

PTC - EP - USP
Princípios da Formação e Processamento de Imagens 2009
Listas de exercícios 01, 02 e 03

Leandro M Biondo - 5090095

Prof. Sérgio S Furuie

20/09/09

1 Lista 01, ImageJ.

Para inverter as cores de uma imagem em escala de cinza 8bits foi feito o seguinte plugin, que subtrai o valor de cada pixel do máximo possível para ele (255), e sobre escreve o resultado no mesmo pixel (seu inverso).

```
* Inversão de tonalidades em escala de cinza */
public static ImageAccess inv_ton(ImageAccess in, double inv_ton) {
    int nx = in.getWidth();
    int ny = in.getHeight();
    ImageAccess out = new ImageAccess(nx, ny);

    // codigo
    for (int i=0; i<nx; i++)
        for (int k=0; k<ny; k++)
            out.putPixel(i, k, 255-in.getPixel(i, k));
    IJ.write("Feito");

return out;
}
```

Exemplo de imagem após aplicado o filtro:



Figura 1: Imagem Bridge.gif à esquerda e o resultado do plugin “inverte_tons.class” à direita.

2 Lista 02, Eclipse.

Para fazer o histograma de uma imagem de 8bits foi executado um teste em todos os pixels da imagem para cada tonalidade possível (seria mais eficiente testar todas as tonalidades para cada pixel), o resultado é apresentado na janela Results do ImageJ, assim com a média e o desvio padrão, calculados com os valores do vetor que armazena a contagem de pixels.

```
public static ImageAccess dohistograma(ImageAccess in, double doh) {
    int nx = in.getWidth();
    int ny = in.getHeight();
    ImageAccess out = new ImageAccess(nx, ny);
    IJ.write("Executando Histograma");
    // codigo
    double[] histog=new double[255];
    for(int afe=0; afe<255; afe++){
        histog[afe]=0;
        for(int i=0;i<nx; i++){
            for(int k=0; k<ny; k++){
                double teste=afe;
                if(in.getPixel(i, k)==teste){
                    histog[afe]++;
                }
            }
        }
    }
    //MEDIA
    double media=0;
    for(int afe=0; afe<255; afe++){
        media=media+histog[afe]*afe;
    }
    media=media/(nx*ny);

    //Desvio Padrao
    double desvio=0;
    for(int afe=0; afe<255; afe++){
        desvio=desvio+histog[afe]*((media-afe)*(media-afe));
    }
    desvio=Math.sqrt(desvio/(nx*ny-1));

    //IMPRESSAO
    String impmedia= "Cor média: "+String.valueOf(media);
    IJ.write(impmedia);
    String impdesvio= "Desvio Padrão: "+String.valueOf(desvio);
    IJ.write(impdesvio);
    IJ.write("HISTOGRAMA");
    String titulo="Cor      Frequencia ";
    IJ.write(titulo);
    for(int tom=0; tom<255; tom++){
        String imprime= String.valueOf(tom)+"          "+String.valueOf(histog[tom]);
        IJ.write(imprime);
    }
    return out;
}
```

