6
Referências

[1] WILLIAM D. CALLISTER, JR. (1994) Materials science and engineering: An Introduction. 3 ed. Canadá: John Wiley & Sons,.

[2] HULL D. (1985) An Introduction to composite materials. New York: Cambridge University Press,

[3] Salvatore Torquato (2002) Random heterogeneous materials: microstructure and macroscopic properties. New York, NY: Springer-Verlag;

[4] Jeremy W. Leggoe, James B. Riggs. (2006) Nth-nearest neighbor statistics for three-dimensional equilibrium arrays of monodisperse spheres. Materials science & engineering A, 426, 289-297.

[5] Cooper DW. (1988);Phys Rev 38:522.

[6] Louis P, Gokhale AM. (1996) Acta Mater;44:1519.

[7] Ghosh S, Nowak Z, Lee K. (1997) Acta Mater;45:2215.

[8] Yang S, Tewari A, Gokhale AM. (1997) Acta Mater;45:3059.

[9] Rintoul MD, Torquato S. J Colloid (1997) Interf Sci;186:467.

[10] Jeni M, Kukuchi M. (1998) Acta Mater;46:3125.

[11] Cule D, Torquato S. (1999) J Appl Phys;86:3428.

[12] Li M, Ghosh S, Richmond O, Weiland H, Rouns TN. (1999) Mater Sci Eng;266:221.

[13] Manwart C, Hilfer R. (2000) Phys Rev E;62:893.

[14] Lepinoux JL, Estrin Y. (2000) Acta Mater;48:4337.

[15] Yang N, Boselli J, Sinclair I. (2001) J Microsc;201:189.

[16] Sheehan N, Torquato S. (2001) J Appl Phys;89:53.

[17] Shan Z, Gokhale AM. (2004) Int J Plasticity;20:1347.

[18] Saylor DM, Fridy J, El-Dasher BS, Jung KY. (2004) Metall Mater Trans A;20:1370.

[19] Tewari A, Gokhale AM. Mater Sci Eng A (2004);35:332.

[20] Jefferson G, Garmestani H, Tannenbaum R, Gokhale AM, Tadd E. (2005) Int J Plasticity;21:185.

[21] Tscheschel A, Lacayo J, Stoyan D. (2005) J Microsc;217:75.

[22] Tewari A, Gokhale AM. (2005) Mater Sci Eng A; 396:22.

[23] Torquato S. (2002) Random heterogeneous materials: microstructure and macroscopic properties. New York, NY: Springer-Verlag.

[24] Sahimi M. (2003) Heterogeneous materials: I. Linear transport and optical properties. New York, NY: Springer-Verlag.

[25] Matsumoto M, Nishimura T. (1998) ACM Trans Model Comput Simulat;1:3.

[26] Metropolis M, Rosenbluth AW, Rosenbluth MN, Teller E, Teller J. (1953) J Chem Phys; 21.

[27] Serra J. (1989) J Microsc; 156:124.

[28] Adler RJ. (1981)The geometry of random fields. New York, NY: Wiley.

[29] Yeong CLY, Torquato S. (1998)Phys Rev E; 58:224.

[30] Lotwick HW. J Stat (1982) Comput Simulat;15:295.

[31] Everett RK, Chu JH. J (1992) Comp Mater;27:3329.

[32] Manwart C, Hilfer R. (1999) Phys Rev E;59:5596

[33] N. Yang. J. Boselli & I. Sinclair (2001) Simulation and quantitative assessment of homogeneous and inhomogeneus particle distributions in particulate metal matrix composites. Journal of Microscopy, 201, 189-200

[34] N. Yang, J. Boselli, I. Sinclair, (2000) Mater. Sci. Technol. 16 797

[35] Susagna F., Yotte S., Riss J. Breysse D., Ghosh S. (2000) Covariance and spatial distribution of particles in metal matrix composites. Proceedings of the 6th International Conference on Sterology and image analysis in material science (STERMAT) Poland: Fotobit Design; 200. P. 397-402

[36] A.Swideska-Sroda, T. Wejrzanowski, K.J. Kurzydlowski, Jw. Wyrzykowski (2003) Quantitative analysis of Al_2O_3 particles in $Al_3Ti/Al_2O_3/Al$ composites

[37] Wray, P.J., Richmond, O. & Morrison, H.L. (1983) Use of the Dirichlet "tessellation" for characterising and modelling nonregular dispersions of second-phase particles. Metallography, 16, 39-58.

