```
// Programa kbox.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 alocação de janela de fundo de céu.

#include <proeikon>
#include <imgpv>

//<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< Variaveis e constantes globais

const int maxx=2048;
const int maxy=2048;

//<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< Funcoes uteis

void retangulo(IMGCOR& a, PONTOI2 p1, PONTOI2 p2, COR cor)
// Desenha retangulo alinhado aos eixos.
{ for (int x=p1.x(); x<=p2.x(); x++) { a(x,p1.y(),'C')=cor; a(x,p2.y(),'C')=cor; }
  for (int y=p1.y(); y<=p2.y(); y++) { a(p1.x(),y,'C')=cor; a(p2.x(),y,'C')=cor; }
}

double maiordistancia(VETOR<PONTOI2>& estrelas, VETOR<PONTOI2>& janelas, int& maiore, int&
maiorj)
// Acha a estrela que esta mais distante da janela mais proxima.
// Retorna a distancia
// maiore e maiorj sao indices para estrela e janela
{ double maior=-infinito;
  maiore=0; maiorj=0;
  for (int e=0; e<estrelas.n(); e++) {
    double menor=infinito; int menorj=0;
    for (int j=0; j<janelas.n(); j++) {
      double m=distancia(janelas(j),estrelas(e));
      if (m<menor) { menor=m; menorj=j; }
    }
    if (menor>maior) { maior=menor; maiore=e; maiorj=menorj; }
  }
  return maior;
}

double maiordistancia(VETOR<PONTOI2>& estrelas, VETOR<PONTOI2>& janelas)
// Acha a estrela que esta mais distante da janela mais proxima.
// E retorna a distância
{ int maiore,maiorj;
  return maiordistancia(estrelas,janelas,maiore,maiorj);
}


VETOR<int> janelapx(VETOR<PONTOI2>& estrelas, VETOR<PONTOI2>& janelas)
// Calcula a janela mais proxima de cada estrela
{ VETOR<int> janpx(estrelas.n());
  for (int e=0; e<estrelas.n(); e++) {
    double menor=infinito; int menorj=0;
    for (int j=0; j<janelas.n(); j++) {
      double m=distancia(janelas(j),estrelas(e));
      if (m<menor) { menor=m; menorj=j; }
    }
```

```
    janpx(e)=menorj;
  }
  return janpx;
}

IMGCOR criaimagem1(VETOR<PONTOI2>& estrelas, VETOR<PONTOI2>& janelas)
// Cria imagem com estrelas e janelas
{ VETOR<int> janpx=janelapx(estrelas,janelas);
  VETOR<PONTOI2> bl(janelas.n(),PONTOI2(maxx-1,maxy-1));
  VETOR<PONTOI2> ur(janelas.n(),PONTOI2(0,0));
  for (int e=0; e<estrelas.n(); e++) {
    int j=janpx(e);
    int x=estrelas(e).x();
    int y=estrelas(e).y();
    if (x<bl(j).x()) bl(j).x()=x;
    if (ur(j).x()<x) ur(j).x()=x;
    if (y<bl(j).y()) bl(j).y()=y;
    if (ur(j).y()<y) ur(j).y()=y;
  }
  IMGCOR a(maxy,maxx,COR(255,255,255)); a.lc()=a.nl()-1; a.cc()=0;
  for (int e=0; e<estrelas.n(); e++) {
    reta(a,estrelas(e),janelas(janpx(e)),COR(255,255,0));
  }
  for (int j=0; j<janelas.n(); j++) {
    if (bl(j)==PONTOI2(maxx-1,maxy-1) && ur(j)==PONTOI2(0,0)) printf("Warning: janela vazia
    %d\n",j);
    else retangulo(a,bl(j),ur(j),COR(255,0,0));
  }

  for (int j=0; j<janelas.n(); j++) {
    int x=janelas(j).x();
    int y=janelas(j).y();
    a(x,y,'N')=COR(255,0,0);
    a(x-1,y,'N')=COR(255,0,0);
    a(x+1,y,'N')=COR(255,0,0);
    a(x,y-1,'N')=COR(255,0,0);
    a(x,y+1,'N')=COR(255,0,0);
  }

  int maiore, maiorj;
  maiordistancia(estrelas, janelas, maiore, maiorj);
