Total variation regularization with Chambolle Algorithm:

[Rudin92] propôs filtrar imagem minimizando a variação total (total variation). A suposição é que a imagem sem ruído é constante por regiões.

[Chambolle04] propôs um algoritmo (entre muitos) para essa minimização.

Uma descrição didática desse algoritmo para sinais 1-D (com código Matlab) encontra-se em [Selesnick09].

Transparências didáticas sobre compressive sensing em [Tao].

A tese [Ozgen2012] é uma boa tese sobre compressive sensing em tomografia.

==========

Minimização de TV é muito usada em compressive sampling ou compressed sensing.

A gradiente de uma imagem discreta *f* é [Chambolle04]:

$$\nabla f(x,y) = \left[\frac{\partial f(x,y)}{\partial x}, \frac{\partial f(x,y)}{\partial y} \right]$$

com:

$$\frac{\partial f(x,y)}{\partial x} = \begin{cases} f(x+1,y) - f(x,y) \text{ para } x < N \\ 0 \text{ para } x = N \end{cases}$$

$$\frac{\partial f(x,y)}{\partial y} = \begin{cases} f(x,y+1) - f(x,y) \text{ para } y < N \\ 0 \text{ para } y = N \end{cases}$$

```
IMGCPX grad(IMGDBL a)
{ IMGCPX d( a.nl(), a.nc(), CPX(0.0,0.0) );
  for (int x=0; x<a.nx()-1; x++)
    for (int y=0; y<a.ny()-1; y++)
      d.atN(x,y) = CPX( a.atN(x+1,y)-a.atN(x,y), a.atN(x,y+1)-a.atN(x,y) );
  return d;
}
```

A divergente (divergence) de uma campo vetorial diferenciável *p* é [Wikipedia]:

$$\text{div } p(x,y) = \frac{\partial p_x(x,y)}{\partial x} + \frac{\partial p_y(x,y)}{\partial y}$$

Se *p* for campo vetorial discreta [Chambolle04]:

$$\text{div } p(x,y) = \begin{bmatrix} p_x(x,y) - p_x(x-1,y), \text{ se } 1 < x < N \\ p_x(x,y), \text{ se } x = 1 \\ -p_x(x-1,y), \text{ se } x = N \end{bmatrix} + \begin{bmatrix} p_y(x,y) - p_y(x,y-1), \text{ se } 1 < y < N \\ p_y(x,y), \text{ se } y = 1 \\ -p_y(x,y-1), \text{ se } y = N \end{bmatrix}$$

```
IMGDBL div(IMGCPX a)
{ IMGDBL d(a.nl(),a.nc());
  for (int x=0; x<a.nx(); x++)
    for (int y=0; y<a.ny(); y++) {
      double tx;
      if      (x==0) tx=real(a.atN(x,y));
      else if (x==a.nx()-1) tx=-real(a.atN(x-1,y));
      else tx=real(a.atN(x,y))-real(a.atN(x-1,y));

      double ty;
      if      (y==0) ty=imag(a.atN(x,y));
      else if (y==a.ny()-1) ty=-imag(a.atN(x,y-1));
      else ty=imag(a.atN(x,y))-imag(a.atN(x,y-1));

      d.atN(x,y) = tx+ty;
    }
  return d;
}
```

A variação total (total variation) de uma imagem suave *f* é definida [Peyre10]:

$$J(f) = \int \|\nabla f(x)\| dx$$

A variação total de uma imagem discreta *f* é definida [Chambolle04]:

$$J(f) = \sum_{1 \le x,y \le N} \|\nabla f(x,y)\|$$

```
double somatoria(IMGDBL a)
{ double d=0.0;
  for (int i=0; i<a.n(); i++) d+=a.atf(i);
  return d;
}

double J(IMGDBL f)
{ IMGDBL m=abs(grad(f));
  return somatoria(m);
}
```

Suponha que uma imagem f_o foi contaminada pelo ruído *w* gerando a imagem $y=f_o+w$. Queremos atenuar ruído da observação *y*. O objetivo é achar *f* que minimize [Peyre10]:

$$\min_f E(f) = \frac{1}{2}\|y-f\|^2 + \lambda J(f)$$

Isto é feito iterativamente com campo vetorial *G*.

$$G^{(l+1)} = \text{Proj}\left(G^{(l)} - \tau\nabla(\text{div}(G^{(l)}) - y/\lambda\right)$$

O valor inicial de *G* é zero.