[38] Spitzig, W.A., Kelly, J.F. & Richmond, O. (1985) Quantitative characterisation of second-phase populations. Metallography, 18, 235-261.

[39] Shehata, M.T. & Boyd, J.D. (1988) Measurement of spatial distribution of inclusions. Inclusions and Their Influence on Material Behaviour (ed. by R. Rungla), pp. 123-131. ASM International, Ohio

[40] Russ, J.C. & Russ, J.C. (1989) Adjacency measurements in "tessellation"s. J. Comput. Assist. Microsc. 1, 217-247.

[41] Bertram, M. & Wendrock, H. (1996) Characterisation of planar local arrangement by means of the Delaunay neighbourhood. J.Microsc. 181, 45-53.

[42] Boselli, J., Pitcher, P.D., Gregson, P.J. & Sinclair, I. (1998) Quantitative assessment of particle distribution effects on short crack growth in SiCP reinforced Al-alloys. Scripta Mater. 38, 839-844.

[43] Boselli, J., Pitcher, P.D., Gregson, P.J. & Sinclair, I. (1999) Secondary phase distribution analysis via finite body "tessellation". J. Microsc. 195, 104-332.

[44] Burger, G. (1986) The effects of microstructural inhomogeneity on damage accumulation and fracture. PhD Thesis, McMaster University, Canada.

[45] J. Boselli1, P.J. Gregson, I. Sinclair (2004) Quantification of particle distribution effects on fatigue

in an Al–SiCp composite Materials Science and Engineering A 379 72–82

[46] N. Chawla *, J.J. Williams, R. Saha (2002) Mechanical behavior and microstructure characterization of sinter-forged SiC particle reinforced aluminum matrix composites Journal of Light Metals 2 215–227

[47] H. Singh, A.M. Gokhale , Y. Mao, J.E. Spowart (2006) Computer simulations of realistic microstructures of discontinuously reinforced aluminum alloy (DRA) composites Materials Science and Engineering, Acta Materialia 54 2131–2143

[48] J. Boselli, P.D. Pitcher, P.J. Gregson, I. Sinclair, (2001) Mater. Sci. Eng. A 300 335.

[49] N. YANG and I. SINCLAIR (SEPTEMBER 2003) Fatigue Crack Growth in a Particulate TiB2-Reinforced Powder Metallurgy Iron-Based Composite METALLURGICAL AND MATERIALS TRANSACTIONS A VOLUME 34A, 2017

[50] X. Deng N. Chawla (2006) Modeling the effect of particle clustering on the mechanical behavior of SiC particle reinforced Al matrix composites J Mater Sci 41:5731–5734

[51]Z.P. Luo, J.H. Koo (2008) Quantitative study of the dispersion degree in carbon nanofiber/polymer and carbon nanotube/polymer nanocompositesMaterials Letters 62 3493–3496

[52] Z.P. Luo, R.L. Littleton, H. Kim,* and J.H. Koo (2008) Quantifying the layer dispersion degree in polymer layered silicate nanocomposites by quantitative transmission electron microscopy Microsc Microanal 14(Suppl 2), Copyright 2008 Microscopy Society of America

[53]Z. P. Luo & J. H. Koo (*February 2007*) Quantifying the dispersion of mixture microstructures *Journal of Microscopy, Vol. 225, Pt 2, pp. 338–125*

[54] Z. P. Luo (2010) Statistical quantification of the microstructural homogeneity of size and orientation distributions J Mater Sci 45:3228–3241

[55] Gilbert Strang, George G.Fix (1973) An analysis of the finite element method Englewood Cliffs, N. J. :Prentice Hall

[56] Andrew C.E. Reid (2009) Modelling Microstructures with OOF2. *Int. J. Materials and Product Technology, Vol. 35, Nos. 3/4,* 361