  reta(a,estrelas(maiore),janelas(maiorj),COR(0,0,255));

  for (int e=0; e<estrelas.n(); e++) {
    int x=estrelas(e).x();
    int y=estrelas(e).y();
    a(x,y,'N')=COR(0,0,0);
  }

  pvMostra(a,0,"janela");
  int ch=cvWaitKey();
  if (ch=='h') pvImp(a,"maxdist.png");
  //cvDestroyAllWindows();

  return a;
}

IMGCOR criaimagem2(VETOR<PONTOI2>& estrelas, VETOR<PONTOI2>& janelas)
{ VETOR<int> janpx=janelapx(estrelas,janelas);
```

```
  IMGCOR a(maxy,maxx,COR(255,255,255)); a.lc()=a.nl()-1; a.cc()=0;
  for (int e=0; e<estrelas.n(); e++) {
    reta(a,estrelas(e),janelas(janpx(e)),COR(255,0,0));
  }

  for (int j=0; j<janelas.n(); j++) {
    int x=janelas(j).x();
    int y=janelas(j).y();
    a(x,y,'N')=COR(255,0,0);
    a(x-1,y,'N')=COR(255,0,0);
    a(x+1,y,'N')=COR(255,0,0);
    a(x,y-1,'N')=COR(255,0,0);
    a(x,y+1,'N')=COR(255,0,0);
  }

  int maiore, maiorj;
  maiordistancia(estrelas, janelas, maiore, maiorj);
  reta(a,estrelas(maiore),janelas(maiorj),COR(0,0,255));

  for (int e=0; e<estrelas.n(); e++) {
    int x=estrelas(e).x();
    int y=estrelas(e).y();
    a(x,y,'N')=COR(0,0,0);
  }

  pvMostra(a,0,"janela");
  int ch=cvWaitKey();
  if (ch=='h') pvImp(a,"maxdist.png");
  //cvDestroyAllWindows();

  return a;
}

//<<<<<<<<<<<<<<<<<<<<<<<<< estrela

void estrela(int argc, char** argv)
{ if (argc!=4) {
    printf("Estrela: Gera estrelas aleatoriamente\n");
    printf("Estrela estrela.txt semente nestrelas\n");
    erro("Erro: Numero de argumentos invalido");
  }

  int semente;
  if (sscanf(argv[2],"%d",&semente)!=1) erro("Erro: Leitura semente");
  mysrand(semente);

  int njanelas;
  if (sscanf(argv[3],"%d",&njanelas)!=1) erro("Erro: Leitura njanelas");

  VETOR<PONTOI2> janelas(njanelas);

  for (int i=0; i<janelas.n(); i++) {
    janelas(i).x()=myrand()%maxx;
    janelas(i).y()=myrand()%maxy;
  }
  imp(janelas,argv[1]);
}
```

```
//<<<<<<<<<<<<<<<<<<<<<<<<<<<< uniform

void uniform2(VETOR<PONTOI2>& janelas, int njanelas)
{ janelas.resize(njanelas);

  double dx=sqrt(double(njanelas)*double(maxx)/double(maxy));
  double dy=dx*double(maxy)/double(maxx);
  int nx=arredonda(dx); int ny=arredonda(dy);
  if (nx*ny>njanelas) {
    nx=chao(dx); ny=arredonda(dy);
    if (nx*ny>njanelas) {
      nx=arredonda(dx); ny=chao(dy);
      if (nx*ny>njanelas) {
        nx=chao(dx); ny=chao(dy);
        if (nx*ny>njanelas) erro("Erro inesperado 1");
      }
    }
  }
  //printf("dx*dy=%f dx=%f dy=%f nx=%d ny=%d\n",dx*dy,dx,dy,nx,ny);
  dx=maxx/(2.0*nx);
  dy=maxy/(2.0*ny);

  int i=0;
  for (int x=0; x<nx; x++)
    for (int y=0; y<ny; y++)
      janelas(i++)=PONTOI2(arredonda(2*x*dx+dx),arredonda(2*y*dy+dy));
  for (; i<janelas.n(); i++)
    janelas(i)=janelas(0);
}

void uniform(int argc, char** argv)
{ if (argc!=3) {
    printf("Uniform: Gera janelas uniformemente\n");
    printf("uniform janela.txt njanelas\n");
    erro("Erro: Numero de argumentos invalido");
  }

  int njanelas;
  if (sscanf(argv[2],"%d",&njanelas)!=1) erro("Erro: Leitura njanelas");

  VETOR<PONTOI2> janelas;
  uniform2(janelas,njanelas);

  imp(janelas,argv[1]);
}

//<<<<<<<<<<<<<<<<<<<<<<<<<<<< aleat

