)
395. {
396. for (int mcr\_j = 0; mcr\_j < mcrNb; mcr\_j++)
397. {
398. for (int k = 0; k < N; k++)
399. {
400. u = mcr\_i \* N + k;
401. for (int l = 0; l < N; l++)
402. {
403. v = mcr\_j \* N + l;
404. sum = 0;
405. for (int i = 0; i < N; i++)
406. {
407. y = N \* mcr\_i + i;
408. for (int j = 0; j < N; j++)
409. {
410. x = mcr\_j \* N + j;
411. double th1 = (double)(2.\*i + 1)\*k\*PI / (2.\*N);
412. double th2 = (double)(2.\*j + 1)\*l\*PI / (2.\*N);
413. sum += (double)cos(th1)\*cos(th2)\*data[y][x];
414. }
415. }
416. double ck;
417. if (l == 0 && k == 0) // l 과 k로 정의
418. ck = 1. / 8.;
419. else if (l != 0 && k != 0)
420. ck = 1. / 4.;
421. else
422. ck = sqrt(2.) / 8.;
423. dct[u][v] = ck \* sum;
424. }
425. }
426. }
427. }
428. ofstream Txt;//텍스트로 결과출력
429. Txt.open("DCT\_re.txt");
430. for (int i = 0; i < 256; i++)
431. {
432. for (int j = 0; j < 256; j++)
433. {
434. Txt << dct[i][j] << "\t";
435. }
436. Txt << endl;
437. }
438. }
439. void IDCT\_2D(double\*\*dct, unsigned char\*\*result\_data)
440. {
441. int N = 8;
442. int mcrNb = 32;
443. double sum = 0;
444. int u, v, y, x;
445. for (int mcr\_i = 0; mcr\_i < mcrNb; mcr\_i++)
446. {
447. for (int mcr\_j = 0; mcr\_j < mcrNb; mcr\_j++)
448. {
449. for (int i = 0; i < N; i++)
450. {
451. y = mcr\_i \* N + i;
452. for (int j = 0; j < N; j++)
453. {
454. sum = 0;
455. x = mcr\_j \* N + j;
456. for (int k = 0; k < N; k++)
457. {
458. u = mcr\_i \* N + k;
459. for (int l = 0; l < N; l++)
460. {
461. v = mcr\_j \* N + l;
462. double ck;
463. if (l == 0 && k == 0) // l 과 k로 정의
464. ck = 1. / 8;
465. else if (l != 0 && k != 0)
466. ck = 1. / 4;
467. else
468. ck = sqrt(2.) / 8.;
469. sum += ck \* cos((2. \* (double)i + 1)\*k\*PI / 16)\*cos((2. \* (double)j + 1)\*l\*PI / 16)\*dct[u][v];
470. }
471. }
472. if (sum < 0)
473. result\_data[y][x] = 0;
474. else if (sum > 255)
475. result\_data[y][x] = 255;
476. else
477. result\_data[y][x] = (int)sum;
478. }
479. }
480. }
481. }
482. }
483. 2차원 FFT+이미지파일 필터링
484. #include<iostream>
485. #include<fstream>
486. #include"complex.h"
487. using namespace std;
488. #define PI 3.141592
489. #define uchar unsigned char
490. #define H 256
491. #define W 256
492. void FFT2Radix(double\* Xr, double\* Xi, double\* Yr, double\* Yi, int nN, bool bInverse);
493. void FFT2D(uchar\*\* img, double\*\* OutputReal, double\*\* OutputImag, int nW, int nH);
494. void FFT2Dinverse(double\*\* InputReal, double\*\* InputImag, uchar\*\* OutputDouble, int nW, int nH);
495. void DNormalize2D(double \*\*p1, uchar \*\*p2, int nW, int nH);
496. int main() {
497. ifstream nzopen;
498. nzopen.open("twin\_noise.bmp", ios::binary);
499. char\* header;
500. uchar\*\* RGB;
501. uchar\*\* R;
502. uchar\*\* Rswch;
503. complex\*\* fft;
504. double\*\* mag;
505. double\*\* fftRe;
506. double\*\* fftIm;
507. uchar\*\* R\_;
508. uchar\*\* RGB\_;
509. header = new char[54];
510. RGB = new uchar\*[H];
511. R = new uchar\*[H];