[57] A. Zivelonghi, A. Brendel, S. Lindig, S. Nawka, B. Kieback, J.H. You (2011) Microstructure-based analysis of thermal- and mechanical behaviors of W/CuCrZr composites and porous W coating. Journal of Nuclear Materials (Article in press)

[58] Tobias Ziegler, Achim Neubrand , Siddhartha Roy, Alexander Wanner, Romana Piat (2009) Elastic constants of metal/ceramic composites with lamellar microstructures: Finite element modelling and ultrasonic experiments. Composites Science and Technology 69 620–626

[59] Srinivasa R. Bakshi, Akanksha Bhargava, Seyedreza Mohammadizadeh, Arvind Agarwal, Igor Tsukanov (2011) Computational estimation of elastic properties of spark plasma sintered TaC by eshfree and finite element methods. Computational Materials Science 50 2615–2620

[60] Leon L. Mishnaevsky (2005) Jr. Automatic voxel-based generation of 3D microstructural FE models and its application to the damage analysis of composites. Materials Science and Engineering A 407 11–23

[61] Julian Varghese, John Whitcomb, Deepak Goyal and Xiao Dong Tang (2007) Hierarchical Analysis of Woven Composite DCB Specimen. Journal of Composite Materials 41: 931

[62] L. Wang, Y. Wang, X.G. Sun, J.Q. He, Z.Y. Pan, Y. Zhou, P.L. (2011) Wu Influence of pores on the thermal insulation behavior of thermal barrier coatings prepared by atmospheric plasma spray. Materials and Design 32 36–47

[63] Fe´ lice M.J. Naus-Thijssen, Scott E. Johnson, Peter O. Koons (2010) Numerical modeling of crenulation cleavage development:A polymineralic approach. Journal of Structural Geology 32 330–341

[64] A.N. Samantn, B. Dahotre (2008) Multilevel residual stress evaluation in laser surface modified alumina ceramic. Appl. Phys. A 90, 493–499

[65] H. Shen, C.J. Lissenden (2002) 3D finite element analysis of particle-reinforced aluminum. Materials Science and Engineering A338 271-281

7
Apêndice 1 - Códigos computacionais para a construção de ensembles virtuais

.Orientação normalizada.

.Distribuição de tamanho homogênea.

.Distribuição espacial heterogênea (E.Z.)

Entrando os valores dos parámetros

```
Clear[a, c, ct, i, j, n, alpha, k];
pixelsx = 1000;
pixelsy = 1000;
n = 850;
r1 = 3;
r2 = 7;
alpha = 0;
vt = 5;
r01 = 200;
r02 = 100;
k[1] = {200, 200};
k[2] = {800, 200};
k[3] = {200, 800};
k[4] = {800, 800};
k[5] = {500, 500};
ct = Complement[Tuples[{Range[0, pixelsx], Range[0, pixelsy]}],
   Tuples[{Range[r1, pixelsx - r1], Range[r2, pixelsy - r2]}] ];
```

Determinando as coordenadas de cada corpo (aquí demora).

```
For[
  i = 1, i < n + 1, i++,
  a[i] = RandomChoice[Complement[Tuples[{Range[0, pixelsx], Range[0, pixelsy]}], ct]];
  If[
   Intersection[Table[((Cos[alpha]^2/r01^2 + Sin[alpha]^2/r02^2) (First[a[i]] - First[k[v]])^2 - 2 Cos[alpha]
          Sin[alpha] (1/r01^2 - 1/r02^2) (First[a[i]] - First[k[v]]) * (Last[a[i]] - Last[k[v]]) +
         (Sin[alpha]^2/r01^2 + Cos[alpha]^2/r02^2) (Last[a[i]] - Last[k[v]])^2 <=
       1, {v, 1, vt}], {True}] ≠ {}, i = i - 1,
   maxx = First[ a[i]] + 2 * r1 + 1;
   minx = First[ a[i]] - 2 * r1 - 1;
   maxy = Last[ a[i]] + 2 * r2 + 1;
   miny = Last[ a[i]] - 2 * r2 - 1;
   c[i] = Tuples[{Range[minx, maxx], Range[miny, maxy]}];
   ct = Union[ct, c[i]]
  ]];
Beep[]
```