Imagem Yatch.tiff



Executando Histograma

Desvio Padrão: 49,20152383700377 Cor média: 100,40881005646106

HISTOGRAMA

Cor	Cont.	Cor	Cont.	Cor	Cont.	Cor	Cont.	Cor	Cont.	Cor	Cont.	Cor	Cont.	Cor	Cont.	Cor	Cont.	Cor	Cont.
0	13	26	746	52	601	78	1325	104	1101	130	887	156	306	182	320	208	111	234	2
1	47	27	812	53	635	79	1261	106	1137	131	774	157	297	183	349	209	102	235	1
2	41	28	832	54	667	80	1256	106	1212	132	644	158	266	184	275	210	85	236	0
3	62	29	845	55	678	81	1257	107	1425	133	519	159	281	185	288	211	53	237	1
4	88	30	785	56	744	82	1273	108	1485	134	466	160	224	186	363	212	56	238	0
5	96	31	787	57	842	83	1329	109	1598	135	444	161	195	187	498	213	53	239	0
6	124	32	760	58	811	84	1297	110	1678	136	485	162	196	188	585	214	38	240	0
7	156	33	727	59	765	85	1333	111	1647	137	456	163	214	189	748	215	30	241	0
8	138	34	697	60	772	86	1349	112	1667	138	541	164	195	190	839	216	46	242	0
9	132	35	634	61	729	87	1360	113	1521	139	615	165	236	191	982	217	34	243	0
10	164	36	592	62	776	88	1396	114	1436	140	629	166	237	192	924	218	46	244	0
11	228	37	607	63	820	89	1367	115	1284	141	680	167	247	193	823	219	40	245	0
12	248	38	582	64	854	90	1365	116	1284	142	643	168	261	194	657	220	30	246	0
13	302	39	517	65	905	91	1390	117	1042	143	634	169	289	195	565	221	30	247	0
14	367	40	525	66	1057	92	1392	118	945	144	684	170	265	196	600	222	27	248	0
15	475	41	514	67	1023	93	1282	119	846	145	727	171	325	197	555	223	26	249	0
16	522	42	516	68	1036	94	1268	120	802	146	816	172	363	198	551	224	22	250	0
17	544	43	491	69	1113	95	1241	121	770	147	793	173	414	199	512	225	20	251	0
18	588	44	474	70	1210	96	1314	122	770	148	671	174	464	200	534	226	24	252	0
19	632	45	442	71	1217	97	1247	123	853	149	544	175	630	201	527	227	29	253	0
20	636	46	460	72	1348	98	1286	124	822	150	413	176	741	202	438	228	13	254	0
21	650	47	483	73	1407	99	1165	125	774	151	379	177	613	203	304	229	5	255	0
22	667	48	486	74	1337	100	1237	126	808	152	325	178	458	204	212	230	3		
23	694	49	448	75	1306	101	1073	127	878	153	326	179	362	205	183	231	3		
24	704	50	527	76	1271	102	1146	128	861	154	320	180	351	206	142	232	1		
25	755	51	596	77	1298	103	1039	129	881	155	302	181	337	207	126	233	0		

Figura 2: Resultado do Plugin “histograma_.class” aplicado à imagem “yacht.tif”. O resultado foi reorganizado em forma de tabela, no ImageJ foi apresentado como uma lista de texto.

Executando Histograma

Desvio Padrão: 55,90130483086753 Cor média: 86,38841043307086

HISTOGRAMA

Cor	Cont.	Cor	Cont.	Cor	Cont.	Cor	Cont.	Cor	Cont.	Cor	Cont.	Cor	Cont.	Cor	Cont.	Cor	Cont.	Cor	Cont.
0	0	27	0	54	0	81	0	108	0	135	0	162	0	189	0	216	1992	243	0
1	0	28	0	55	0	82	0	109	0	136	0	163	0	190	0	217	0	244	0
2	0	29	0	56	0	83	0	110	0	137	0	164	0	191	0	218	0	245	0
3	0	30	0	57	0	84	0	111	0	138	0	165	0	192	3829	219	0	246	0
4	0	31	0	58	0	85	0	112	0	139	0	166	0	193	0	220	0	247	0
5	0	32	0	59	0	86	0	113	0	140	0	167	0	194	0	221	0	248	0
6	0	33	0	60	0	87	0	114	0	141	0	168	7456	195	0	222	0	249	0
7	0	34	0	61	0	88	0	115	0	142	0	169	0	196	0	223	0	250	0
8	0	35	0	62	0	89	0	116	0	143	0	170	0	197	0	224	0	251	0
9	0	36	0	63	0	90	0	117	0	144	10396	171	0	198	0	225	0	252	0
10	0	37	0	64	0	91	0	118	0	145	0	172	0	199	0	226	0	253	0
11	0	38	0	65	0	92	0	119	0	146	0	173	0	200	0	227	0	254	0
12	0	39	0	66	0	93	0	120	7731	147	0	174	0	201	0	228	0	255	0
13	0	40	0	67	0	94	0	121	0	148	0	175	0	202	0	229	0		
14	0	41	0	68	0	95	0	122	0	149	0	176	0	203	0	230	0		
15	0	42	0	69	0	96	3511	123	0	150	0	177	0	204	0	231	0		
16	0	43	0	70	0	97	0	124	0	151	0	178	0	205	0	232	0		
17	0	44	0	71	0	98	0	125	0	152	0	179	0	206	0	233	0		
18	0	45	0	72	1209	99	0	126	0	153	0	180	0	207	0	234	0		
19	0	46	0	73	0	100	0	127	0	154	0	181	0	208	0	235	0		
20	0	47	0	74	0	101	0	128	0	155	0	182	0	209	0	236	0		
21	0	48	304	75	0	102	0	129	0	156	0	183	0	210	0	237	0		
22	0	49	0	76	0	103	0	130	0	157	0	184	0	211	0	238	0		
23	0	50	0	77	0	104	0	131	0	158	0	185	0	212	0	239	0		
24	53	51	0	78	0	105	0	132	0	159	0	186	0	213	0	240	1394		
25	0	52	0	79	0	106	0	133	0	160	0	187	0	214	0	241	0		
26	0	53	0	80	0	107	0	134	0	161	0	188	0	215	0	242	0		