512. Rswch = new uchar\*[H];
513. fft = new complex\*[H];
514. mag = new double\*[H];
515. fftRe = new double\*[H];
516. fftIm = new double\*[H];
517. R\_ = new uchar\*[H];
518. RGB\_ = new uchar\*[H];
519. for (int i = 0; i < H; i++) {
520. RGB[i] = new uchar[3 \* W];
521. R[i] = new uchar[W];
522. Rswch[i] = new uchar[W];
523. fft[i] = new complex[W];
524. mag[i] = new double[W];
525. fftRe[i] = new double[W];
526. fftIm[i] = new double[W];
527. R\_[i] = new uchar[W];
528. RGB\_[i] = new uchar[3 \* W];
529. }
530. nzopen.read((char\*)header, 54);
531. for (int i = 0; i < H; i++)
532. nzopen.read((char\*)RGB[i], 3 \* W);
533. for (int i = 0; i < H; i++) {
534. for (int j = 0, jj = 0; j < W; j++, jj += 3) {
535. R[i][j] = RGB[i][jj];
536. }
537. }
538. FFT2D(R, fftRe, fftIm, W, H);//R값을 fftRe, fftIm값과 관련지어줌
539. for (int i = 0; i < H; i++) {
540. for (int j = 0; j < W; j++) {
541. fft[i][j] = complex(fftRe[i][j], fftIm[i][j]);
542. mag[i][j] = 10 \* log(fft[i][j].mag() + 1);
543. }
544. }
545. DNormalize2D(mag, R\_, W, H);//R과 관련된 값을 R\_로 변경
546. for (int i = 0; i < H; i++) {
547. for (int j = 0, jj = 0; j < W; j++, jj += 3) {
548. RGB\_[i][jj] = R\_[i][j];
549. RGB\_[i][jj + 1] = R\_[i][j];
550. RGB\_[i][jj + 2] = R\_[i][j];
551. }
552. }
553. //재정렬, R\_값을 Rswch에 입력하고 Rswch값을 RGB\_에 입력
555. for(int i=0;i<H;i++)
556. for (int j = 0; j < W; j++)//원본기준으로 할 예정
557. {
558. if (i < 128 && j < 128)//왼위
559. Rswch[i + 128][j + 128] = R\_[i][j];
560. else if (i >= 128 && j >= 128)//오아래
561. Rswch[i - 128][j - 128] = R\_[i][j];
562. else if (i < 128 && j >= 128)//오위
563. Rswch[i + 128][j - 128] = R\_[i][j];
564. else//왼아래
565. Rswch[i - 128][j + 128] = R\_[i][j];
566. }
567. for (int i = 0; i < H; i++) {
568. for (int j = 0, jj = 0; j < W; j++, jj += 3) {
569. RGB\_[i][jj] = Rswch[i][j];
570. RGB\_[i][jj + 1] = Rswch[i][j];
571. RGB\_[i][jj + 2] = Rswch[i][j];
572. }
573. }
574. ofstream Outfile11;
575. Outfile11.open("노이즈전환버전.bmp", ios::binary);
576. Outfile11.write((char\*)header, 54);
577. for (int i = 0; i < H; i++)
578. Outfile11.write((char\*)RGB\_[i], 3 \* W);
579. //필터
580. for (int i = 0; i < H; i++)
581. {
582. for (int j = 0; j < W; j++)
583. {
584. if ((i > 22 && i < 28) || (i > 226 && i < 230) || (j > 39 && j < 45) || (j > 212 && j < 220))
585. {
586. R\_[i][j] = 0;
587. fftRe[i][j] = 0;
588. fftIm[i][j] = 0;
589. }
590. }
591. }
593. FFT2Dinverse(fftRe, fftIm, R\_, W, H);//출력까지, R\_에다가 변형된 값을 넣어줘야함
594. for (int i = 0; i < H; i++) {
595. for (int j = 0, jj = 0; j < W; j++, jj += 3) {
596. RGB\_[i][jj] = R\_[i][j];
597. RGB\_[i][jj + 1] = R\_[i][j];
598. RGB\_[i][jj + 2] = R\_[i][j];
599. }
600. }
601. ofstream Outfile2;
602. Outfile2.open("복원영상.bmp", ios::binary);
603. Outfile2.write((char\*)header, 54);
604. for (int i = 0; i < H; i++)
605. Outfile2.write((char\*)RGB\_[i], 3 \* W);
606. system("pause");
607. return 0;
608. }
609. void FFT2Radix(double\* Xr, double\* Xi, double\* Yr, double\* Yi, int nN, bool bInverse)
