Medidas de distorção de imagens:

F: Imagem de entrada (ou original) de tamanho $J \times K$
$\hat{F}$: Imagem de saída (ou processada ou estimada) de tamanho $J \times K$

1) MAE: Mean absolute error

$$MAE = \frac{\sum_j \sum_k \left| F(j,k) - \hat{F}(j,k) \right|}{J \times K}$$

Nota: Para expressar o erro em %, basta multiplicar MAE por 100/255.

2) MSE: Mean square error

$$MSE = \frac{\sum_j \sum_k \left(F(j,k) - \hat{F}(j,k) \right)^2}{J \times K}$$

3) RMSE: Root mean square error

$$RMSE = \sqrt{MSE}$$

Nota: Para expressar o erro em %, basta multiplicar RMSE por 100/255.

4) PSNR: Peak signal to noise ratio (dado em decibéis dB). Há duas definições diferentes em uso, sendo que a mais usada (para imagens em níveis de cinza) parece ser a segunda definição:

a) $$PSNR = 10 \log_{10} \left(\frac{\left[\max\{F(j,k)\} \right]^2}{MSE} \right) = 20 \log_{10} \left(\frac{\max\{F(j,k)\}}{\sqrt{MSE}} \right)$$

b) $$PSNR = 10 \log_{10} \left(\frac{255^2}{MSE} \right) = 20 \log_{10} \left(\frac{255}{RMSE} \right)$$ (definição mais amplamente usada)

5) SNR: Signal to noise ratio

$$SNR = 10 \log_{10} \left(\frac{\sum_j \sum_k \left(F(j,k) \right)^2 \Big/ (J \times K)}{MSE} \right)$$

6) Noise reduction factor

$$\delta = \sqrt{\frac{\sum_n \left[x(n) - s(n) \right]^2}{\sum_n \left[\hat{x}(n) - s(n) \right]^2}}$$

where s is the original noiseless signal, x is the noisy signal and $\hat{x}$ is the filtered signal.

7) SSIM - structural similarity [Wang, 2004]

Nota: Para imagens coloridas no sistema RGB, as somatórias devem ser calculadas nas 3 bandas e divididas por $3 \times J \times K$.

Nota: Para imagens coloridas usando SSIM, converter RGB para CieLAB e utilizar CSSIM [Araujo2011].

Nota: No sistema ProEikon, o comando IMG DistG calcula a distorção entre duas imagens em níveis de cinzas.
C:\>IMG distg

Referências:
[1] http://en.wikipedia.org/wiki/PSNR
[2] http://bmrc.berkeley.edu/courseware/cs294/fall97/assignment/psnr.html
[Wang2004] Zhou Wang, Alan Conrad Bovik, Hamid Rahim Sheikh, and Eero P. Simoncelli, *Image Quality Assessment: From Error Visibility to Structural Similarity*, *IEEE T. Image Processing*, Vol. 13, No. 4, pp. 600-612, April 2004.
[Araujo2011] S. A. Araújo and H. Y. Kim, "Ciratefi: An RST-Invariant Template Matching with Extension to Color Images,"Integrated Computer-Aided Engineering, vol. 18, no. 1, pp. 75-90, 2011.

Structural Similarity:

Execute: (Nota: O comando D_SSIMG só está disponível a partir do Proeikon 4.09).
C:\>IMG D_SSIMG

Veja as figuras do artigo [Wang, 2004].

Queremos comparar as imagens em níveis de cinza X e Y. Vamos comparar usando janela móvel.

Sejam x e y os vetores obtidos copiando pixels dentro da janela das imagens X e Y.

$$l(x,y)=\frac{2\mu_x\mu_y+C_1}{\mu_x^2+\mu_y^2+C_1}$$

Onde μ_x é a média do vetor x; C_1 é um número pequeno, muito próximo de zero. Esta função calcula a diferença dos brilhos. Dá um número entre 0 e 1.

$$c(x,y)=\frac{2\sigma_x\sigma_y+C_2}{\sigma_x^2+\sigma_y^2+C_2}$$

Onde σ_x é o desvio-padrão do vetor x; C_2 é um número pequeno, muito próximo de zero. Esta função calcula a diferença dos contrastes. Dá um número entre 0 e 1.

$$s(x,y)=\frac{\sigma_{xy}+C_3}{\sigma_x\sigma_y+C_3}$$

$$[\sigma_{xy}=\frac{1}{N-1}\sum_{i=1}^{N}(x_i-\mu_x)(y_i-\mu_y) \quad \text{ou seria} \quad \sigma_{xy}=\frac{1}{N}\sum_{i=1}^{N}(x_i-\mu_x)(y_i-\mu_y)\,?]$$

A função $s(x,y)$ calcula a correlação normalizada entre os vetores x e y. Dá um número entre -1 e +1.