. Gerando a imagen.

```
imagen1 = Graphics[(*{EdgeForm[{Thin,Black}],FaceForm[Black],*)
  Table[Disk[a[h], {r1, r2}], {h, 1, n}], PlotRange → {{0, pixelsx}, {0, pixelsy}},
  PlotRangeClipping → True, ImageSize → {pixelsx, pixelsy}]
```

.Imagem binaria.

```
ColorNegate[Binarize[imagem1, 0.8]]
```

.Distribuição de orientação homogênea.

.Distribuição de tamanho homogênea.

.Distribuição espacial heterogênea (E.Z.).

Entrando os valores dos parámetros.

```
Clear[a, c, ct, i, j, n, theta, rotm, v, h, r1, r2, rmax, rmin];
pixelsx = 1000;
pixelsy = 1000;
n = 10;
r1 = 4;
r2 = 8;
rmax = Max[r1, r2];
theta = 1;
rotm = RotationMatrix[theta];
ct = Complement[Tuples[{Range[0, pixelsx], Range[0, pixelsy]}],
   Tuples[{Range[rmax, pixelsx - rmax], Range[rmax, pixelsy - rmax]}] ];
c[0] = {};
alpha = 0;
vt = 5;
r01 = 200;
r02 = 100;
k[1] = {200, 200};
k[2] = {800, 200};
k[3] = {200, 800};
k[4] = {800, 800};
k[5] = {500, 500};
```

Determinando as coordenadas de cada corpo (aquí demora).

```
Timing[
 For[
  i = 1, i < n + 1, i++,
  a[i] = RandomChoice[Complement[Tuples[{Range[0, pixelsx], Range[0, pixelsy]}], ct]];
  If[
   Intersection[Table[(Cos[alpha]^2/r01^2 + Sin[alpha]^2/r02^2) (First[a[i]] - First[k[v]])^2 - 2 Cos[alpha]
         Sin[alpha] (1/r01^2 - 1/r02^2) (First[a[i]] - First[k[v]]) * (Last[a[i]] - Last[k[v]]) +
       (Sin[alpha]^2/r01^2 + Cos[alpha]^2/r02^2) (Last[a[i]] - Last[k[v]])^2 <=
     1, {v, 1, vt}], {True}] ≠ {}, i = i - 1,
   imagem = Binarize[Graphics[Table[Rotate[Disk[a[j], {r1, r2}], theta], {j, 1, i}],
      PlotRange → {{0, pixelsx}, {0, pixelsy}},
      PlotRangeClipping → True, ImageSize → {pixelsx, pixelsy}]];
   ncorps = Max[Flatten[MorphologicalComponents[ColorNegate[imagem]]]];
   If[ncorps == i,
    maxx = First[ a[i]] + r1 + 1;
    minx = First[ a[i]] - r1 - 1;
    maxy = Last[ a[i]] + r2 + 1;
    miny = Last[ a[i]] - r2 - 1;
    c[i] = Tuples[{Range[minx, maxx], Range[miny, maxy]}];
    c[i] = Table[
      Round[Plus[Dot[rotm, Plus[Part[c[i], v], -a[i]]], a[i]]], {v, 1, Length[c[i]]}];
    ct = Union[ct, c[i]], i = i - 1
   ]
  ]; Print[i]
 ];
 Beep[]
]
```

. Gerando a imagen.

```
imagem1 = Graphics[(*{EdgeForm[{Thin,Black}],FaceForm[Black],*)
  Table[Rotate[Disk[a[h], {r1, r2}], theta], {h, 1, n}],
  PlotRange → {{0, pixelsx}, {0, pixelsy}},
  PlotRangeClipping → True, ImageSize → {pixelsx, pixelsy}]
```

.Imagem binaria.

```
ColorNegate[Binarize[imagem1, 0.8]]
```

.Distribuição de orientação homogênea.