610. {
611. //N =8
612. double T, Wr, Wi;
613. if (nN <= 1) return;
614. for (int i = 0; i < nN; i++) {
615. Yr[i] = Xr[i];
616. Yi[i] = Xi[i];
617. }
618. int j = 0, k = 0;
619. for (int i = 1; i < (nN - 1); i++) { // (1) i = 1, j = 0, k = 0 (2) i =2, (3) i=3 (4) i =4,j=6 (5)i = 5, j =4
620. k = nN / 2; // // k = 4 (2) k=4 (3) k=4,j=2 (4) k = 4 (5) k=4
621. while (k <= j) {// (1) k=4, j=0 (x) (2) j =4, k=4 (o) (3) x (4) (o) (5) (o)
622. j = j - k; // (2) j =0 (4) j = 2 (5) j = 0
623. k = k / 2; // (2)k =2 (4) k = 2 \*5) k = 2
624. }
625. j = j + k; //(1) j = 4, (2) j = 2 (3) j=6, (4) j = 4,
626. if (i < j) { //(2) j =2 k=2 (3) i = 3, j=6 (4) j =4, i = 4
627. T = Yr[j]; // (1) i=1 j =4 swap, (3) 3, 6 swap
628. Yr[j] = Yr[i];
629. Yr[i] = T; // j = 4, i = 1 swap
630. T = Yi[j]; // j = 4, i = 1 swap
631. Yi[j] = Yi[i];
632. Yi[i] = T;// j = 4, i = 1 swap (1,4)
633. }
634. }
635. double Tr, Ti;
636. int iter, j2, pos;
637. k = nN >> 1; // (0) k= 1000<2> k = 100<2> =4
638. iter = 1;
639. while (k > 0) {
640. j = 0;
641. j2 = 0;
642. for (int i = 0; i < nN >> 1; i++) { // 0 ~i ~4 (1) i =0, (2) i=1 j=2, k =4 (3) i=2, j =4 (4) j =6,
643. Wr = cos(2.\*PI\*(j2\*k) / nN); // Wk(re)
644. if (bInverse == 0)
645. Wi = -sin(2.\*PI\*(j2\*k) / nN); //Wk(im)
646. else
647. Wi = sin(2.\*PI\*(j2\*k) / nN);
648. pos = j + (1 << (iter - 1)); // (1) pos = 1 = 1 (2)pos 2+1=3 (3) pos =5 j =4 (4) j =6, pos=7
649. Tr = Yr[pos] \* Wr - Yi[pos] \* Wi; // Y[2]
650. Ti = Yr[pos] \* Wi + Yi[pos] \* Wr;
651. Yr[pos] = Yr[j] - Tr; // (1) Y[pos=1], j = 0 (2) y[2] y[3]: X[2],X[6], (3)y[4],[5] (4) y[6],[7]
652. Yi[pos] = Yi[j] - Ti;
653. Yr[j] += Tr;
654. Yi[j] += Ti;
655. j += 1 << iter; // j= 0+ 10<2>= 2 (2) j = 2+ 2 = 4 =100(3) j=1000=6 , j =12
656. if (j >= nN) j = ++j2; //(1) x (2) (3) o j = 1, j2=1
657. }
658. k >>= 1; // (1) k = k>>1; 100=4 :: 10=2
659. iter++; // (1) iter=2
660. }
661. if (bInverse) {
662. for (int i = 0; i < nN; i++) {
663. Yr[i] /= nN;
664. Yi[i] /= nN;
665. }
666. }
667. }
668. void FFT2D(uchar\*\* img, double\*\* OutputReal, double\*\* OutputImag, int nW, int nH)
669. {
670. int x, y;
671. double \*dRealX, \*dImagX;
672. double \*dRealY, \*dImagY;
673. dRealX = new double[nW];
674. dImagX = new double[nW];
675. dRealY = new double[nW];
676. dImagY = new double[nW];
677. for (y = 0; y < nH; y++) {
678. for (x = 0; x < nW; x++) {
679. dRealX[x] = img[y][x];
680. dImagX[x] = 0.;
681. }
682. FFT2Radix(dRealX, dImagX, dRealY, dImagY, nW, false);
683. for (x = 0; x < nW; x++) {
684. OutputReal[y][x] = dRealY[x];
685. OutputImag[y][x] = dImagY[x];
686. }
687. }
688. delete[] dRealX;
689. delete[] dImagX;
690. delete[] dRealY;
691. delete[] dImagY;
692. dRealX = new double[nH];
693. dImagX = new double[nH];
694. dRealY = new double[nH];
695. dImagY = new double[nH];
696. for (x = 0; x < nW; x++) {
697. for (y = 0; y < nH; y++) {
698. dRealX[y] = OutputReal[y][x];
699. dImagX[y] = OutputImag[y][x];