Proj (clip) significa saturar os valores em -1 e +1.

```
IMGCPX clip(IMGCPX a)
{ IMGCPX d=a;
  for (int i=0; i<a.n(); i++) {
    double re=real(a(i)); double im=imag(a(i));
    if (re<-1.0) re=-1.0;
    if (re>1.0) re=1.0;
    if (im<-1.0) im=-1.0;
    if (im>1.0) im=1.0;
    d(i)=CPX(re,im);
  }
  return d;
}
```

Na teoria, deve-se ter $\tau \leq 1/8$ para obter a convergência. [Chambolle04] nota que na prática resultado melhor foi obtido com $\tau=1/4$. Nos meus testes, usando $\tau=1/4$ e λ adaptativo (tv3), o algoritmo ou oscila ou diverge. Algoritmo com λ fixo parece funcionar bem com $\tau=1/4$.

A imagem restaurada é y-lambda*div(G).

```
IMGDBL tv1(IMGDBL y, double lambda, int nit)
{ double tau = 1.0/8.0;
  IMGCPX G(y.nl(),y.nc(),0.0);
  IMGCPX dG;
  for (int i=0; i<nit; i++) {
    dG = grad( div(G) - (1.0/lambda)*y );
    G = clip( G + tau*dG );
  }
  return y - lambda*div(G);
}
```

Se quiser obter as imagens restauradas intermediárias da iteração:

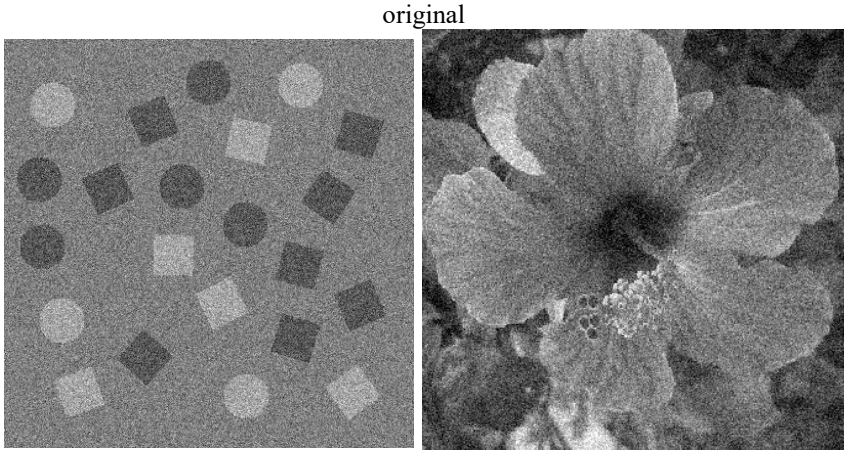
```
IMGDBL tv2(IMGDBL y, double lambda, int nit)
{ double tau = 1.0/8.0;
  IMGCPX G(y.nl(),y.nc(),0.0);
  IMGCPX dG;
  IMGDBL f;
  for (int i=0; i<nit; i++) {
    f = y - lambda*div(G);
    dG = grad( (-1.0/lambda)*f );
    G = clip( G + tau*dG );
  }
  f = y - lambda*div(G); // pode eliminar
  return f;
}
```

Chamando:

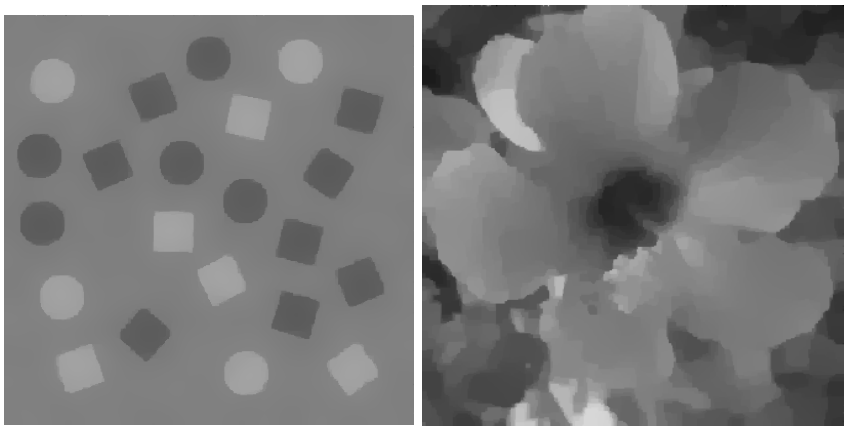
```
IMGDBL y=f0;
double sigma=0.1;
for (int l=0; l<f0.nl(); l++)
  for (int c=0; c<f0.nc(); c++)
    y(l,c)=f0(l,c)+sigma*mygauss();
IMGDBL a2=tv2(y, 0.2, 500);
```