.Distribuição de tamanho homogênea.

.Distribuição espacial heterogênea.

Entrando os valores dos parámetros.

```
Clear[a, c, ct, i, j, n, theta, rotm, v, h, r1, r2, rmax, rmin];
pixelsx = 1000;
pixelsy = 1000;
n = 10;
r1 = 4;
r2 = 8;
rmax = Max[r1, r2];
theta = 1;
rotm = RotationMatrix[theta];
vt = 5;
k[1] = {200, 200};
k[2] = {800, 200};
k[3] = {200, 800};
k[4] = {800, 800};
k[5] = {500, 500};
sigma = {70, 30};
ct = Complement[Tuples[{Range[0 - First[sigma], pixelsx + First[sigma]],
      Range[0 - Last[sigma], pixelsy + Last[sigma]]}],
   Tuples[{Range[rmax + 1, pixelsx - rmax - 1], Range[rmax + 1, pixelsy - rmax - 1]}] ];
c[
   0] =
  {};
```

Determinando as coordenadas de cada corpo (aquí demora).

```
Timing[
 For[
  i = 1, i < n + 1, i++,
  sementes = Table[{Round[Random[NormalDistribution[First[k[v]], First[sigma]]]],
     Round[Random[NormalDistribution[Last[k[v]], Last[sigma]]]]}, {v, 1, vt}];
  sementesmaduras = Complement[sementes, ct];
  If[sementesmaduras == {}, i = i - 1,
   a[i] = RandomChoice[sementesmaduras];
   imagem = Binarize[Graphics[Table[Rotate[Disk[a[j], {r1, r2}], theta], {j, 1, i}],
      PlotRange → {{0, pixelsx}, {0, pixelsy}},
      PlotRangeClipping → True, ImageSize → {pixelsx, pixelsy}]];
   ncorps = Max[Flatten[MorphologicalComponents[ColorNegate[imagem]]]];
   If[ncorps == i,
    maxx = First[ a[i]] + r1 + 1;
    minx = First[ a[i]] - r1 - 1;
    maxy = Last[ a[i]] + r2 + 1;
    miny = Last[ a[i]] - r2 - 1;
    c[i] = Tuples[{Range[minx, maxx], Range[miny, maxy]}];
    c[i] = Table[
      Round[Plus[Dot[rotm, Plus[Part[c[i], v], -a[i]]], a[i]]], {v, 1, Length[c[i]]}];
    ct = Union[ct, c[i]], i = i - 1]
  ]; Print[i]
 ];
 Beep[]
]
```

. Gerando a imagen.

```
imagem1 = Graphics[(*{EdgeForm[{Thin,Black}],FaceForm[Black],*)
  Table[Rotate[Disk[a[h], {r1, r2}], theta], {h, 1, n}],
  PlotRange → {{0, pixelsx}, {0, pixelsy}},
  PlotRangeClipping → True, ImageSize → {pixelsx, pixelsy}]
```

.Imagem binaria.

```
ColorNegate[Binarize[imagem1, 0.8]]
```

.Distribuição de orientação homogênea.

.Distribuição de tamanho heterogênea.

.Distribuição espacial heterogênea (E.Z).

Entrando os valores dos parámetros

```
Clear[a, c, ct, i, j, n, theta, rotm, v, h, r1, r2, rmax, rmin];
pixelsx = 1000;
pixelsy = 1000;
n = 10;
r1max = 8;
r1min = 3;
r2max = 4;
r2min = 1.5;
rmax = Max[r2max, r1max];
theta = 1;
rotm = RotationMatrix[theta];
ct = Complement[Tuples[{Range[0, pixelsx], Range[0, pixelsy]}],
   Tuples[{Range[rmax, pixelsx - rmax], Range[rmax, pixelsy - rmax]}] ];
c[0] = {};
alpha = 0;
vt = 5;
r01 = 200;
r02 = 100;
k[1] = {200, 200};
k[2] = {800, 200};
k[3] = {200, 800};
k[4] = {800, 800};
k[5] = {500, 500};
```