700. }
701. FFT2Radix(dRealX, dImagX, dRealY, dImagY, nH, false);
702. for (y = 0; y < nH; y++) {
703. OutputReal[y][x] = dRealY[y];
704. OutputImag[y][x] = dImagY[y];
705. }
706. }
707. delete[] dRealX;
708. delete[] dImagX;
709. delete[] dRealY;
710. delete[] dImagY;
711. }
712. void FFT2Dinverse(double\*\* InputReal, double\*\* InputImag, uchar\*\* OutputDouble, int nW, int nH)
713. {
714. int x, y;
715. double \*dRealX, \*dImagX;
716. double \*dRealY, \*dImagY;
717. double\*\* OutputReal, \*\*OutputImag;
718. OutputReal = new double\*[nH];
719. OutputImag = new double\*[nH];
720. for (int i = 0; i < nH; i++) {
721. OutputReal[i] = new double[nW];
722. OutputImag[i] = new double[nW];
723. }
724. dRealX = new double[nW];
725. dImagX = new double[nW];
726. dRealY = new double[nW];
727. dImagY = new double[nW];
728. for (y = 0; y < nH; y++) {
729. for (x = 0; x < nW; x++) {
730. dRealX[x] = InputReal[y][x];
731. dImagX[x] = InputImag[y][x];
732. }
733. FFT2Radix(dRealX, dImagX, dRealY, dImagY, nW, true);
734. for (x = 0; x < nW; x++) {
735. OutputReal[y][x] = dRealY[x];
736. OutputImag[y][x] = dImagY[x];
737. }
738. }
739. delete[] dRealX;
740. delete[] dImagX;
741. delete[] dRealY;
742. delete[] dImagY;
743. dRealX = new double[nH];
744. dImagX = new double[nH];
745. dRealY = new double[nH];
746. dImagY = new double[nH];
747. for (x = 0; x < nW; x++) {
748. for (y = 0; y < nH; y++) {
749. dRealX[y] = OutputReal[y][x];
750. dImagX[y] = OutputImag[y][x];
751. }
752. FFT2Radix(dRealX, dImagX, dRealY, dImagY, nH, true);
753. for (y = 0; y < nH; y++) {
754. OutputReal[y][x] = dRealY[y];
755. OutputImag[y][x] = dImagY[y];
756. }
757. }
758. delete[] dRealX;
759. delete[] dImagX;
760. delete[] dRealY;
761. delete[] dImagY;
762. for (y = 0; y < nH; y++) {
763. for (x = 0; x < nW; x++) {
764. OutputDouble[y][x] = OutputReal[y][x];
765. }
766. }
767. for (int i = 0; i < nH; i++) {
768. delete[] OutputReal[i];
769. delete[] OutputImag[i];
770. }
771. delete[] OutputReal;
772. delete[] OutputImag;
773. }
774. void DNormalize2D(double \*\*p1, uchar \*\*p2, int nW, int nH)//p2에 값 입력, p1에 mag입력
775. {
776. int x, y;
777. double min = 9999.;
778. double max = -9999.;
779. double val;
780. for (y = 0; y < nH; y++)
781. {
782. for (x = 0; x < nW; x++)
783. {
784. val = p1[y][x];
785. if (val > max) max = val;
786. if (val < min) min = val;
787. }
788. }
789. if (max == min)
790. {
791. for (y = 0; y < nH; y++)
792. {
793. for (x = 0; x < nW; x++)
794. p2[y][x] = 0;
795. }
796. return;
797. }
798. double dfactor = 255 / (max - min);
799. for (y = 0; y < nH; y++)
800. {
801. for (x = 0; x < nW; x++)
802. p2[y][x] = (unsigned char)((p1[y][x] - min)\*dfactor);
803. }
804. }
805. 2차원 DFT+Z영역 이미지파일 필터링(디리클레)
806. #include<iostream>
807. #include<fstream>
808. #include"complex.h"
809. using namespace std;
810. #define PI 3.141592
811. void DFT(int, complex\*, complex\*);
812. void IDFT(int, complex\*, short\*);
