```
// Programa janelas.cpp
// Prof. Dr. Hae Yong Kim
// Aluna de mestrado Luciana Guidon Coelho
// Escola Politécnica da Universidade de São Paulo
// 2012

// Programa para escolher a melhor técnica de estimação de janela de fundo de céu.

#include <proeikon>

double mae(IMGDBL a, IMGDBL b)
{ if (a.nl()!=b.nl() || a.nc()!=b.nc())
    erro("Erro mae: dimensoes diferentes");
  double soma=0;
  for (int l=0; l<a.nl(); l++)
    for (int c=0; c<a.nc(); c++) {
      soma += abs(a(l,c)-b(l,c));
    }
  return soma/a.n();
}

double rmse(IMGDBL a, IMGDBL b)
{ if (a.nl()!=b.nl() || a.nc()!=b.nc())
    erro("Erro rmse: dimensoes diferentes");
  double soma=0;
  for (int l=0; l<a.nl(); l++)
    for (int c=0; c<a.nc(); c++) {
      soma += elev2(a(l,c)-b(l,c));
    }
  return sqrt(soma/a.n());
}

void distd(int argc, char** argv)
{ if (argc!=3) {
    printf("DistD: Calcula diferenca entre duas IMGDBL\n");
    printf("DistD a.img b.img\n");
    erro("Erro: Numero de argumentos invalido");
  }
  IMGDBL a,b;
  le(a,argv[1]);
  le(b,argv[2]);
  printf("::Sem correcao media: MAE=%f RMSE=%f\n",mae(a,b),rmse(a,b));
  a = -media(a)+a;
  b = -media(b)+b;
  printf("::Com correcao media: MAE=%f RMSE=%f\n",mae(a,b),rmse(a,b));
}

void mediad(int argc, char** argv)
{ if (argc<3) {
    printf("MediaD sai.img ent1.img ent2.img...\n");
    erro("Erro: Numero de argumentos invalido");
  }
  IMGDBL ent; le(ent,argv[2]);
  IMGDBL sai=ent;
  for (int i=3; i<argc; i++) {
    le(ent,argv[i]);
    sai = sai+ent;
  }
  sai = (1.0/(argc-2.0)) * sai;
```

```
  imp(sai,argv[1]);
}

void marcaim(int argc, char** argv)
{ if (argc!=6) {
    printf("Marca imagem com estrelas (r), janelas32 (g) e janelas512 (b)\n");
    printf("MarcaIm ent.img estrelas.txt jan32.txt jan512.txt sai.ppm\n");
    printf("  IMGDBL e' dividida por 2000\n");
    erro("Erro: Numero de argumentos invalido");
  }
  IMGDBL ent; le(ent,argv[1]);
  IMGCOR sai(ent.nl(),ent.nc());
  for (int i=0; i<ent.n(); i++) {
    double temp=255.0*ent(i)/2000;
    //if (temp>255) temp=255;
    //if (temp<0) temp=0;
    BYTE b=double2G(temp);
    sai(i) = COR(b,b,b);
  }

  IMGDBL _est; le(_est,argv[2]);
  VETOR<PONTOI2> est(_est.nl());
  for (int i=0; i<est.n(); i++)
    est(i)=PONTOI2(arredonda(_est(i,0)),arredonda(_est(i,1)));
  int delta=3;
  for (int i=0; i<est.n(); i++) {
    int xi=est(i).x(); int yi=est(i).y();
    for (int x=-delta; x<=delta; x++) {
      sai.atN(xi+x, yi+delta)=COR(255,0,0);
      sai.atN(xi+x, yi-delta)=COR(255,0,0);
    }
    for (int y=-delta; y<=delta; y++) {
      sai.atN(xi+delta, yi+y)=COR(255,0,0);
      sai.atN(xi-delta, yi+y)=COR(255,0,0);
    }
  }

  IMGDBL _j32; le(_j32,argv[3]);
  VETOR<PONTOI2> j32(_j32.nl());
  for (int i=0; i<j32.n(); i++)
    j32(i)=PONTOI2(arredonda(_j32(i,0)),arredonda(_j32(i,1)));
  for (int i=0; i<j32.n(); i++) {
    int xi=j32(i).x(); int yi=j32(i).y();
    for (int x=0; x<10; x++) {
      sai.atN(xi+x,yi)=COR(0,255,0);
      sai.atN(xi+x,yi+9)=COR(0,255,0);
    }
    for (int y=0; y<10; y++) {
      sai.atN(xi,  yi+y)=COR(0,255,0);
      sai.atN(xi+9,yi+y)=COR(0,255,0);
    }
  }