Obtém:

original



ruidosa



restaurada com lambda fixo

É possível calcular automaticamente λ a partir do desvio-padrão σ do ruído *w*.

$$\lambda^{(l+1)} = \lambda^{(l)} \frac{N\sigma}{\|f - y\|}$$

onde *N* é o lado da imagem. Se imagem não for quadrada, $N = \sqrt{m}$ onde *m* é o número de pixels da imagem. O valor inicial de λ pode ser qualquer número positivo.

```
IMGDBL tv3(IMGDBL y, double sigma, int nit)
{ double tau = 1.0/8.0;
  IMGCPX G(y.nl(),y.nc(),0.0);
  IMGCPX dG;
  IMGDBL f;
  double lambda=0.2;
  for (int i=0; i<nit; i++) {
    f = y - lambda*div(G);
    dG = grad( (-1.0/lambda)*f );
    G = clip( G + tau*dG );
    double d=comprimento(f-y); // Na 1a vez, d=0
    if (d>epsilon) lambda = lambda * sigma * sqrt(double(f.n())) / d;
  }
  f = y - lambda*div(G); // pode eliminar
  return f;
}
```

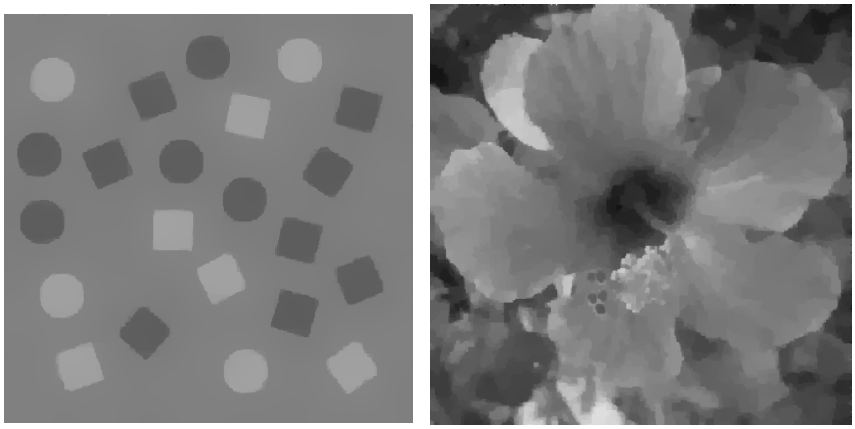


Imagem restaurada com cálculo automático de λ a partir de σ.

Referências bibliográficas:

[Chambolle04] A. Chambolle, "An Algorithm for Total Variation Minimization and Applications," *Journal of Mathematical Imaging and Vision*, vol. 20, 2004, pp. 89-97.

[Peyre10] "Total Variation Regularization with Chambolle Algorihtm," http://www.ceremade.dauphine.fr/~peyre/numerical-tour/tours/denoisingadv_3_chambollealgo/#16

[Rudin92] L. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Physica D: Nonlinear Phenomena*, vol. 60, 1992, pp. 259-268.

[Selesnick09] I. Selesnick, I. Bayram, "Total Variation Filtering," vol. 1, 2009, pp. 1-8. http://passthru.cnx.org/plone/content/m31292/1.1/

[Tao] http://www.math.hkbu.edu.hk/~ttang/UsefulCollections/compressed-sensing1.pdf

[Ozgen2012] Özgen, C. (2012). *COMPRESSED SENSING BASED COMPUTERIZED TOMOGRAPHY IMAGING* (Doctoral dissertation, MIDDLE EAST TECHNICAL UNIVERSITY). https://etd.lib.metu.edu.tr/upload/12614170/index.pdf

Atenuação de ruído gaussiano (C:\haepi\filtro\tv\tvart):

Vamos tentar achar o filtro mais adequado para filtrar uma imagem constante por regiões contaminada por ruído gaussiano.