Determinando as coordenadas de cada corpo (aquí demora).

```
Timing[
 For[
   i = 1, i < n + 1, i++,
   a[i] = RandomChoice[Complement[Tuples[{Range[0, pixelsx], Range[0, pixelsy]}], ct]];
   If[Intersection[
       Table[((Cos[alpha]^2/r01^2 + Sin[alpha]^2/r02^2) (First[a[i]] - First[k[v]])^2 - 2 Cos[alpha]
           Sin[alpha] (1/r01^2 - 1/r02^2) (First[a[i]] - First[k[v]]) * (Last[a[i]] - Last[k[v]]) +
          (Sin[alpha]^2/r01^2 + Cos[alpha]^2/r02^2) (Last[a[i]] - Last[k[v]])^2 <=
         1, {v, 1, vt}], {True}] ≠ {}, i = i - 1,
    r1[i] = RandomChoice[Range[r1min, r1max]];
    r2[i] = RandomChoice[Range[r2min, r2max]];
    imagem = Binarize[Graphics[Table[Rotate[Disk[a[j], {r1[j], r2[j]}], theta], {j, 1, i}],
       PlotRange → {{0, pixelsx}, {0, pixelsy}},
       PlotRangeClipping → True, ImageSize → {pixelsx, pixelsy}]];
    ncorps = Max[Flatten[MorphologicalComponents[ColorNegate[imagem]]]];
    If[ncorps == i,
     maxx = First[ a[i]] + r1[i] + 1;
     minx = First[ a[i]] - r1[i] - 1;
     maxy = Last[ a[i]] + r2[i] + 1;
     miny = Last[ a[i]] - r2[i] - 1;
     c[i] = Tuples[{Range[minx, maxx], Range[miny, maxy]}];
     c[i] = Table[
       Round[Plus[Dot[rotm, Plus[Part[c[i], v], -a[i]]], a[i]]], {v, 1, Length[c[i]]}];
     ct = Union[ct, c[i]], i = i - 1]
   ]; Print[i]
  ]
  Beep[]
]
```

. Gerando a imagen.

```
imagem1 = Graphics[(*{EdgeForm[{Thin,Black}],FaceForm[Black],*)
  Table[Rotate[Disk[a[h], {r1[h], r2[h]}], theta], {h, 1, n}],
  PlotRange → {{0, pixelsx}, {0, pixelsy}},
  PlotRangeClipping → True, ImageSize → {pixelsx, pixelsy}]
```

.Imagem binaria.

```
ColorNegate[Binarize[imagem1, 0.8]]
```

.Distribuição de orientação homogênea.

.Distribuição de tamanho heterogênea.

.Distribuição espacial homogênea.

Entrando os valores dos parámetros

```
Clear[a, c, ct, i, j, n, theta, rotm, v, h, r1, r2, rmax, rmin];
pixelsx = 1000;
pixelsy = 1000;
n = 15;
r1max = 8;
r1min = 3;
r2max = 4;
r2min = 1.5;
theta = 1;
rotm = RotationMatrix[theta];
vt = 5;
k[1] = {200, 200};
k[2] = {800, 200};
k[3] = {200, 800};
k[4] = {800, 800};
k[5] = {500, 500};
rmax = Max[r1max, r2max];
sigma = {70, 30};
ct = Complement[Tuples[{Range[0 - First[sigma], pixelsx + First[sigma]],
      Range[0 - Last[sigma], pixelsy + Last[sigma]]}],
   Tuples[{Range[rmax + 1, pixelsx - rmax - 1], Range[rmax + 1, pixelsy - rmax - 1]}] ];
c[
   0] =
  {};
```