  IMGDBL _j512; le(_j512,argv[4]);
  VETOR<PONTOI2> j512(_j512.nl());
  for (int i=0; i<j512.n(); i++)
    j512(i)=PONTOI2(arredonda(_j512(i,0)),arredonda(_j512(i,1)));
  for (int i=0; i<j512.n(); i++) {
    int xi=j512(i).x(); int yi=j512(i).y();
```

```
    for (int x=0; x<10; x++) {
      sai.atN(xi+x,yi)=COR(0,0,255);
      sai.atN(xi+x,yi+9)=COR(0,0,255);
    }
    for (int y=0; y<10; y++) {
      sai.atN(xi,  yi+y)=COR(0,0,255);
      sai.atN(xi+9,yi+y)=COR(0,0,255);
    }
  }

  imp(sai,argv[5]);
}

void pegacol(int argc, char** argv)
{ if (argc!=4) {
    printf("PegaCol: Pega uma coluna num arquivo TXT para tracar grafico\n");
    printf("PegaCol ent.img sai.txt coluna\n");
    erro("Erro: Numero de argumentos invalido");
  }
  IMGDBL a; le(a,argv[1]);
  int coluna; ConvArg(coluna,argv[3]);
  VETOR<double> v(a.nl());
  for (int i=0; i<a.nl(); i++)
    v(i)=a(i,coluna);
  imp(v,argv[2]);
}

void minmax(int argc, char** argv)
{ if (argc!=2) {
    printf("MinMax: Acha colunas cujas medianas sao minimo e maximo\n");
    printf("minmax ent.img\n");
    erro("Numero de argumentos invalido");
  }
  IMGDBL a; le(a,argv[1]);
  VETOR<double> m(a.nc());
  VETOR<double> w(a.nl());
  for (int c=0; c<a.nc(); c++) {
    for (int l=0; l<a.nl(); l++) w(l)=a(l,c);
    m(c)=mediana(w);
  }
  int minimo=argmin(m);
  int maximo=argmax(m);
  printf("Coluna minima=%d mediana=%f   maxima=%d mediana=%f\n",minimo,m(minimo),maximo,m(
  maximo));
}

void txt2img(int argc, char** argv)
{ if (argc!=3) {
    printf("txt2img: Converte TXT para IMG para acelerar a leitura\n");
    printf("txt2img ent.txt sai.img\n");
    erro("Erro: Numero de argumentos invalido");
  }
  IMGDBL a; le(a,argv[1]);
  printf("Li %s de tamanho %d %d\n",argv[1],a.nl(),a.nc());
  imp(a,argv[2]);
}

void reesmed(int argc, char** argv)
{ if (argc!=3) {
```

```
    printf("ReesMed: Reestima os dados pela mediana linha/coluna e comb lin\n");
    printf("  Nao se descarta nenhuma linha na reestimacao\n");
    printf("ReesMed ent.img sai.img\n");
    erro("Erro: Numero de argumentos invalido");
  }
  IMGDBL a; le(a,argv[1]);

  VETOR<double> medlin(a.nl());
  VETOR<double> temp1(a.nc());
  for (int l=0; l<a.nl(); l++) {
    for (int c=0; c<a.nc(); c++) temp1(c)=a(l,c);
    medlin(l)=mediana(temp1);
  }

  VETOR<double> medcol(a.nc());
  VETOR<double> temp2(a.nl());
  for (int c=0; c<a.nc(); c++) {
    for (int l=0; l<a.nl(); l++)
      temp2(l)=a(l,c);
    medcol(c)=mediana(temp2);
  }

  printf("Linha a linha:   Media=%f desvio=%f\n",media(medlin),desvio(medlin));
  printf("Coluna a coluna: Media=%f desvio=%f\n",media(medcol),desvio(medcol));

  for (double alpha=0.0; alpha<=1.0; alpha+=0.1) {
    double beta=1.0-alpha;
    //double medianatotal=alpha*mediana(medlin)+beta*mediana(medcol);
    IMGDBL b(a.nl(),a.nc());
    for (int l=0; l<a.nl(); l++)
      for (int c=0; c<a.nc(); c++) {
        //b(l,c)=medianatotal+(medlin(l)-medianatotal)+(medcol(c)-medianatotal))/2;
        b(l,c)=alpha*medlin(l)+beta*medcol(c);
      }
    imp(b,argv[2]);
    printf("Alpha=%f beta=%f MAE=%f RMSE=%f\n",alpha,beta,mae(a,b),rmse(a,b));
  }
}