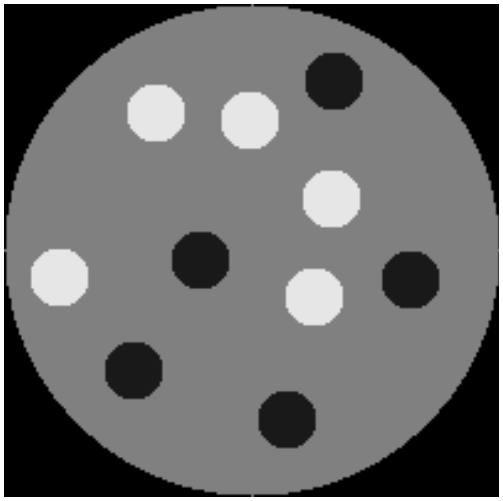
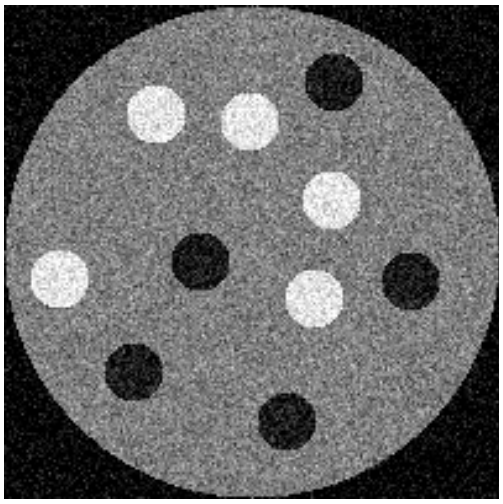


imagem original “fantom.tga”

Vamos colocar ruído gaussiano com desvio-padrão 0.1:

```
img ruidogag fantom.tga ruido10.tga 0.1 7
```

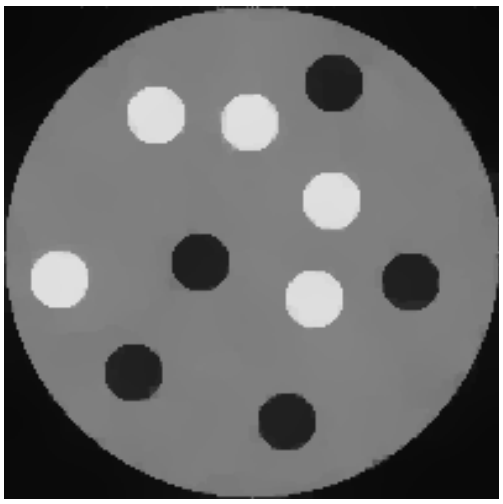


ruido10.tga

```
MAE (max=100%)...................:               6.99%
RMSE (max=100%)..................:               9.29%
PSNR considering highest=255.....:            20.6416 dB
```

1) TV com sigma:

```
img tv ruido10.tga ruido10tv.tga sigma     200 0.09
```

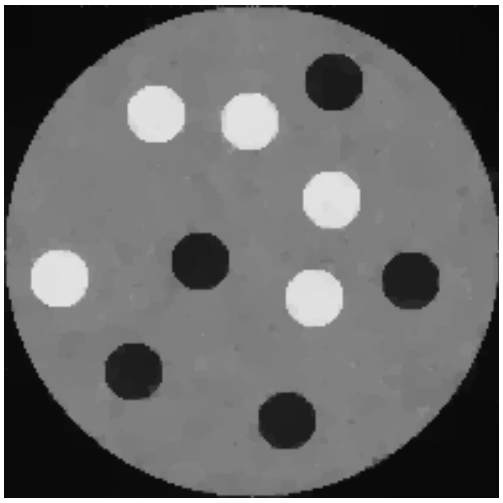


```
MAE (max=100%)..................:               1.99%
RMSE (max=100%).................:               2.97%
PSNR considering highest=255.....:             30.5402 dB
```

Interessante que o melhor resultado não foi obtido com sigma=0.1, mas com sigma=0.09.