Determinando as coordenadas de cada corpo (aquí demora).

```
Timing[
 For[
   i = 1, i < n + 1, i++,
   sementes = Table[{Round[Random[NormalDistribution[First[k[v]], First[sigma]]]],
      Round[Random[NormalDistribution[Last[k[v]], Last[sigma]]]]}, {v, 1, vt}];
   sementesmaduras = Complement[sementes, ct];
   If[sementesmaduras == {}, i = i - 1,
    a[i] = RandomChoice[sementesmaduras];
    r1[i] = RandomChoice[Range[r1min, r1max]];
    r2[i] = RandomChoice[Range[r2min, r2max]];
    imagem = Binarize[Graphics[Table[Rotate[Disk[a[j], {r1[j], r2[j]}], theta], {j, 1, i}],
       PlotRange → {{0, pixelsx}, {0, pixelsy}},
       PlotRangeClipping → True, ImageSize → {pixelsx, pixelsy}]];
    ncorps = Max[Flatten[MorphologicalComponents[ColorNegate[imagem]]]];
    If[ncorps == i,
     maxx = First[ a[i]] + r1[i] + 1;
     minx = First[ a[i]] - r1[i] - 1;
     maxy = Last[ a[i]] + r2[i] + 1;
     miny = Last[ a[i]] - r2[i] - 1;
     c[i] = Tuples[{Range[minx, maxx], Range[miny, maxy]}];
     c[i] = Table[
       Round[Plus[Dot[rotm, Plus[Part[c[i], v], -a[i]]], a[i]]], {v, 1, Length[c[i]]}];
     ct = Union[ct, c[i]], i = i - 1
    ]
   ]; Print[i]
  ]
  Beep[]
]
```

. Gerando a imagen.

```
imagem1 = Graphics[(*{EdgeForm[{Thin,Black}],FaceForm[Black],*)
  Table[Rotate[Disk[a[h], {r1[h], r2[h]}], theta], {h, 1, n}],
  PlotRange → {{0, pixelsx}, {0, pixelsy}},
  PlotRangeClipping → True, ImageSize → {pixelsx, pixelsy}]
```

.Imagem binaria.

```
ColorNegate[Binarize[imagem1, 0.8]]
```

Distribuição de orientação heterogênea.

Distribuição de tamanho homogênea.

Distribuição espacial heterogênea (E.Z).

Entrando os valores dos parámetros.

```
Clear[a, c, ct, i, j, n, theta, rotm, v, h, r1, r2, rmax, rmin];
n = 10;
pixelsx = 1000;
pixelsy = 1000;
r1 = 8;
r2 = 4;
rmax = Max[r2, r1];
ct = Complement[Tuples[{Range[0, pixelsx], Range[0, pixelsy]}],
   Tuples[{Range[rmax, pixelsx - rmax], Range[rmax, pixelsy - rmax]}] ];
c[0] = {};
alpha = 0;
vt = 5;
r01 = 200;
r02 = 100;
k[1] = {200, 200};
k[2] = {800, 200};
k[3] = {200, 800};
k[4] = {800, 800};
k[5] = {500, 500};
```

Determinando as coordenadas de cada corpo. (aquí demora)

```
Timing[
 For[
   i = 1, i < n + 1, i++,
   a[i] = RandomChoice[Complement[Tuples[{Range[0, pixelsx], Range[0, pixelsy]}], ct]];
   If[Intersection[
       Table[((Cos[alpha]^2/r01^2 + Sin[alpha]^2/r02^2) (First[a[i]] - First[k[v]])^2 - 2 Cos[alpha]
           Sin[alpha] (1/r01^2 - 1/r02^2) (First[a[i]] - First[k[v]]) * (Last[a[i]] - Last[k[v]]) +
          (Sin[alpha]^2/r01^2 + Cos[alpha]^2/r02^2) (Last[a[i]] - Last[k[v]])^2 <=
         1, {v, 1, vt}], {True}] ≠ {}, i = i - 1,
    theta[i] = Random[Real, {0, Pi}];
    rotm[i] = RotationMatrix[theta[i]];
    imagem = Binarize[Graphics[Table[Rotate[Disk[a[j], {r1, r2}], theta[j]], {j, 1, i}],
       PlotRange → {{0, pixelsx}, {0, pixelsy}},
       PlotRangeClipping → True, ImageSize → {pixelsx, pixelsy}]];
    ncorps = Max[Flatten[MorphologicalComponents[ColorNegate[imagem]]]];
    If[ncorps == i,
     maxx = First[ a[i]] + r1 + 1;
     minx = First[ a[i]] - r1 - 1;
     maxy = Last[ a[i]] + r2 + 1;
     miny = Last[ a[i]] - r2 - 1;
     c[i] = Tuples[{Range[minx, maxx], Range[miny, maxy]}];
     c[i] = Table[
       Round[Plus[Dot[rotm[i], Plus[Part[c[i], v], -a[i]]], a[i]]], {v, 1, Length[c[i]]}];
     ct = Union[ct, c[i]], i = i - 1
    ]; Print[i]
  ]
  Beep[]
]
```