void reesmean(int argc, char** argv)
{ if (argc!=3) {
    printf("ReesMean: Reestima os dados pela media linha/coluna e comb lin\n");
    printf("  Nao se descarta nenhuma linha na reestimacao\n");
    printf("ReesMean ent.img sai.img\n");
    erro("Erro: Numero de argumentos invalido");
  }
  IMGDBL a; le(a,argv[1]);

  VETOR<double> medlin(a.nl());
  VETOR<double> temp1(a.nc());
  for (int l=0; l<a.nl(); l++) {
    for (int c=0; c<a.nc(); c++) temp1(c)=a(l,c);
    medlin(l)=media(temp1);
  }

  VETOR<double> medcol(a.nc());
  VETOR<double> temp2(a.nl());
  for (int c=0; c<a.nc(); c++) {
    for (int l=0; l<a.nl(); l++)
```

```
      temp2(l)=a(l,c);
    medcol(c)=media(temp2);
  }

  printf("Linha a linha:   Media=%f desvio=%f\n",media(medlin),desvio(medlin));
  printf("Coluna a coluna: Media=%f desvio=%f\n",media(medcol),desvio(medcol));

  for (double alpha=0.0; alpha<=1.0; alpha+=0.1) {
    double beta=1.0-alpha;
    //double medianatotal=alpha*mediana(medlin)+beta*mediana(medcol);
    IMGDBL b(a.nl(),a.nc());
    for (int l=0; l<a.nl(); l++)
      for (int c=0; c<a.nc(); c++) {
        //b(l,c)=medianatotal+(alpha*(medlin(l)-medianatotal)+beta*(medcol(c)-medianatotal));
        b(l,c)=alpha*medlin(l)+beta*medcol(c);
      }
    imp(b,argv[2]);
    printf("Alpha=%f beta=%f MAE=%f RMSE=%f\n",alpha,beta,mae(a,b),rmse(a,b));
  }
}

void estmed(int argc, char** argv)
{ if (argc!=2) {
    printf("EstMed: Estima a linha desconhecida pela mediana da coluna\n");
    printf("EstMed ent.img\n");
    erro("Erro: Numero de argumentos invalido");
  }
  IMGDBL a; le(a,argv[1]);
  IMGDBL b(a.nl(),a.nc());

  VETOR<double> temp(a.nl()-1);
  for (int l=0; l<a.nl(); l++) { // linha considerada desconhecida
    for (int c=0; c<a.nc(); c++) {
      int i=0;
      for (int l2=0; l2<a.nl(); l2++) {
        if (l!=l2) { temp(i)=a(l2,c); i++; }
      }
      b(l,c)=mediana(temp);
    }
  }
  printf("MAE=%f RMSE=%f\n",mae(a,b),rmse(a,b));
}

void estmean(int argc, char** argv)
{ if (argc!=2) {
    printf("EstMean: Estima a linha desconhecida pela media da coluna\n");
    printf("EstMean ent.img\n");
    erro("Erro: Numero de argumentos invalido");
  }
  IMGDBL a; le(a,argv[1]);
  IMGDBL b(a.nl(),a.nc());

  VETOR<double> temp(a.nl()-1);
  for (int l=0; l<a.nl(); l++) { // linha considerada desconhecida
    for (int c=0; c<a.nc(); c++) {
      int i=0;
      for (int l2=0; l2<a.nl(); l2++) {
        if (l!=l2) { temp(i)=a(l2,c); i++; }
      }
```

```
      b(l,c)=media(temp);
    }
  }
  printf("MAE=%f RMSE=%f\n",mae(a,b),rmse(a,b));
}

void estknn(int argc, char** argv)
{ if (argc!=4) {
    printf("EstKnn: Estima a linha desconhecida pela knn - media\n");
    printf("EstKnn ent.img pos.txt k\n");
    erro("Erro: Numero de argumentos invalido");
  }
  const int grande=10000;
  IMGDBL a; le(a,argv[1]);
  IMGDBL _pos; le(_pos,argv[2]);
  VETOR<PONTOI2> pos(_pos.nl());
  for (int i=0; i<pos.n(); i++)
    pos(i)=PONTOI2(arredonda(_pos(i,0)),arredonda(_pos(i,1)));

  int k; ConvArg(k,argv[3]);
  IMGSHT nn(pos.n(),k);

  for (int l=0; l<nn.nl(); l++) {
    VETOR<PONTOI2> p=pos;
    p(l)=PONTOI2(grande,grande);