2) TV com lambda:

```
img tv ruido10.tga ruido10la.tga lambda    200 0.075
```

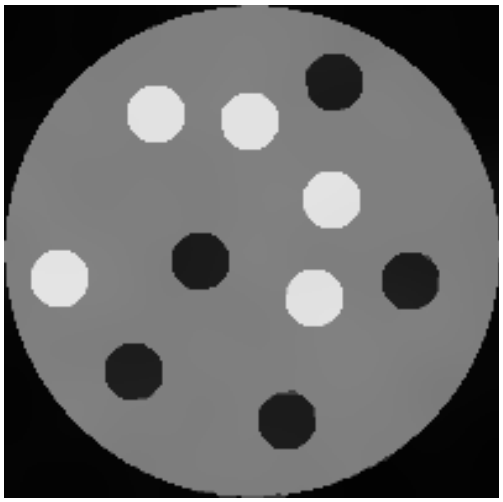


```
MAE (max=100%)..................:               1.93%
RMSE (max=100%).................:               2.84%
PSNR considering highest=255.....:             30.9403 dB
```

O resultado é quase igual ao TV com sigma. O problema é que tem que chutar lambda sem saber o que representa. A qualidade visual é pior.

3) Difusão anisotrópica intercalado com poucas medianas:

```
img ad ruido10.tga ruido10ad.tga 200 0.055 t 1 3 50
(AD ent.tga sai.tga maxt sigma [metodo lambda eemediana cdqto])
```

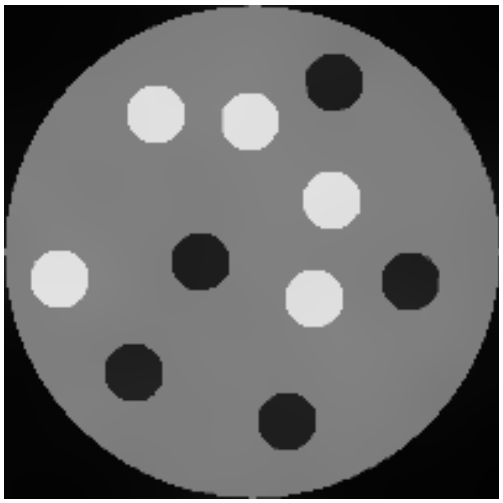


```
MAE (max=100%)..................:              0.96%
RMSE (max=100%).................:              2.24%
PSNR considering highest=255.....:            33.0011 dB
```

A qualidade é melhor do que TV, tanto medido quando perceptual.

4) Mediana + TV + AD:

```
img tv ruido10.tga ruido10me.tga mla       200 0.045 3 200 0.055
(TV ent.pgm sai.pgm metodo nit param [tee maxt sigma])
```



```
MAE (max=100%)..................:              1.21%
RMSE (max=100%).................:              2.14%
PSNR considering highest=255.....:            33.3900 dB
```

Praticamente idêntico ao item (3). Não há vantagem em usar TV

Conclusão: AD é melhor que TV para eliminar ruído gaussiano.

Atenuação de ruído gaussiano + sal-pimenta (C:\haepi\filtro\tv\tvart):

Vamos tentar achar o filtro mais adequado para filtrar uma imagem constante por regiões contaminada por ruído gaussiano + ruído sal e pimenta.

Este tipo de contaminação é comum em diferentes áreas. Por exemplo, na tomografia.