813. void DoDirichlet(int, int, complex\*, complex\*);
814. int main()
815. {
816. ifstream Infile;//읽어올 것
817. Infile.open("noiseMusic.wav", ios::binary);
818. char\* header = new char[44];
819. Infile.read((char\*)header, 44);//헤더작성+필요한 값 추출
820. unsigned int fs= \*(unsigned int\*)(header + 24);
821. cout << "fs=" << fs << endl;//8000
822. int size = (int)((\*(unsigned int\*)(header + 4))) + 8 - 44;
823. cout << "size=" << size << endl;//32000
825. short\*data = new short[size/2];
826. Infile.read((char\*)data, size);//여기까지 header와 data저장
827. complex\* compdata=new complex[size/2];
828. for (int i = 0; i < size/2; i++)
829. {
830. compdata[i] = complex(data[i], 0);
831. }
832. complex \*dft = new complex[fs\*2];//dft
833. DFT(fs, compdata, dft);
834. ofstream dftfile("dft.txt");
835. for (int i = 0; i < 2 \* fs; i++)
836. dftfile << dft[i].mag() << endl;
837. complex\*output = new complex[fs \* 2];//디리디리
838. int L = (int)((double)fs/1000);//L=8
839. DoDirichlet(L, fs, dft, output);
841. short\*result = new short[size/2];//idft
842. IDFT(fs, output, result);
843. ofstream Filtered("Filted.wav", ios::binary);
844. Filtered.write((char\*)header, 44);
845. Filtered.write((char\*)result, 32000);
846. return 0;
847. }
848. void DFT(int fs, complex\* data, complex\* dft)
849. {
850. int Fs = 2 \* fs;
851. cout << "DFT 수행" << endl;
852. for (int k = 0; k < Fs; k++)
853. {
854. for (int n = 0; n < Fs; n++)
855. {
856. dft[k] += data[n] \* complex(cos((-2. \* PI\*k\*n) / (double)Fs), sin((-2. \* PI\*k\*n) / (double)Fs));
857. }
858. }
859. }
860. void IDFT(int fs, complex\*output, short\*result)
861. {
862. int Fs = 2 \* fs;
863. complex\* x=new complex[Fs];
864. cout << "IDFT 수행" << endl;
865. for (int n = 0; n < Fs; n++)
866. {
867. for (int k = 0; k < Fs; k++)
868. {
869. x[n] += output[k] \* complex(cos((2. \* PI\*k\*n) / (double)Fs), sin((2. \* PI\*k\*n) / (double)Fs));
870. }
871. x[n] = x[n]/(double)(Fs);
872. }
873. for (int i = 0; i < Fs; i++)
874. result[i] = x[i].re;
875. }
876. void DoDirichlet(int L, int fs, complex\* dft, complex\* output)
877. {
878. cout << "디리클레 수행" << endl;
879. complex \*H = new complex[2\*fs];
880. complex upper, bottom;
881. double lim = 8.;
882. for (int k = 0; k < 2\*fs; k++)
883. {
884. bottom = complex(sin(2.\*PI\*k / (double)(4.\*fs)), 0);
885. if (bottom.mag() == 0.0)
886. {
887. H[k] = complex(cos(-2.\*PI\*k\*((L - 1) / 2.) / (double)(2\*fs)),
888. sin(-2.\*PI\*k\*((L - 1) / 2.) / (double)(2\*fs)))\*lim;
889. }
890. else
891. {
892. upper = complex(sin(2.\*PI\*k\*L / (double)(4.\*fs)), 0.0);
893. H[k] = upper / bottom \* complex(cos(-2.\*PI\*k\*((L - 1) / 2.) / (double)(2\*fs)),
894. sin(-2.\*PI\*k\*((L - 1) / 2.) / (double)(2\*fs)));
895. }
896. }
897. ofstream Hfile("Hfil.txt");
898. for (int i = 0; i < 2 \* fs; i++)
899. Hfile << H[i].mag() << endl;
900. for (int k = 0; k < 2\*fs; k++)
901. {
902. output[k] = dft[k]\* H[k]/5.;//증폭되는효과발생하므로 사이즈를 줄여줘도 좋음