    for (int c=0; c<nn.nc(); c++) {
      double menord=1e10;
      int menori=-1;
      for (int i=0; i<nn.nl(); i++) {
        if (distancia(pos(l),p(i))<=menord) {
          menord=distancia(pos(l),p(i));
          menori=i;
        }
      }
      nn(l,c)=menori;
      p(menori)=PONTOI2(grande,grande);
    }
  }
  //cout << nn << endl;

  IMGDBL b(a.nl(),a.nc());
  //VETOR<double> temp(a.nl()-1);
  for (int l=0; l<a.nl(); l++) { // l = linha considerada desconhecida
    for (int c=0; c<a.nc(); c++) {
      double estimativa=0;
      for (int i=0; i<nn.nc(); i++)
        estimativa += a( nn(l,i), c );
      b(l,c)=estimativa/nn.nc();
    }
  }
  printf("MAE=%f RMSE=%f\n",mae(a,b),rmse(a,b));
  imp(b,"teste_estknn.txt");
}

void estknnmed(int argc, char** argv)
{ if (argc!=4) {
    printf("EstKnnMed: Estima a linha desconhecida pela knn - mediana\n");
    printf("EstKnnMed ent.img pos.txt k\n");
    erro("Erro: Numero de argumentos invalido");
```

```
  }
  const int grande=10000;
  IMGDBL a; le(a,argv[1]);
  // a(l,c) contem leitura da janela l no instante c
  IMGDBL _pos; le(_pos,argv[2]);
  VETOR<PONTOI2> pos(_pos.nl());
  for (int i=0; i<pos.n(); i++)
    pos(i)=PONTOI2(arredonda(_pos(i,0)),arredonda(_pos(i,1)));
  //pos(i) contem a posicao da janela i

  int k; ConvArg(k,argv[3]);
  IMGSHT nn(pos.n(),k);
  for (int l=0; l<nn.nl(); l++) {
    VETOR<PONTOI2> p=pos;
    p(l)=PONTOI2(grande,grande);
    for (int c=0; c<nn.nc(); c++) {
      double menord=1e10;
      int menori=-1;
      for (int i=0; i<nn.nl(); i++) {
        if (distancia(pos(l),p(i))<=menord) {
          menord=distancia(pos(l),p(i));
          menori=i;
        }
      }
      nn(l,c)=menori;
      p(menori)=PONTOI2(grande,grande);
    }
  }
  //cout << nn << endl;
  //nn(l,c) contem c-esima janela mais proxima da janela l

  IMGDBL b(a.nl(),a.nc());
  VETOR<double> temp(nn.nc());
  for (int l=0; l<a.nl(); l++) { // l = linha considerada desconhecida
    for (int c=0; c<a.nc(); c++) { // c = instante
      for (int i=0; i<nn.nc(); i++) { // i = i-esimo vizinho mais proximo
        temp(i) = a( nn(l,i), c );
      }
      b(l,c)=mediana(temp);
    }
  }
  // b contem estimacao de a (calculado supondo desconhecer cada linha l por vez)

  printf("MAE=%f RMSE=%f\n",mae(a,b),rmse(a,b));
}

void estklnn(int argc, char** argv)
{ if (argc!=5) {
    printf("EstKLnn: Est lin. desc. pela media dos k-viz espac. e l-viz. temp.\n");
    printf("EstKLnn ent.img pos.txt k l\n");
    erro("Erro: Numero de argumentos invalido");
  }
  const int grande=10000;
  IMGDBL a; le(a,argv[1]);
  // a(l,c) contem leitura da janela l no instante c
  IMGDBL _pos; le(_pos,argv[2]);
  VETOR<PONTOI2> pos(_pos.nl());
  for (int i=0; i<pos.n(); i++)
    pos(i)=PONTOI2(arredonda(_pos(i,0)),arredonda(_pos(i,1)));
```

```
  //pos(i) contem a posicao da janela i

  int K; ConvArg(K,argv[3]);
  IMGSHT nn(pos.n(),K);
  int L; ConvArg(L,argv[4]);

  for (int l=0; l<nn.nl(); l++) {
    VETOR<PONTOI2> p=pos;
    p(l)=PONTOI2(grande,grande);
    for (int c=0; c<nn.nc(); c++) {
      double menord=1e10;
      int menori=-1;