Nota: Correlação cruzada normalizada também pode ser calculada como: $r_{xy}=\frac{\tilde{x}\tilde{y}}{\|\tilde{x}\|\|\tilde{y}\|}$

onde $\tilde{x}$ é o vetor x com correção de média (subtrai a média de todos os elementos).

Exemplo: x=[1 3 5 2] y=[2 4 6 3] s(x,y)=1	Exemplo: x=[1 3 5 2] y=[3 7 11 5] s(x,y)=1

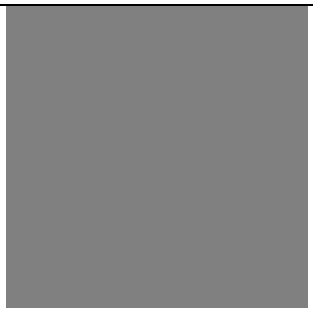
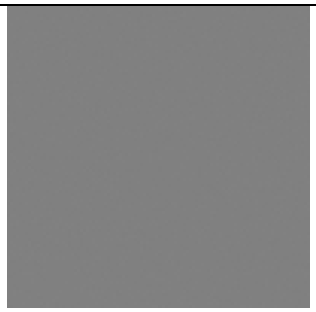
$$SSIM(x,y)=[l(x,y)]^{\alpha}[c(x,y)]^{\beta}[s(x,y)]^{\gamma}$$

O artigo original usa $\alpha=\beta=\gamma=1$.

SSIM entre as imagens X e Y é a média aritmética do SSIM de todos os pixels.

"Defeito" do SSIM: Consider uma região com nível de cinza quase constante e algum ruído. Nessa região, o ruído fará com que as duas imagens não estejam correlacionadas e consequentemente SSIM será baixa. Porém, a intuição diz que SSIM deveria ser alta.

Exemplo:

 cinzaa.pgm	 cinzab.pgm

As duas imagens são perceptualmente idênticas.

```
#include <cekeikon.h>
int main() {
  Mat_<GRY> a(100,100,128);
  Mat_<GRY> b(100,100,128);
  srand(7);
  for (unsigned i=0; i<b.total(); i++) {
    int r=rand()%10;
    if (r==0) b(i)--;
    if (r==1) b(i)++;
  }
  imp(a,"cinzaa.pgm");
  imp(b,"cinzab.pgm");
}
```

Porém, SSIM diz que as duas imagens são completamente diferentes:

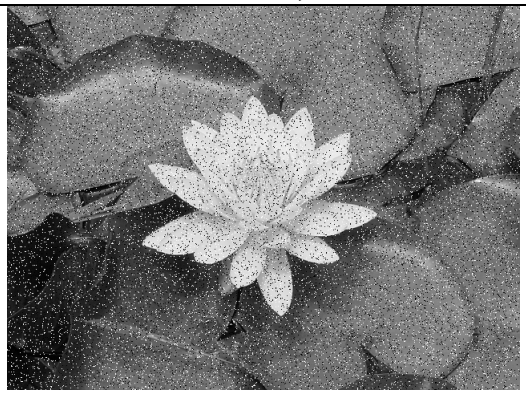
```
>img d_ssimg cinzaa.pgm cinzab.pgm 8 8
Media de SSIM=8.52061e-010
```