903. }
904. }
905. 2차원 DFT+Z영역 이미지파일 필터링(특정 주파수 필터)
906. #include<iostream>
907. #include<fstream>
908. #include"complex.h"
909. using namespace std;
910. #define PI 3.141592
911. void DFT(int, complex\*, complex\*);
912. void IDFT(int, complex\*, short\*);
913. void Filter(int, complex\*, complex\*);
914. int main()
915. {
916. ifstream Infile;//읽어올 것
917. Infile.open("noiseMusic.wav", ios::binary);
918. char\* header = new char[44];
919. Infile.read((char\*)header, 44);//헤더작성+필요한 값 추출
920. unsigned int fs = \*(unsigned int\*)(header + 24);
921. cout << "fs=" << fs << endl;//8000
922. int size = (int)((\*(unsigned int\*)(header + 4))) + 8 - 44;
923. cout << "size=" << size << endl;//32000
924. short\*data = new short[size / 2];
925. Infile.read((char\*)data, size);//여기까지 header와 data저장
926. complex\* compdata = new complex[size / 2];
927. for (int i = 0; i < size / 2; i++)
928. {
929. compdata[i] = complex(data[i], 0);
930. }
931. complex \*dft = new complex[fs \* 2];//dft
932. DFT(fs, compdata, dft);
933. ofstream dftfile("dft.txt");
934. for (int i = 0; i < 2 \* fs; i++)
935. dftfile << dft[i].mag() << endl;
936. complex\*output = new complex[fs \* 2];//필터링
937. Filter(fs, dft, output);
938. short\*result = new short[size / 2];//idft
939. IDFT(fs, output, result);
940. ofstream Filtered("Filted.wav", ios::binary);
941. Filtered.write((char\*)header, 44);
942. Filtered.write((char\*)result, 32000);
943. return 0;
944. }
945. void DFT(int fs, complex\* data, complex\* dft)
946. {
947. int Fs = 2 \* fs;
948. cout << "DFT 수행" << endl;
949. for (int k = 0; k < Fs; k++)
950. {
951. for (int n = 0; n < Fs; n++)
952. {
953. dft[k] += data[n] \* complex(cos((-2. \* PI\*k\*n) / (double)Fs), sin((-2. \* PI\*k\*n) / (double)Fs));
954. }
955. }
956. }
957. void IDFT(int fs, complex\*output, short\*result)
958. {
959. int Fs = 2 \* fs;
960. complex\* x = new complex[Fs];
961. cout << "IDFT 수행" << endl;
962. for (int n = 0; n < Fs; n++)
963. {
964. for (int k = 0; k < Fs; k++)
965. {
966. x[n] += output[k] \* complex(cos((2. \* PI\*k\*n) / (double)Fs), sin((2. \* PI\*k\*n) / (double)Fs));
967. }
968. x[n] = x[n] / (double)(Fs);
969. }
970. for (int i = 0; i < Fs; i++)
971. result[i] = x[i].re;
972. }
973. void Filter(int fs, complex\* dft, complex\* output)
974. {
975. ofstream out\_mag;
976. out\_mag.open("mag.txt");
977. complex\* H = new complex[2 \* fs];
978. complex\* Z = new complex[2 \* fs];
979. complex zero1 = complex(2 \* PI \* 0 / 16000)\*0.9;
980. complex zero2 = complex(2 \* PI \* 8000 / 16000)\*0.9;
981. complex pole1 = complex(2 \* PI \* 4000 / 16000)\*0.9;
982. complex pole2 = complex(2 \* PI \* 12000 / 16000)\*0.9;
983. for (int k = 0; k < 2 \* fs; k++)
984. {
985. Z[k] = complex(2 \* PI\*k / (double)(2 \* fs));
986. H[k] = (Z[k] - zero1)\*(Z[k] - zero2) / ((Z[k] - pole1)\*(Z[k] - pole2));
987. }
988. for (int k = 0; k < 2 \* fs; k++)
989. out\_mag << H[k].mag() << endl;
990. }