      for (int i=0; i<nn.nl(); i++) {
        if (distancia(pos(l),p(i))<=menord) {
          menord=distancia(pos(l),p(i));
          menori=i;
        }
      }
      nn(l,c)=menori;
      p(menori)=PONTOI2(grande,grande);
    }
  }
  //cout << nn << endl;
  //nn(l,c) contem c-esima janela mais proxima da janela l

  IMGDBL b(a.nl(),a.nc());
  //VETOR<double> temp(a.nl()-1);
  for (int l=0; l<a.nl(); l++) { // l = linha considerada desconhecida
    for (int c=0; c<a.nc(); c++) { // c = instante
      // tenta estimar a(l,c) e armazena resultado em b(l,c)
      double estimativa=0;
      int conta=0;
      for (int i=0; i<nn.nc(); i++) { // i = i-esimo vizinho espacial mais proximo
        for (int c2=c-L/2; c2<c-L/2+L; c2++) { // c2 = vizinho temporal
          if (0<=c2 && c2<b.nc()) {
            estimativa += a( nn(l,i), c2 );
            conta++;
          }
        }
      }
      b(l,c)=estimativa/conta;
    }
  }
  printf("MAE=%f RMSE=%f\n",mae(a,b),rmse(a,b));
}

void estimag(int argc, char** argv)
{ if (argc!=4) {
    printf("EstImag: Estima a linha desconhecida pelo valor na imagem\n");
    printf("EstImag ent.img pos.txt imag.img\n");
    erro("Erro: Numero de argumentos invalido");
  }
  IMGDBL a; le(a,argv[1]); // leituras das janelas no tempo
  IMGDBL _pos; le(_pos,argv[2]); // posicoes das janelas
  if (a.nl()!=_pos.nl()) erro("Erro: Numero de janelas nao bate");
  VETOR<PONTOI2> pos(_pos.nl());
  for (int i=0; i<pos.n(); i++)
    pos(i)=PONTOI2(arredonda(_pos(i,0)),arredonda(_pos(i,1)));
  // pos contem as posicoes das janelas
```

```
  IMGDBL img; le(img,argv[3]); // img=imagem completa

  IMGDBL b(a.nl(),a.nc()); // leituras das janelas estimadas
  for (int l=0; l<a.nl(); l++) { // l = linha considerada desconhecida
    int xi=pos(l).x(); int yi=pos(l).y();
    double estimativa=0;
    for (int x=xi; x<xi+10; x++)
      for (int y=yi; y<yi+10; y++)
        estimativa += img.atN(x,y);
    estimativa = estimativa/100;
    for (int c=0; c<a.nc(); c++) {
      b(l,c)=estimativa;
    }
  }
  printf("MAE=%f RMSE=%f\n",mae(a,b),rmse(a,b));
  for (int l=0; l<b.nl(); l++)
    printf("%9.3f ",abs(a(l,0)-b(l,0)));
  printf("\n");
}

void estknnim(int argc, char** argv)
{ if (argc!=5) {
    printf("EstKnnIm: Estima a linha desconhecida pelo k-NN\n");
    printf("  seguida pela correcao de media pela imagem completa\n");
    printf("EstKnnIM ent.img pos.txt k imag.img\n");
    erro("Erro: Numero de argumentos invalido");
  }

  const int grande=10000;
  IMGDBL a; le(a,argv[1]);
  IMGDBL _pos; le(_pos,argv[2]);
  VETOR<PONTOI2> pos(_pos.nl());
  for (int i=0; i<pos.n(); i++)
    pos(i)=PONTOI2(arredonda(_pos(i,0)),arredonda(_pos(i,1)));

  int k; ConvArg(k,argv[3]);
  IMGSHT nn(pos.n(),k);

  for (int l=0; l<nn.nl(); l++) {
    VETOR<PONTOI2> p=pos;
    p(l)=PONTOI2(grande,grande);
    for (int c=0; c<nn.nc(); c++) {
      double menord=1e10;
      int menori=-1;
      for (int i=0; i<nn.nl(); i++) {
        if (distancia(pos(l),p(i))<=menord) {
          menord=distancia(pos(l),p(i));
          menori=i;
        }
      }
      nn(l,c)=menori;
      p(menori)=PONTOI2(grande,grande);