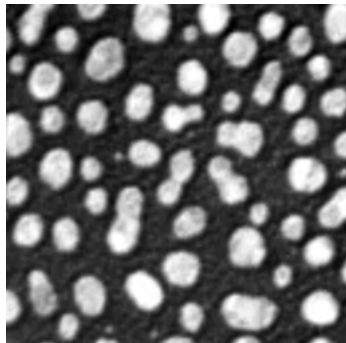


Figura 3: Resultado do Plugin “histograma_.class” aplicado à imagem “blobs1.tif”. O resultado foi reorganizado em forma de tabela, no ImageJ foi apresentado como uma lista de texto.

A seguir foi feito um plugin que faz a média local, gerando uma matriz NxN (N ímpar, dado pelo usuário) de pesos iguais ($Peso_{ij} = \frac{1}{(N \times N)}$) e fazendo a sua convolução com todos os pixels da imagem aberta.

MÉDIA EM TORNO DE CADA PIXEL

```
//usado codigo da aula sobre eclipse com uma entrada para o tamanho do kernel a convolu
public void run(ImageProcessor ip) {
// criando e usando o ImageAccess
```

```

ImageAccess im_acc=new ImageAccess(ip);
// obtendo as dimensoes da imagem
int C= im_acc.getWidth();
int R=im_acc.getHeight();
GenericDialog gd = new GenericDialog("Lados da matriz média");
gd.addNumericField("Dimensoes para media (impar)", 3, 0);
gd.showDialog();
int N = (int) gd.getNextNumber();
int nc=N/2;

// criando novo array p/ nao interferir no original
double[ ][ ] pesos=new double[N][N];
for(int r=0; r<N; r++)
    for(int c=0; c<N; c++) pesos[r][c]=1.0/(N*N);

// realizando a media
for(int r=0; r< R; r++) // considera bordas
    for (int c=0; c<C; c++) // considera bordas
    {
        double media=0;
        for(int r1=-nc; r1<=nc; r1++)
            for(int c1=-nc; c1<=nc; c1++)
            {
                media=media + pesos[r1+nc][c1+nc]*im_acc.getPixel(c-c1,r-r1);
                im_acc.putPixel(c,r, media);
            }
    }
im_acc.show("Resultado");
}

```



Figura 4: Imagem gerada à partir de uma fotografia, para seu uso foi redefinida para 8bits e o tamanho reduzido para 1279x850 pixels. A primeira imagem é a original, e as seguintes tiveram seus pixels substituídos pela média em seu entorno. As convoluções foram feitas, respectivamente, com matrizes 3x3, 5x5, 9x9, 17x17 e 33x33.

Aproveitando o mesmo plugin, foi criado um outro que faz o gradiente em cada ponto, pedindo ao usuário um número N ímpar a ser usado na convolução de uma matriz para a derivada parcial horizontal (N linhas com colunas -1, 0 e 1) e vertical (N colunas com linhas $\begin{matrix} -1 \\ 0 \\ 1 \end{matrix}$), foram usadas duas matrizes peso com os tamanhos adequados, e os sinais foram adicionados ao percorrer a figura nas leituras de pixels.