void aleat2(VETOR<PONTOI2>& janelas, int semente, int njanelas)
{ mysrand(semente);
  janelas.resize(njanelas);
  for (int i=0; i<janelas.n(); i++) {
    janelas(i).x()=myrand()%maxx;
    janelas(i).y()=myrand()%maxy;
  }
}

void aleat(int argc, char** argv)
{ if (argc!=4) {
```

```
    printf("Aleat: Gera janelas aleatoriamente\n");
    printf("aleat janela.txt semente njanelas\n");
    erro("Erro: Numero de argumentos invalido");
  }

  int semente;
  if (sscanf(argv[2],"%d",&semente)!=1) erro("Erro: Leitura semente");
  int njanelas;
  if (sscanf(argv[3],"%d",&njanelas)!=1) erro("Erro: Leitura njanelas");

  VETOR<PONTOI2> janelas;
  aleat2(janelas,semente,njanelas);
  imp(janelas,argv[1]);
}

void naleat(int argc, char** argv)
{ if (argc!=6) {
    printf("nAleat: Gera n sets de janelas aleatoriamente e escolhe o melhor\n");
    printf("naleat estrela.txt janela.txt njanelas semente nsemente\n");
    erro("Erro: Numero de argumentos invalido");
  }

  int njanelas;
  if (sscanf(argv[3],"%d",&njanelas)!=1) erro("Erro: Leitura njanelas");
  int semente;
  if (sscanf(argv[4],"%d",&semente)!=1) erro("Erro: Leitura semente");
  int nsemente;
  if (sscanf(argv[5],"%d",&nsemente)!=1) erro("Erro: Leitura nsemente");

  VETOR<PONTOI2> estrelas; le(estrelas,argv[1]);

  VETOR<PONTOI2> janelas;
  VETOR<PONTOI2> melhoresjanelas;
  double melhor=infinito;

  for (int i=semente; i<semente+nsemente; i++) {
    printf("%8d %10g\r",i-semente,melhor);
    aleat2(janelas,i,njanelas);
    double d=maiordistancia(estrelas,janelas);
    if (d<melhor) { melhor=d; melhoresjanelas=janelas; }
  }
  printf("\n");
  imp(melhoresjanelas,argv[2]);
}

//<<<<<<<<<<<<<<<<<<<<<<<<<<< kbb

void kbb3(VETOR<PONTOI2>& estrelas, VETOR<PONTOI2>& janelas)
{ VETOR<int> janpx=janelapx(estrelas,janelas);
  VETOR<PONTOI2> bl(janelas.n(),PONTOI2(maxx-1,maxy-1));
  VETOR<PONTOI2> ur(janelas.n(),PONTOI2(0,0));
  for (int e=0; e<estrelas.n(); e++) {
    int j=janpx(e);
    int x=estrelas(e).x();
    int y=estrelas(e).y();
    if (x<bl(j).x()) bl(j).x()=x;
    if (ur(j).x()<x) ur(j).x()=x;
    if (y<bl(j).y()) bl(j).y()=y;
    if (ur(j).y()<y) ur(j).y()=y;
```

```
  }
  for (int j=0; j<janelas.n(); j++) {
    janelas(j).x()=(bl(j).x()+ur(j).x())/2;
    janelas(j).y()=(bl(j).y()+ur(j).y())/2;
  }
}

bool operator==(VETOR<PONTOI2>& a, VETOR<PONTOI2>& b)
{ if (a.n()!=b.n()) return false;
  bool igual=true;
  for (int i=0; i<a.n(); i++)
    if (!(a(i)==b(i))) { igual=false; break; }
  return igual;
}

void kbb2(VETOR<PONTOI2>& estrelas, VETOR<PONTOI2>& janelas)
{
  double melhor=maiordistancia(estrelas,janelas);
  VETOR<PONTOI2> melhoresjanelas=janelas;
  //printf("melhor=%f dist=%f\n",melhor,melhor);

  VETOR<PONTOI2> janelasant;
  do {
    janelasant=janelas;
    kbb3(estrelas,janelas);
    double d=maiordistancia(estrelas,janelas);
    if (d<melhor) { melhor=d; melhoresjanelas=janelas; }
    //printf("melhor=%f dist=%f\n",melhor,d);
  } while (!(janelas==janelasant));

  janelas=melhoresjanelas;
}

void kbb(int argc, char** argv)
{ if (argc!=5) {
    printf("kbb: Gera janelas aleatoriamente e otimiza com kBB\n");