    }
  }

  IMGDBL b(a.nl(),a.nc());
  for (int l=0; l<a.nl(); l++) { // l = linha considerada desconhecida
    for (int c=0; c<a.nc(); c++) {
      double estimativa=0;
```

```
      for (int i=0; i<nn.nc(); i++)
        estimativa += a( nn(l,i), c );
      b(l,c)=estimativa/nn.nc();
    }
  }
  // b e' a estimacao por k-NN

  VETOR<double> medlin(b.nl());
  VETOR<double> temp1(b.nc());
  for (int l=0; l<b.nl(); l++) {
    for (int c=0; c<b.nc(); c++) temp1(c)=b(l,c);
    medlin(l)=media(temp1);
  }
  // medlin tem media linha a linha

  IMGDBL img; le(img,argv[4]); // img=imagem completa
  VETOR<double> d(b.nl()); // janelas estimadas pela imagem completa
  for (int l=0; l<b.nl(); l++) { // l = linha considerada desconhecida
    int xi=pos(l).x(); int yi=pos(l).y();
    double estimativa=0;
    for (int x=xi; x<xi+10; x++)
      for (int y=yi; y<yi+10; y++)
        estimativa += img.atN(x,y);
    d(l) = estimativa/100;
  }
  // d contem estimativa pela imagem completa

  for (int l=0; l<b.nl(); l++)
    for (int c=0; c<b.nc(); c++)
      b(l,c) = b(l,c) - medlin(l) + d(l);

  printf("MAE=%f RMSE=%f\n",mae(a,b),rmse(a,b));
}

void estmeanim(int argc, char** argv)
{ if (argc!=4) {
    printf("EstMeanIm: Estima a linha desconhecida pelo media das janelas\n");
    printf("  seguida pela correcao de media pela imagem completa\n");
    printf("EstMeanIm ent.img pos.txt imag.img\n");
    erro("Erro: Numero de argumentos invalido");
  }

  IMGDBL a; le(a,argv[1]);

  IMGDBL b(a.nl(),a.nc());
  VETOR<double> temp(a.nl()-1);
  for (int l=0; l<a.nl(); l++) { // linha considerada desconhecida
    for (int c=0; c<a.nc(); c++) {
      int i=0;
      for (int l2=0; l2<a.nl(); l2++) {
        if (l!=l2) { temp(i)=a(l2,c); i++; }
      }
      b(l,c)=media(temp);
    }
  }
  // b e' a estimacao por media coluna a coluna

  VETOR<double> medlin(b.nl());
  VETOR<double> temp1(b.nc());
```

```
  for (int l=0; l<b.nl(); l++) {
    for (int c=0; c<b.nc(); c++) temp1(c)=b(l,c);
    medlin(l)=media(temp1);
  }
  // medlin tem media linha a linha

  IMGDBL _pos; le(_pos,argv[2]);
  VETOR<PONTOI2> pos(_pos.nl());
  for (int i=0; i<pos.n(); i++)
    pos(i)=PONTOI2(arredonda(_pos(i,0)),arredonda(_pos(i,1)));

  IMGDBL img; le(img,argv[3]); // img=imagem completa
  VETOR<double> d(b.nl()); // janelas estimadas pela imagem completa
  for (int l=0; l<b.nl(); l++) { // l = linha considerada desconhecida
    int xi=pos(l).x(); int yi=pos(l).y();
    double estimativa=0;
    for (int x=xi; x<xi+10; x++)
      for (int y=yi; y<yi+10; y++)
        estimativa += img.atN(x,y);
    d(l) = estimativa/100;
  }
  // d contem estimativa pela imagem completa

  for (int l=0; l<b.nl(); l++)
    for (int c=0; c<b.nc(); c++)
      b(l,c) = b(l,c) - medlin(l) + d(l);

  printf("MAE=%f RMSE=%f\n",mae(a,b),rmse(a,b));
}

void estmedim(int argc, char** argv)
{ if (argc!=4) {
    printf("EstMedIm: Estima a linha desconhecida pelo mediana das janelas\n");
    printf("  seguida pela correcao de media pela imagem completa\n");
    printf("EstMedIm ent.img pos.txt imag.img\n");
    erro("Erro: Numero de argumentos invalido");