MÓDULO DO GRADIENTE EM CADA PIXEL.

```
public void run(ImageProcessor ip) {
    // criando e usando o ImageAccess
    ImageAccess im_acc=new ImageAccess(ip);
    // obtendo as dimensoes da imagem
    int C= im_acc.getWidth();
```

```

int R=im_acc.getHeight();
GenericDialog gd = new GenericDialog("Lados da matriz média");
gd.addNumericField("Dimensoes para media (impar)", 3, 0);
gd.showDialog();
int N = (int) gd.getNextNumber();
int nc=N/2;

// criando novo array p/ nao interferir no original
double[ ][ ] pesosx=new double[3][N];
double[ ][ ] pesosity=new double[N][3];
double[ ][ ] grad=new double[C][R];
for(int r=0; r<3; r++)
    for(int c=0; c<N; c++) pesosx[r][c]=1.0/(3*N);
for(int r=0; r<3; r++)
    for(int c=0; c<N; c++) pesosity[c][r]=1.0/(3*N);

// realizando derivada em x
for(int r=0; r< R; r++)
    for (int c=0; c<C; c++)
    {
        double dx=0;
        for(int r1=-1; r1<=1; r1++)
            for(int c1=-nc; c1<=nc; c1++)
            {
                dx=dx + r1*pesosx[r1+1][c1+nc]*im_acc.getPixel(c-c1,r-r1);
                grad[c][r]=dx*dx;
            }
    }

// realizando derivada em y
for(int r=0; r< R; r++)
    for (int c=0; c<C; c++)
    {
        double dy=0;
        for(int r1=-nc; r1<=nc; r1++)
            for(int c1=-1; c1<=1; c1++)
            {
                dy=dy + c1*pesosity[r1+nc][c1+1]*im_acc.getPixel(c-c1,r-r1);
                grad[c][r]=Math.sqrt(grad[c][r]+dy*dy);
                im_acc.putPixel(c,r, grad[c][r]);
            }
    }
im_acc.show("Resultado");
}

```

3 Lista 03 - Transformada de Fourier

3.1 FFT direct

Pode-se notar o aparecimento de padrões de repetição na figura da transformada, que é equivalente a uma figura de difração de um computador ótico, então o esperado é que os padrões verticais apareçam na horizontal no plano de fourier e os horizontais apareçam na vertical. Além disso, o centro da transformada (região com mais informações) tende a ter a forma predominante da imagem original.



Figura 5: Resultados do gradiente aplicado à figura usada na média, com N igual a 3, 9 e 17. O aumento do comprimento das matrizes de convolução fazem com que sejam destacados trechos mais homogêneos com relação às derivadas vertical e horizontal, selecionando contornos de objetos maiores na imagem.

No caso dos retângulos brancos sobre fundo preto, é fácil perceber que a imagem de intensidade da transformada tem um padrão na mesma forma do retângulo, girado 90 graus, que se repete. Pelas `rect3` e `rect4` é perceptível que a mudança de posição não altera o módulo das frequências que formam a figura, mas altera a fase delas.

No caso das figuras naturais é possível encontrar alguns padrões, em especial ao longo dos eixos da transformada há o acúmulo dos detalhes da figura, nos quadrantes aparecem ruídos ou respostas à componentes da imagem que não seguem padrão.

3.2 FFT inverse

Para uma aplicação correta da transformada de fourier, o esperado é que não seja perdida informação entre a aplicação da FFT e de sua inversa, ao aplicar a FFT `direct` e em seguida aplicar a FFT `inverse` não encontrei diferença entre a original e o resultado, para ter certeza subtraí uma imagem da outra e obtive uma imagem completamente escura, toda preenchida por zeros.