    printf("kbb estrela.txt janela.txt semente njanelas\n");
    erro("Erro: Numero de argumentos invalido");
  }

  int semente;
  if (sscanf(argv[3],"%d",&semente)!=1) erro("Erro: Leitura semente");

  int njanelas;
  if (sscanf(argv[4],"%d",&njanelas)!=1) erro("Erro: Leitura njanelas");

  VETOR<PONTOI2> estrelas; le(estrelas,argv[1]);

  VETOR<PONTOI2> janelas;
  //uniform2(janelas,njanelas);
  aleat2(janelas,semente,njanelas);

  kbb2(estrelas,janelas);
  //printf("Maior=%f\n",maiordistancia(estrelas,janelas));
  //criaimagem1(estrelas,janelas);
  imp(janelas,argv[2]);
}

void nkbb(int argc, char** argv)
```

```
{ if (argc!=6) {
    printf("nkBB: Gera n sets de janelas aleat, otimiza com kBB e escolhe o melhor\n");
    printf("nkbb estrela.txt janela.txt njanelas semente nsemente\n");
    erro("Erro: Numero de argumentos invalido");
  }

  int njanelas;
  if (sscanf(argv[3],"%d",&njanelas)!=1) erro("Erro: Leitura njanelas");
  int semente;
  if (sscanf(argv[4],"%d",&semente)!=1) erro("Erro: Leitura semente");
  int nsemente;
  if (sscanf(argv[5],"%d",&nsemente)!=1) erro("Erro: Leitura nsemente");

  VETOR<PONTOI2> estrelas; le(estrelas,argv[1]);

  VETOR<PONTOI2> janelas;
  VETOR<PONTOI2> melhoresjanelas;
  double melhor=infinito;

  for (int i=semente; i<semente+nsemente; i++) {
    printf("%8d %10g\r",i-semente,melhor);
    aleat2(janelas,i,njanelas);
    kbb2(estrelas,janelas);
    double d=maiordistancia(estrelas,janelas);
    if (d<melhor) { melhor=d; melhoresjanelas=janelas; }
  }
  printf("\n");
  imp(melhoresjanelas,argv[2]);
}

//<<<<<<<<<<<<<<<<<<<<<<<<< kmeans

int janelapx(PONTOI2 estrela, VETOR<PONTOI2>& janelas)
// Calcula a janela mais proxima de uma estrela e retorna o seu indice
{ double menor=infinito; int menorj=0;
  for (int j=0; j<janelas.n(); j++) {
    double m=distancia(janelas(j),estrela);
    if (m<menor) { menor=m; menorj=j; }
  }
  return menorj;
}

double distanciamedia(VETOR<PONTOI2>& estrelas, VETOR<PONTOI2>& janelas)
// Acha a distancia media entre cada estrela e a sua janela mais proxima
{ double soma=0.0;
  for (int e=0; e<estrelas.n(); e++) {
    int j=janelapx(estrelas(e),janelas);
    double d=distancia(estrelas(e),janelas(j));
    soma += d;
  }
  return soma/estrelas.n();
}

void kmeans3(VETOR<PONTOI2>& estrelas, VETOR<PONTOI2>& janelas)
// Para cada janela, calcula as estrelas mais proximas, e calcula a soma vetorial das suas
posicoes
// Move cada janela para o centro da soma vetorial
{ VETOR<int> janpx=janelapx(estrelas,janelas); // indice de janela mais proxima para cada
estrela
```

```
  VETOR<PONTOI2> soma(janelas.n(),PONTOI2(0,0)); // para cada janela, a soma vetorial da
  posicao das estrelas
  VETOR<int> nsoma(janelas.n(),0); // para cada janela, o numero de estrelas na sua zona de
  influencia

  for (int e=0; e<estrelas.n(); e++) {
    int j=janpx(e);
    PONTOI2 est=estrelas(e);
    soma(j) = soma(j)+est;
    nsoma(j)++;
  }
  for (int j=0; j<janelas.n(); j++) {
    if (nsoma(j)>0)
      janelas(j) = soma(j) / nsoma(j);
    else // desloca janela para posicao aleatoria
      janelas(j) = PONTOI2( myrand()%maxx, myrand()% maxy);