Programa usado para calcular s(x,y):

```
//struct1.cpp - grad2016
#include <cekeikon.h>
int main()
{ Mat_<FLT> x=(Mat_<FLT>(1,4) << 1, 3, 5, 2);
  Mat_<FLT> y=(Mat_<FLT>(1,4) << 2, 4, 6, 3);

  double mediaX,desvioX;
  mediaDesvio(x, mediaX, desvioX);
  xprint(mediaX);
  xprint(desvioX);

  double mediaY,desvioY;
  mediaDesvio(y, mediaY, desvioY);
  xprint(mediaY);
  xprint(desvioY);

  double desvioXY=0.0;
  for (unsigned i=0; i<x.total(); i++)
    desvioXY = desvioXY+(x(i)-mediaX)*(y(i)-mediaY);
  desvioXY=desvioXY/(x.total());
  xprint(desvioXY);
  double C3=1e-10;
  double estr=(desvioXY+C3)/(desvioX*desvioY+C3);
  xprint(estr);
}
```

```
//struct2.cpp - grad2016
#include <cekeikon.h>
int main()
{ Mat_<FLT> x=(Mat_<FLT>(1,4) << 1, 3, 5, 2);
  Mat_<FLT> y=(Mat_<FLT>(1,4) << 2, 4, 6, 3);
  x=dcReject(x);
  y=dcReject(y);
  float estr=x.dot(y)/(norm(x)*norm(y));
  xprint(estr);
}
```

```
//d_ssimg.cpp
#include <proeikon>

int main(int argc, char** argv)
{ if (argc!=5 && argc!=8) {
    printf("D_SSIMG: Diferenca de 2 IMGGRY pela structural similarity\n");
    printf("D_SSIMG ent1.pgm ent2.pgm nl nc [alpha beta gama]\n");
    printf("  nl,nc: tamanho da janela\n");
    printf("  alpha=brilho beta=contraste gama=estrutura(correlacao) [entre 0 e 1]\n");
    printf("  Default: alpha=beta=gama=1\n");
    erro("Erro: Numero de argumentos invalido");
  }

  IMGGRY a1; le(a1,argv[1]); a1.backg()=0;
  IMGGRY a2; le(a2,argv[2]); a2.backg()=0;
  if (a1.nl()!=a2.nl() || a1.nc()!=a2.nc()) erro("Erro: Dimensoes diferentes");

  int nl,nc;
  if (sscanf(argv[3],"%d",&nl)!=1) erro("Erro nl");
  if (sscanf(argv[4],"%d",&nc)!=1) erro("Erro nc");
  IMGGRY jan(nl,nc);

  double alpha=1.0;
  double beta=1.0;
  double gama=1.0;
  if (argc==8) {
    if (sscanf(argv[5],"%lf",&alpha)!=1) erro("Erro alpha");
    if (sscanf(argv[6],"%lf",&beta)!=1)  erro("Erro beta");
    if (sscanf(argv[7],"%lf",&gama)!=1)  erro("Erro gama");
  }

  VETOR<double> Y(nl*nc);
  VETOR<double> X(nl*nc);
  double soma=0.0;
  for (int l=0; l<a1.nl()-jan.nl()+1; l++)
    for (int c=0; c<a1.nc()-jan.nc()+1; c++) {
      int i=0;
      for (int l2=0; l2<jan.nl(); l2++)
        for (int c2=0; c2<jan.nc(); c2++) {
          X(i)=a1(l+l2,c+c2);
          Y(i)=a2(l+l2,c+c2);
          i++;
        }
      soma+=ssimgry(X,Y,alpha,beta,gama);
    }
  printf("Media de SSIM=%g\n",soma/((a1.nl()-jan.nl()+1)*(a1.nc()-jan.nc()+1)));
}
```

 flor.tga MAE=0% RMSE=0% PSNR=∝dB SSIM=1,00	 flor01.jpg (qualidade 1) MAE=4,0% RMSE=5,5% PSNR=25,3dB SSIM=0,19
 flor-bril20.tga (brilho +20) MAE=5,5% RMSE=5,5% PSNR=25,2dB SSIM=0,98	 flor-cont25.tga (contraste +25) MAE=5,0% RMSE=6,4% PSNR=23,9 dB SSIM=0,86
 flor-gauss9.tga (gauss 9 pixels) MAE=4,4% RMSE=7,0% PSNR=23,1dB SSIM=0,21	 flor-sp10.tga (sal-pimenta 1 cada 10) MAE=5,0% RMSE=17,2% PSNR=15,3dB SSIM=0,11

Color Strutural Similarity:

In CIELAB space, the lightness L* varies from 0 to 100. The range of chromaticity components a*b* depends on the original color space of the image. If the original color space is RGB, one can assume the range -100 to +100.

Let $\mathbf{x} = \{x_1, x_2, ..., x_n\}$ and $\mathbf{y} = \{y_1, y_2, ..., y_n\}$ be two vectors of colors. Each component x_i or y_i is composed by a set of tristimulus values L*, a* and b* denoted, respectively, as x_{iL}, x_{ia}, x_{ib} and y_{iL}, y_{ia}, y_{ib}. The similarity functions *l*, *c*, and *s* are computed on the component L* as in the grayscale case. For similarity of chromaticity *h*, we use the Euclidean distance of components a* and b*, because it is typically used as the distance measure in CIELAB color space [5, 17]:

$$h(\mathbf{x},\mathbf{y}) = 1 - \frac{\sum_{i=1}^{n} \sqrt{(x_{ia} - y_{ia})^2 + (x_{ib} - y_{ib})^2}}{200 \cdot \sqrt{2} \cdot n}$$

where 200 is the greatest possible difference between the components a* and b*. To obtain the similarity measure, the distance is subtracted from one. We define the color structural similarity as:

$$\mathrm{CSSIM}(\mathbf{x},\mathbf{y}) = [l(\mathbf{x},\mathbf{y})]^{\alpha} [c(\mathbf{x},\mathbf{y})]^{\beta} [s(\mathbf{x},\mathbf{y})]^{\gamma} [h(\mathbf{x},\mathbf{y})]^{\delta}$$

We use weighted geometric mean (instead of weighted arithmetic mean between SSIM and *h*) because either complete chromaticity dissimilarity or complete lightness dissimilarity represents a complete dissimilarity of the two patches. In the improved color Ciratefi, we use CSSIM as the similarity measure in all the three filters.

Dizer que existe como avaliar a qualidade sem imagem de referência.