O histograma desta imagem obtida comprovou que o máximo e o mínimo da imagem era 0, indicando assim que o plugin FFT `direct` e o `Inverse` funcionam corretamente. Ao aplicar filtros de valor 0 sobre a transformada (após aplicar FFT) foi possível retirar padrões da imagem, resultado visível após aplicar a inversa na transformada filtrada. A seguir um resultado obtido com a FFT do próprio ImageJ, realizadas no primeiro semestre de 2009, na disciplina FGE0214 do Instituto de Física (obtive resultados equivalentes com a FFT `direct` e `inve`se).



Figura 6: Imagens do Fourier com aplicação de filtro para frequências baixas, ruídos e altas. A primeira usa um filtro que bloqueia as frequências baixas no plano de fourier, o segundo retira o ruído nos 4 quadrantes e a terceira retira apenas o centro, deixando passar frequências médias e baixas. As imagens apresentadas são a original, a filtrada e uma menos a outra. Figuras geradas no ImageJ com o uso da FFT.

3.3 FFT - Fase e Módulo

Foram feitas a FFT direct da figura cat.tif, e a FFT inverse entre a sua fase e a figura zero.tif e o mesmo com o seu módulo.

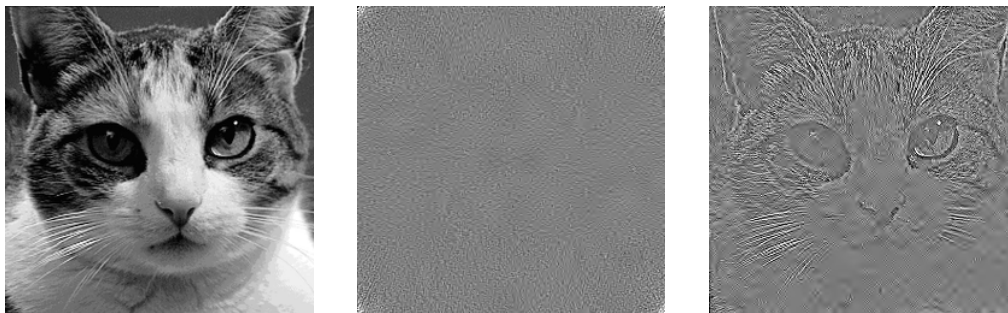


Figura 7: Figura cat.tif, resultado após aplicar a FFT direct e posteriormente a FFT inverse apenas com o módulo e apenas com a fase. Quando se utiliza apenas o módulo a figura fica com intensidades variáveis distribuídas igualmente em toda a imagem, pois não há informações sobre a posição correta de cada componente da imagem. Já quando utilizamos apenas a fase, o desenho é distinguível mas não tem os tons de cor corretos, pois a intensidade usada é igual para toda a figura.

Depois utilizadas as imagens cat.tif e lena.tif, para misturar a fase e o módulo de ambas.

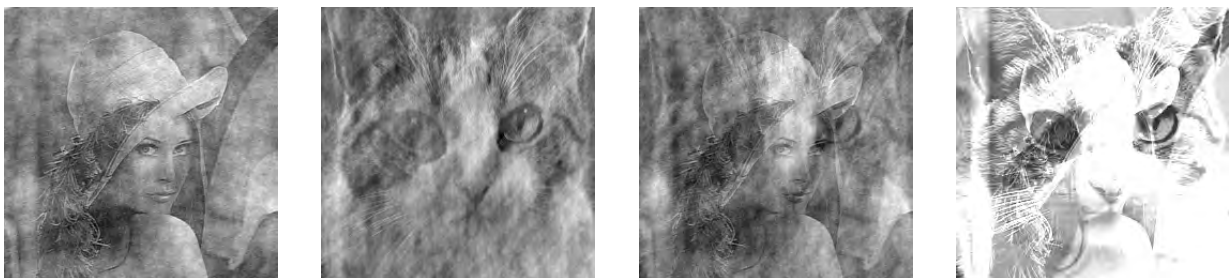


Figura 8: A primeira imagem utiliza a fase do cat.tif e o módulo da lena.tif, a segunda a fase da lena.tif e módulo cat.tif . A terceira imagem é uma soma das duas, e a quarta a soma das duas imagens originais, ao comparar estas últimas a imagem mais clara obtida à partir das originais indica que houve perda de intensidade total das imagens, mais que isso, como a imagem cat.tif tinha intensidade maior, ao juntar o seu módulo com a fase de lena.tif, percebe-se que a imagem da moça ficou mais destacada (terceira imagem) do que originalmente (quarta, em que o gato sobressai).