  }

  IMGDBL a; le(a,argv[1]);

  IMGDBL b(a.nl(),a.nc());
  VETOR<double> temp(a.nl()-1);
  for (int l=0; l<a.nl(); l++) { // linha considerada desconhecida
    for (int c=0; c<a.nc(); c++) {
      int i=0;
      for (int l2=0; l2<a.nl(); l2++) {
        if (l!=l2) { temp(i)=a(l2,c); i++; }
      }
      b(l,c)=mediana(temp);
    }
  }
  // b e' a estimacao por media coluna a coluna

  VETOR<double> medlin(b.nl());
  VETOR<double> temp1(b.nc());
  for (int l=0; l<b.nl(); l++) {
    for (int c=0; c<b.nc(); c++) temp1(c)=b(l,c);
    medlin(l)=media(temp1);
  }
```

```
  // medlin tem media linha a linha

  IMGDBL _pos; le(_pos,argv[2]);
  VETOR<PONTOI2> pos(_pos.nl());
  for (int i=0; i<pos.n(); i++)
    pos(i)=PONTOI2(arredonda(_pos(i,0)),arredonda(_pos(i,1)));

  IMGDBL img; le(img,argv[3]); // img=imagem completa
  VETOR<double> d(b.nl()); // janelas estimadas pela imagem completa
  for (int l=0; l<b.nl(); l++) { // l = linha considerada desconhecida
    int xi=pos(l).x(); int yi=pos(l).y();
    double estimativa=0;
    for (int x=xi; x<xi+10; x++)
      for (int y=yi; y<yi+10; y++)
        estimativa += img.atN(x,y);
    d(l) = estimativa/100;
  }
  // d contem estimativa pela imagem completa

  for (int l=0; l<b.nl(); l++)
    for (int c=0; c<b.nc(); c++)
      b(l,c) = b(l,c) - medlin(l) + d(l);

  printf("MAE=%f RMSE=%f\n",mae(a,b),rmse(a,b));
}

void estmedknn(int argc, char** argv)
{ if (argc!=4) {
    printf("EstMedKnn: Estima a linha desconhecida pelo mediana das janelas\n");
    printf("  seguida pela correcao de media pela knn da mediana das linhas\n");
    printf("EstMedKnn ent.img pos.txt k\n");
    erro("Erro: Numero de argumentos invalido");
  }

  IMGDBL a; le(a,argv[1]); // dados de entrada

  IMGDBL b(a.nl(),a.nc()); // dados estimados de saida pela mediana
  VETOR<double> temp(a.nl()-1);
  for (int l=0; l<a.nl(); l++) { // linha considerada desconhecida
    for (int c=0; c<a.nc(); c++) {
      int i=0;
      for (int l2=0; l2<a.nl(); l2++) {
        if (l!=l2) { temp(i)=a(l2,c); i++; }
      }
      b(l,c)=mediana(temp);
    }
  }
  // b(l,c) contem mediana da coluna c supondo linha l desconhecida

  VETOR<DBL> d(a.nl()); // mediana das linhas
  VETOR<double> temp2(a.nc());
  for (int l=0; l<a.nl(); l++) {
    for (int c=0; c<a.nc(); c++) {
      temp2(c)=a(l,c);
    }
    d(l)=mediana(temp2);
  }
  // d(l) contem mediana da linha l
```

```
  IMGDBL _pos; le(_pos,argv[2]);
  VETOR<PONTOI2> pos(_pos.nl());
  for (int i=0; i<pos.n(); i++)
    pos(i)=PONTOI2(arredonda(_pos(i,0)),arredonda(_pos(i,1)));
  int k; ConvArg(k,argv[3]);
  IMGSHT nn(pos.n(),k);
  const int grande=10000;
  for (int l=0; l<nn.nl(); l++) {
    VETOR<PONTOI2> p=pos;
    p(l)=PONTOI2(grande,grande);
    for (int c=0; c<nn.nc(); c++) {
      double menord=1e10;
      int menori=-1;
      for (int i=0; i<nn.nl(); i++) {
        if (distancia(pos(l),p(i))<=menord) {
          menord=distancia(pos(l),p(i));
          menori=i;
        }
      }
      nn(l,c)=menori;
      p(menori)=PONTOI2(grande,grande);
    }
  }
  // cout << nn << endl;
  // nn(l,c) contem c-esimo NN de janela l.