  }
}

void kmeans2(VETOR<PONTOI2>& estrelas, VETOR<PONTOI2>& janelas)
{
  double melhor=distanciamedia(estrelas,janelas);
  VETOR<PONTOI2> melhoresjanelas=janelas;

  VETOR<PONTOI2> janelasant;
  do {
    janelasant=janelas;
    kmeans3(estrelas,janelas);
    double d=distanciamedia(estrelas,janelas);
    if (d<melhor) { melhor=d; melhoresjanelas=janelas; }
    //printf("melhor=%f dist=%f\n",melhor,d);
  } while (!(janelas==janelasant));

  janelas=melhoresjanelas;
}

void kmeans(int argc, char** argv)
{ if (argc!=5) {
    printf("kMeans: Gera janelas aleatoriamente e otimiza com kMeans\n");
    printf("kmeans estrela.txt janela.txt semente njanelas\n");
    erro("Erro: Numero de argumentos invalido");
  }

  int semente;
  if (sscanf(argv[3],"%d",&semente)!=1) erro("Erro: Leitura semente");

  int njanelas;
  if (sscanf(argv[4],"%d",&njanelas)!=1) erro("Erro: Leitura njanelas");

  VETOR<PONTOI2> estrelas; le(estrelas,argv[1]);

  VETOR<PONTOI2> janelas;
  uniform2(janelas,njanelas);
  //aleat2(janelas,semente,njanelas);

  kmeans2(estrelas,janelas);
  //printf("Maior=%f\n",maiordistancia(estrelas,janelas));
  //criaimagem1(estrelas,janelas);
  imp(janelas,argv[2]);
```

```
}

void nkmeans(int argc, char** argv)
{ if (argc!=6) {
    printf("nkmeans: Gera n sets de janelas aleat, otimiza com kmeans e escolhe o melhor\n");
    printf("nkmeans estrela.txt janela.txt njanelas semente nsemente\n");
    erro("Erro: Numero de argumentos invalido");
  }

  int njanelas;
  if (sscanf(argv[3],"%d",&njanelas)!=1) erro("Erro: Leitura njanelas");
  int semente;
  if (sscanf(argv[4],"%d",&semente)!=1) erro("Erro: Leitura semente");
  int nsemente;
  if (sscanf(argv[5],"%d",&nsemente)!=1) erro("Erro: Leitura nsemente");

  VETOR<PONTOI2> estrelas; le(estrelas,argv[1]);

  VETOR<PONTOI2> janelas;
  VETOR<PONTOI2> melhoresjanelas;
  double melhor=infinito;

  for (int i=semente; i<semente+nsemente; i++) {
    printf("%8d %10g\r",i-semente,melhor);
    aleat2(janelas,i,njanelas);
    kmeans2(estrelas,janelas);
    double d=maiordistancia(estrelas,janelas);
    if (d<melhor) { melhor=d; melhoresjanelas=janelas; }
  }
  printf("\n");
  imp(melhoresjanelas,argv[2]);
}

//<<<<<<<<<<<<<<<<<<<<<<<< maxdist

void maxdist(int argc, char** argv)
{ if (argc!=3) {
    printf("MaxDist estrela.txt janela.txt\n");
    erro("Erro: Numero de argumentos invalido");
  }

  VETOR<PONTOI2> estrelas;
  le(estrelas,argv[1]);

  VETOR<PONTOI2> janelas;
  le(janelas,argv[2]);

  double maior=maiordistancia(estrelas,janelas);
  printf("Maior distancia=%f\n",maior);
  criaimagem1(estrelas,janelas);
  cvDestroyAllWindows();
}

//<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

struct DI2 { DBL v; int l,c; };
struct DI2menor {
  bool operator() (const DI2& x, const DI2& y) const
    { return (x.v<y.v); }
```

```
};
typedef priority_queue<DI2,vector<DI2>,DI2menor> DI2DEC; // Ordena em ordem decrescente
class KMAXD {
 public:
  DI2DEC pq; // Heap ou priority-queue
  vector<PONTOI2> v; // Lista de elementos retirados de pq. Distancia entre qualquer par
  deles e' maior que d.