3.4 FourierProgressiveReconstruction

Na análise de reconstrução progressiva foi usada a imagem do fourier, depois visualmente comparados os resultados obtidos com a aplicação de cada opção do FourierProgressiveReconstruction, a imagem mais parecida com a original (utilizando 5000 coeficientes) foi a com maiores coeficientes (largest).

Após essa conclusão foi feita a FFT (do ImageJ) de cada imagem recuperada, que é composta apenas pelas partes executadas pela reconstrução progressiva. Pela comparação das FFT's vê-se que a de coeficientes maiores cobre a maior parte da área central da FFT, região com mais informações relevantes por área no plano de fourier. Portanto, com essa reconstrução perde-se um pouco de detalhes mas a maior parte das informações é recuperada rapidamente (com menos iterações).

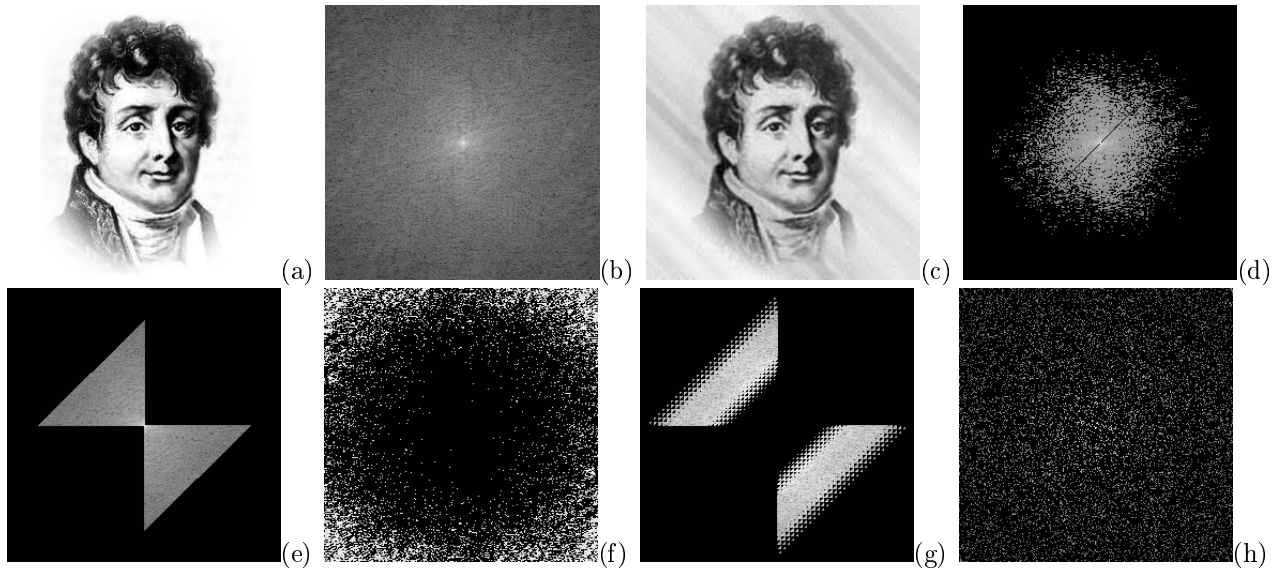


Figura 9: Resultados da FourierProgressiveReconstruction. (a)Imagem original;(b)FFT da original;(c)Resultado obtido com Largest Coefficients;(d)FFT do resultado de Largest Coeff.;(e)FFT do resultado de lowest frequency;(f)FFT de Smallest Coefficients;(g)FFT de Medium Frequency;(h)FFT de Random.

Referências

- [1] Material de aula de PTC2892-2009, Prof. Sérgio S Furuie.
- [2] Documentação ImageJ - <http://rsb.info.nih.gov/ij/docs/>
- [3] Centro para desenvolvedores Java, Sun Microsystems.<http://java.sun.com/>