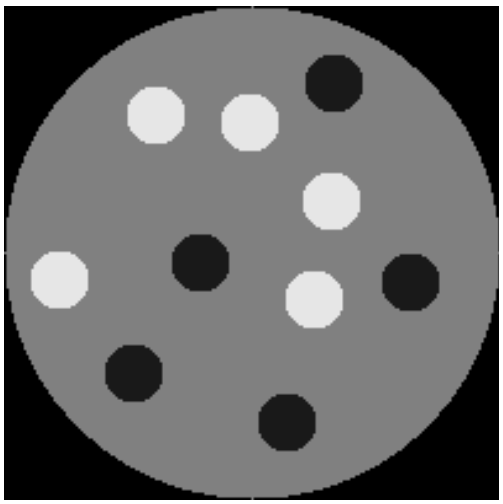
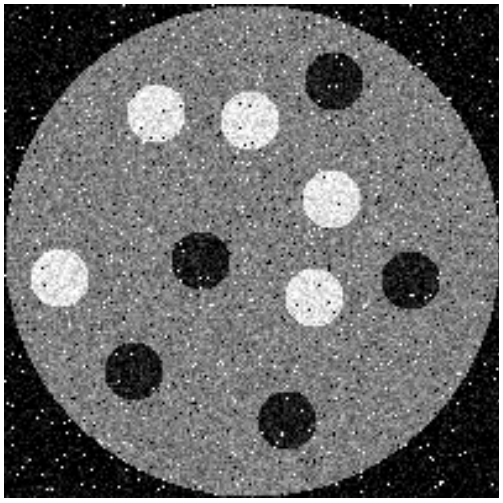


imagem original “fantom.tga”

Vamos colocar ruído gaussiano com desvio-padrão 0.1 e sal-e-pimenta estragando 1 em 40 pixels:

```
img ruidogag fantom.tga  ruigs14.tga 0.1 8
img ruidospg ruigs14.tga ruigs14.tga 40 8
```

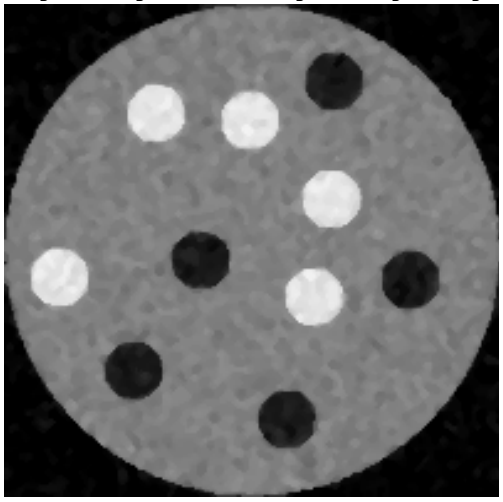


ruigs14.tga

```
MAE (max=100%)..................:              8.04%
RMSE (max=100%).................:             12.81%
PSNR considering highest=255.....:           17.8497 dB
```

1) Usar 4 vezes filtro mediana 3x3:

```
img medianag ruigs14.tga ruigs14mg.tga 3 3
img medianag ruigs14mg.tga ruigs14mg.tga 3 3
img medianag ruigs14mg.tga ruigs14mg.tga 3 3
img medianag ruigs14mg.tga ruigs14mg.tga 3 3
img distg fantom.tga ruigs14mg.tga
```

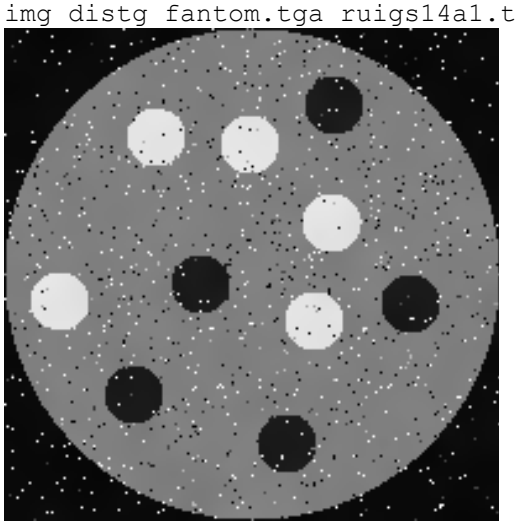


```
MAE (max=100%).....:                2.24%
RMSE (max=100%).....:               3.71%
PSNR considering highest=255.....:  28.6188 dB
```

Elimina bem ruído sal-pimenta. Não consegue eliminar ruído gaussiano.