  void remove_fim();
  void insere(IMGDBL& a); // Insere todos os pixels a(l,c)>0 em heap pq
  PONTOI2 retira(double d); // Retira elementos do heap pq ate achar um cuja distancia a
  todos os elementos de v seja maior que d.
                            // Caso nao consiga, devolve (-1,-1)
};
struct DI2maior {
  bool operator() (const DI2& x, const DI2& y) const
    { return (x.v>y.v); }
};
typedef priority_queue<DI2,vector<DI2>,DI2maior> DI2INC; // Ordena em ordem crescente

vector<PONTOI2> kmax(IMGDBL& a, int k, double d=0.0);
// Acha k maximos que estao separados por mais de d pixels
// Elementos <=0.0 nao sao levados em conta, e portanto a saida pode ter menos que k
elementos.
vector<PONTOI2> kmin(IMGDBL& a, int k, double d=0.0);
// Acha k minimos que estao separados por mais de d pixels
// Nota: pixels >=1.0 nao sao levados em conta, e portanto a saida pode ter menos que k
elementos.

void KMAXD::remove_fim()
{ if (v.size()>0) v.pop_back(); }

void KMAXD::insere(IMGDBL& a) // Insere todos os pixels a(l,c)>0 em heap pq
{ DI2 f;
  for (int l=0; l<a.nl(); l++)
    for (int c=0; c<a.nc(); c++) {
      if (a.atf(l,c)>0.0) {
        f.v=a.atf(l,c); f.l=l; f.c=c;
        pq.push(f);
      }
    }
}

PONTOI2 KMAXD::retira(double d) // Retira elementos do heap pq ate achar um cuja distancia a
todos os elementos de v seja maior que d.
                                // Caso nao consiga, devolve (-1,-1)
{
  PONTOI2 p;
  bool coloca=false;
  while (!coloca && !pq.empty()) {
    DI2 f=pq.top(); pq.pop();
    p=PONTOI2(f.l,f.c);
    coloca=true;
    for (unsigned j=0; j<v.size(); j++)
      if (distancia(v[j],p)<=d) {
        coloca=false; break;
      }
    if (coloca) v.push_back(p);
  }
  if (pq.empty()) { return PONTOI2(-1,-1); }
```

```
  else return p;
}

vector<PONTOI2> kmax(IMGDBL& a, int k, double d)
{ // Nota: pixels <=0.0 nao sao levados em conta => retirei esta condicao
  DI2DEC pq;
  { DI2 f;
    for (int l=0; l<a.nl(); l++)
      for (int c=0; c<a.nc(); c++) {
        //if (a.atf(l,c)>0.0) {
          f.v=a.atf(l,c); f.l=l; f.c=c;
          pq.push(f);
        //}
      }
  }

  vector<PONTOI2> psai;

  if (d>0.0) {
    if (!pq.empty()) {
      for (int i=0; i<k; i++) {
        bool coloca=false;
        while (!coloca) {
          DI2 f=pq.top();
          //printf("v=%f l=%d c=%d\n",f.v,f.l,f.c);
          pq.pop();
          PONTOI2 p(f.l,f.c);
          coloca=true;
          for (unsigned j=0; j<psai.size(); j++)
            if (distancia(psai[j],p)<=d) {
              coloca=false;
              break;
            }
          if (coloca) psai.push_back(p);
          if (pq.empty()) goto saida;
        }
      }
    }
  } else {
    if (!pq.empty()) {
      for (int i=0; i<k; i++) {
        DI2 f=pq.top();
        pq.pop();
        PONTOI2 p(f.l,f.c);
        psai.push_back(p);
        if (pq.empty()) goto saida;
      }
    }
  }

  saida:
  return psai;
}

vector<PONTOI2> kmin(IMGDBL& a, int k, double d)