  VETOR<DBL> e(a.nl()); // estimativa de media da linha e
  for (int l=0; l<a.nl(); l++) { // l = linha considerada desconhecida
    double estimativa=0;
    for (int i=0; i<nn.nc(); i++)
      estimativa += d( nn(l,i) );
    e(l)=estimativa/nn.nc();
  }
  //cout << e << endl;
  // e(l) e' a estimacao do valor medio pela kNN da mediana das linhas

  VETOR<double> medlin(b.nl());
  VETOR<double> temp1(b.nc());
  for (int l=0; l<b.nl(); l++) {
    for (int c=0; c<b.nc(); c++) temp1(c)=b(l,c);
    medlin(l)=media(temp1);
  }
  //cout << medlin << endl;
  // medlin tem media linha a linha

  for (int l=0; l<b.nl(); l++)
    for (int c=0; c<b.nc(); c++)
      b(l,c) = b(l,c) - medlin(l) + e(l);

  printf("MAE=%f RMSE=%f\n",mae(a,b),rmse(a,b));
}

//<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< main <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
const char about[]=
"< Janela: Programas para processar janelas de Corot 1.05 >\n"
"_______________________________________________________________________________\n";

const char help[]=
"Programs:\n"
```

```
"  DistD    - Diferenca entre duas IMGDBL\n"
"  Mediad   - Media de n IMGDBL\n"
"  MarcaIm  - Marca a imagem completa com estrelas/janelas\n"
"  PegaCol  - Pega uma coluna num arquivo TXT para tracar grafico\n"
"  MinMax   - Acha colunas cujas medianas sao minimo e maximo\n"
"  TXT2IMG  - Converte .TXT para .IMG para acelerar a leitura\n"
"  ReesMed  - Reestima os dados pela mediana linha/coluna e comb linear\n"
"  ReesMean - Reestima os dados pela media linha/coluna e comb linear\n"
"  EstMed   - Estima a linha desconhecida pela mediana da coluna\n"
"  EstMean  - Estima a linha desconhecida pela media da coluna\n"
"  EstKnn   - Estima a linha desconhecida pela knn - media\n"
"  EstKnnMed- Estima a linha desconhecida pela knn - mediana\n"
"  EstKLnn  - Est lin. desc. pela media dos k-viz espac. e l-viz. temp.\n"
"  EstImag  - Estima a linha desconhecida pelo valor na imagem\n"
"  EstKnnIm - Estima pela knn-media seguida pela correcao pela imagem completa\n"
"  EstMeanIm- Estima pela media seguida pela correcao pela imagem completa\n"
"  EstMedIm - Estima pela mediana seguida pela correcao pela imagem completa\n"
"  EstMedKnn- Est pela mediana seguida pela correcao pela KNN de mediana de lin.\n"
"..............................................................................\n";

int main(int argc, char** argv)
{
  if (argc<2) {
    printf(about);
    printf(help);
    erro("Erro: Numero de argumentos invalido");
  } else {
    string comando=strlwr(argv[1]);
    if      (comando=="distd")     distd(argc-1,&argv[1]);
    else if (comando=="mediad")    mediad(argc-1,&argv[1]);
    else if (comando=="marcaim")   marcaim(argc-1,&argv[1]);
    else if (comando=="pegacol")   pegacol(argc-1,&argv[1]);
    else if (comando=="minmax")    minmax(argc-1,&argv[1]);
    else if (comando=="txt2img")   txt2img(argc-1,&argv[1]);
    else if (comando=="reesmed")   reesmed(argc-1,&argv[1]);
    else if (comando=="reesmean")  reesmean(argc-1,&argv[1]);
    else if (comando=="estmed")    estmed(argc-1,&argv[1]);
    else if (comando=="estmean")   estmean(argc-1,&argv[1]);
    else if (comando=="estknn")    estknn(argc-1,&argv[1]);
    else if (comando=="estknnmed") estknnmed(argc-1,&argv[1]);
    else if (comando=="estklnn")   estklnn(argc-1,&argv[1]);
    else if (comando=="estimag")   estimag(argc-1,&argv[1]);
    else if (comando=="estknnim")  estknnim(argc-1,&argv[1]);
    else if (comando=="estmeanim") estmeanim(argc-1,&argv[1]);
    else if (comando=="estmedim")  estmedim(argc-1,&argv[1]);
    else if (comando=="estmedknn") estmedknn(argc-1,&argv[1]);
    else erro("Erro: Programa inexistente ",comando.c_str());
  }
}
```