2) Difusão anisotrópica:

```
img ad ruigs14.tga ruigs14a1.tga 50 0.1 t 1 1
img distg fantom.tga ruigs14a1.tga
```

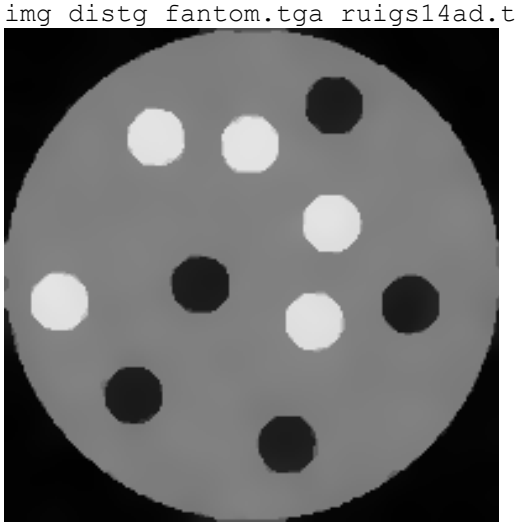


```
MAE (max=100%)..................:              2.73%
RMSE (max=100%).................:              9.55%
PSNR considering highest=255.....:           20.4032 dB
```

Elimina bem ruído gaussiano. Não elimina ruído sal-pimenta.

3) Difusão anisotrópica intercalado com filtro mediano:

```
img ad ruigs14.tga ruigs14ad.tga 50 0.03 t 1 3
img distg fantom.tga ruigs14ad.tga
```

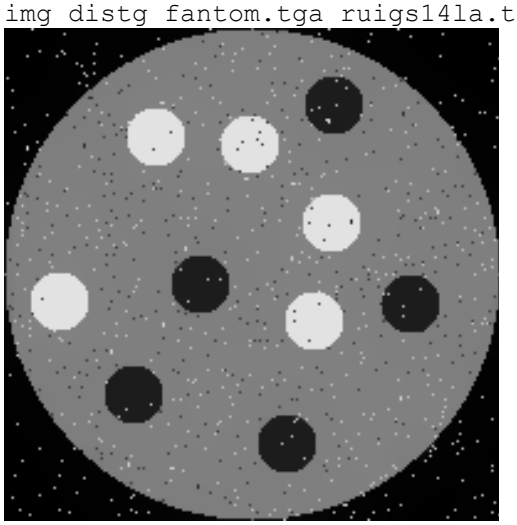


```
MAE (max=100%)..................:              1.28%
RMSE (max=100%).................:              3.19%
PSNR considering highest=255.....:           29.9305 dB
```

Elimina bem ruído gaussiano e ruído sal-pimenta. PSNR é o mais alto até agora.

4) TV com lambda:

```
img tv ruigs14.tga ruigs14la.tga lambda 200 0.2
img distg fantom.tga ruigs14la.tga
```

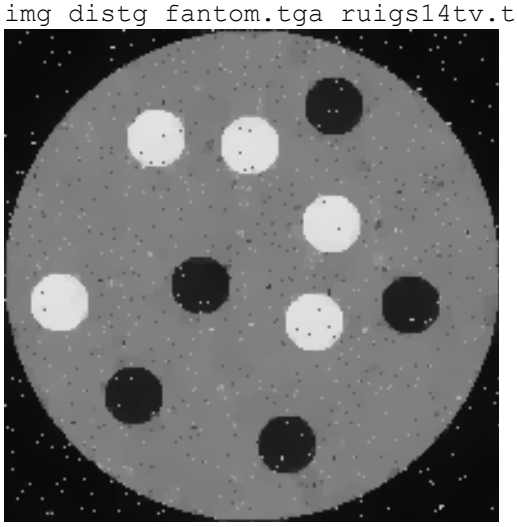


```
MAE (max=100%)..................:               2.97%
RMSE (max=100%).................:               4.61%
PSNR considering highest=255.....:            26.7291 dB
```

Elimina bem ruído gaussiano. Não consegue eliminar ruído sal-pimenta. O parâmetro lambda tem que ser ajustado sem ter ideia do que representa.