{ // Nota: pixels >=1.0 nao sao levados em conta => retirei esta condicao
  DI2INC pq;
  { DI2 f;
    for (int l=0; l<a.nl(); l++)
```

```
      for (int c=0; c<a.nc(); c++) {
        if (a.atf(l,c)<infinito) {
          f.v=a.atf(l,c); f.l=l; f.c=c;
          pq.push(f);
        }
      }
  }

  vector<PONTOI2> psai;

  if (d>0.0) {
    if (!pq.empty()) {
      for (int i=0; i<k; i++) {
        bool coloca=false;
        while (!coloca) {
          DI2 f=pq.top();
          //printf("v=%f l=%d c=%d\n",f.v,f.l,f.c);
          pq.pop();
          PONTOI2 p(f.l,f.c);
          coloca=true;
          for (unsigned j=0; j<psai.size(); j++)
            if (distancia(psai[j],p)<=d) {
              coloca=false;
              break;
            }
          if (coloca) psai.push_back(p);
          if (pq.empty()) goto saida;
        }
      }
    }
  } else {
    if (!pq.empty()) {
      for (int i=0; i<k; i++) {
        DI2 f=pq.top();
        pq.pop();
        PONTOI2 p(f.l,f.c);
        psai.push_back(p);
        if (pq.empty()) goto saida;
      }
    }
  }

  saida:
  return psai;
}

void mostra(IMGDBL a)
{
  IMGGRY sai(a.nl(),a.nc());
  for (int i=0; i<a.n(); i++) {
    double temp=255.0*a(i)/2000;
    sai(i)=double2G(temp);
  }
  pvMostra(sai);
}

void kmin(int argc, char** argv)
{ if (argc!=5 && argc!=9) {
    printf("kmin ent.img sai.ppm k dist [teto chao esq dir]\n");
```

```
  printf("kmin ent.img sai.txt k dist [teto chao esq dir]\n");
  printf("  Acha k minimos que estao separados por mais de d pixels por heap\n");
  printf("  ent.img e' IMGDBL (pode ser grayscale ou IMGFLT)\n");
  printf("  Os pixels com valor>=infinito sao desprezados\n");
  printf("  teto chao esq dir: numero de linhas/colunas a serem desprezadas\n");
  erro("Erro: Numero de argumentos invalido");
}

IMGDBL ent; le(ent,argv[1]);
//mostra(ent);

int k;
if (sscanf(argv[3],"%d",&k)!=1) erro("Erro: Leitura k");
if (k<=0) erro("Erro: k<=0");

double dist;
if (sscanf(argv[4],"%lf",&dist)!=1) erro("Erro: Leitura dist");
if (dist<0.0) erro("Erro: dist<0.0");

int teto=0;
int chao=0;
int esq=0;
int dir=0;
if (argc==9) {
  ConvArg(teto,argv[5]);
  ConvArg(chao,argv[6]);
  ConvArg(esq, argv[7]);
  ConvArg(dir, argv[8]);
}

int tamjan=10;
IMGDBL a(ent.nl(),ent.nc()); a.lc()=a.nl()-1; a.cc()=0; a.backg()=0.0;
for (int x=0; x<a.nc()-tamjan; x++)
  for (int y=0; y<a.nl()-tamjan; y++) {
    double soma=0.0;
    for (int x2=0; x2<tamjan; x2++)
      for (int y2=0; y2<tamjan; y2++)
        soma += ent.atN(x+x2,y+y2);
    a.atN(x,y) = soma / (tamjan*tamjan);
  }

for (int y=0; y<a.nl(); y++) {
  for (int x=0; x<esq; x++) a.atN(x,y)=infinito;
  for (int x=a.nc()-dir-tamjan; x<a.nc(); x++) a.atN(x,y)=infinito;
}
for (int x=0; x<a.nc(); x++) {
  for (int y=0; y<chao; y++) a.atN(x,y)=infinito;
  for (int y=a.nl()-teto-tamjan; y<a.nl(); y++) a.atN(x,y)=infinito;
}
//mostra(a);

vector<PONTOI2> v=kmin(a,k,dist);
printf("Achei %d minimos\n",v.size());

if (sufixo(argv[2])=="txt") {
  VETOR<PONTOI2> w(v.size());