5) TV com sigma:

```
img tv ruigs14.tga ruigs14tv.tga sigma  200 0.1
img distg fantom.tga ruigs14tv.tga
```

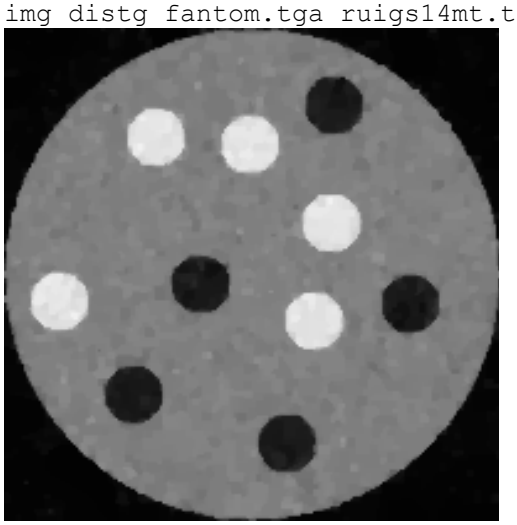


```
MAE (max=100%)..................:               2.62%
RMSE (max=100%).................:               5.60%
PSNR considering highest=255.....:            25.0395 dB
```

Elimina bem ruído gaussiano. Não consegue eliminar ruído sal-pimenta. O parâmetro sigma é fácil de ajustar, se conhecer o desvio-padrão do ruído.

7) mediana + TV com lambda:

```
img tv ruigs14.tga ruigs14mt.tga mla    200 0.03 3 0 0.0
img distg fantom.tga ruigs14mt.tga
```

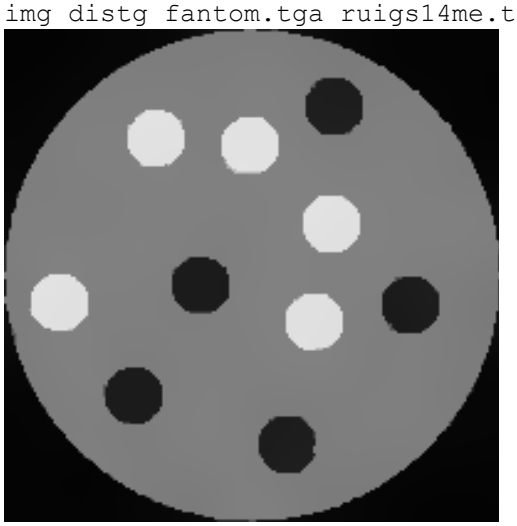


```
MAE (max=100%)..................:              1.91%
RMSE (max=100%).................:              3.29%
PSNR considering highest=255.....:            29.6473 dB
```

Executando filtro mediana antes de TV, consegue eliminar ruído sal-pimenta e gaussiana. PSNR ficou bem alto. Mas há variação de nível de cinza de fundo.

8) mediana + TV com lambda + AD

```
img tv ruigs14.tga ruigs14me.tga mla    200 0.03 3 200 0.05
img distg fantom.tga ruigs14me.tga
```

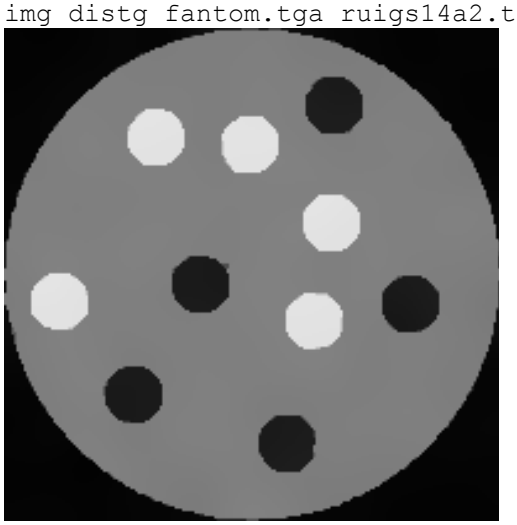


```
MAE (max=100%)..................:              1.16%
RMSE (max=100%).................:              2.59%
PSNR considering highest=255.....:            31.7453 dB
```

Executando filtro mediana+TV+AD, consegue eliminar ruído sal-pimenta e gaussiana. Obteve o maior PSNR.

9) Difusão anisotrópica intercalado com poucas medianas

```
img ad ruigs14.tga ruigs14a2.tga                150 0.05 t 1 3 80
img distg fantom.tga ruigs14a2.tga
```



```
MAE (max=100%).....................:                0.98%
RMSE (max=100%)....................:                2.49%
PSNR considering highest=255.....:              32.0783 dB
```

Elimina bem ruído gaussiano e ruído sal-pimenta. PSNR é o mais alto até agora. Não apresenta ruídos.

Conclusão: AD intercalado com medianas é a melhor solução. Não valeu a pena usar TV.