  for (int i=0; i<w.n(); i++) { w(i).x()=v[i].c(); w(i).y()=a.nl()-1-v[i].l(); }
  imp(w,argv[2]);
} else {
```

```
    IMGCOR sai(ent.nl(),ent.nc());
    for (int i=0; i<ent.n(); i++) {
      double temp=255.0*ent(i)/2000;
      BYTE b=double2G(temp);
      sai(i)=COR(b,b,b);
    }

    VETOR<PONTOI2> w(v.size());
    for (int i=0; i<w.n(); i++) { w(i).x()=v[i].c(); w(i).y()=a.nl()-1-v[i].l(); }
    for (int i=0; i<w.n(); i++) {
      int xi=w(i).x(); int yi=w(i).y();

      int l=sai.nl()-1-yi; int c=xi;
      char st[256]; sprintf(st,"%d",i); puttxt(sai,l,c,st,COR(0,255,255));

      for (int x=0; x<tamjan; x++) {
        sai.atN(xi+x,yi)=COR(255,255,0);
        sai.atN(xi+x,yi+tamjan-1)=COR(255,255,0);
      }
      for (int y=0; y<tamjan; y++) {
        sai.atN(xi,  yi+y)=COR(255,255,0);
        sai.atN(xi+tamjan-1,yi+y)=COR(255,255,0);
      }
    }
    //pvMostra(sai);
    imp(sai,argv[2]);
  }

  for (int i=0; i<int(v.size()); i++) {
    int x=v[i].c(); int y=a.nl()-1-v[i].l();
    printf("i=%3d x=%4d y=%4d a(x,y)=%16f\n",i,x,y,a.atN(x,y));
  }
}

//<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< main <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
const char about[]=
"< Kbox: Alocacao de janelas para fundo do ceu, versao 1.05>\n"
"_______________________________________________________________________________\n";

const char help[]=
"Programas:\n"
"Estrela  - Gera estrelas aleatoriamente\n"
"Uniform  - Gera janelas uniformemente\n"
"Aleat    - Gera janelas aleatoriamente\n"
"nAleat   - Gera n sets de janelas aleatoriamente e escolhe o melhor\n"
"kBB      - Gera janelas aleatoriamente e otimiza com kBB\n"
"nkBB     - Gera n sets de janelas aleat, otimiza com kBB e escolhe o melhor\n"
"kMeans   - Gera janelas aleatoriamente e otimiza com kMeans\n"
"nkMeans  - Gera n sets de janelas aleat, otimiza com kMeans e escolhe o melhor\n"
"MaxDist  - Calcula estrela mais distante de janela\n"
"kmin     - Aloca k janelas em regioes mais escuras\n"
"...............................................................................\n";

int main(int argc, char** argv)
{
  if (argc<2) {
    printf(about);
    printf(help);
    erro("Erro: Numero de argumentos invalido");
```

```
  } else {
    int t1=centseg();
    string comando=strlwr(argv[1]);
    if      (comando=="estrela")  estrela(argc-1,&argv[1]);
    else if (comando=="uniform")  uniform(argc-1,&argv[1]);
    else if (comando=="aleat")    aleat(argc-1,&argv[1]);
    else if (comando=="naleat")   naleat(argc-1,&argv[1]);
    else if (comando=="kbb")      kbb(argc-1,&argv[1]);
    else if (comando=="nkbb")     nkbb(argc-1,&argv[1]);
    else if (comando=="kmeans")   kmeans(argc-1,&argv[1]);
    else if (comando=="nkmeans")  nkmeans(argc-1,&argv[1]);
    else if (comando=="maxdist")  maxdist(argc-1,&argv[1]);
    else if (comando=="kmin")     kmin(argc-1,&argv[1]);
    else erro("Erro: Programa inexistente ",comando.c_str());
    t1=centseg()-t1;
    printf("Tempo gasto=%d.%02ds\n",t1/100,t1%100);
  }
}
```