Gerando a imagen.

```
imagem1 = Graphics[(*{EdgeForm[{Thin,Black}],FaceForm[Black],*)
  Table[Rotate[Disk[a[h], {r1, r2}], theta[h]], {h, 1, n}],
  PlotRange → {{0, pixelsx}, {0, pixelsy}},
  PlotRangeClipping → True, ImageSize → {pixelsx, pixelsy}]
```

Imagem binaria.

```
ColorNegate[Binarize[imagem1, 0.8]]
```

Distribuição de orientação heterogênea.

Distribuição de tamanho homogênea.

Distribuição espacial homogênea.

Entrando os valores dos parámetros.

```
Clear[a, c, ct, i, j, n, cita, rotm, v, h, r1, r2, rmax, rmin];
n = 10;
pixelsx = 1000;
pixelsy = 1000;
r1 = 8;
r2 = 4;
rmax = Max[r2, r1];
vt = 5;
k[1] = {200, 200};
k[2] = {800, 200};
k[3] = {200, 800};
k[4] = {800, 800};
k[5] = {500, 500};
sigma = {70, 30};
ct = Complement[Tuples[{Range[0 - First[sigma], pixelsx + First[sigma]],
      Range[0 - Last[sigma], pixelsy + Last[sigma]]}],
    Tuples[{Range[rmax + 1, pixelsx - rmax - 1], Range[rmax + 1, pixelsy - rmax - 1]}] ];
c[
   0] =
  {};
```

Determinando as coordenadas de cada corpo. (aquí demora)

```
Timing[
 For[
   i = 1, i < n + 1, i++,
   sementes = Table[{Round[Random[NormalDistribution[First[k[v]], First[sigma]]]],
      Round[Random[NormalDistribution[Last[k[v]], Last[sigma]]]]}, {v, 1, vt}];
   sementesmaduras = Complement[sementes, ct];
   If[sementesmaduras ≠ {},
    a[i] = RandomChoice[sementesmaduras];
    cita[i] = Random[Real, {0, Pi}];
    rotm[i] = RotationMatrix[cita[i]];
    maxx = First[ a[i]] + r1 + 1;
    minx = First[ a[i]] - r1 - 1;
    maxy = Last[ a[i]] + r2 + 1;
    miny = Last[ a[i]] - r2 - 1;
    c[i] = Tuples[{Range[minx, maxx], Range[miny, maxy]}];
    c[i] = Table[
      Round[Plus[Dot[rotm[i], Plus[Part[c[i], v], -a[i]]], a[i]]], {v, 1, Length[c[i]]}];
    If[Intersection[c[i], ct] ≠ {}, i = i - 1, ct = Union[ct, c[i]]], i = i - 1
   ]
  ]
  Beep[]
]
```

Gerando a imagen.

```
imagem1 = Graphics[(*{EdgeForm[{Thin,Black}],FaceForm[Black],*)
  Table[Rotate[Disk[a[h], {r1, r2}], cita[h]], {h, 1, n}],
  PlotRange → {{0, pixelsx}, {0, pixelsy}},
  PlotRangeClipping → True, ImageSize → {pixelsx, pixelsy}]
```

Imagem binaria.

```
ColorNegate[Binarize[imagem1, 0.8]]
```

Distribuição de orientação heterogênea.

Distribuição de tamanho homogênea.

Distribuição espacial homogênea.

Entrando os valores dos parámetros.

```
Clear[a, c, ct, i, j, n, theta, rotm, v, h, r1, r2, rmax, rmin];
n = 10;
pixelsx = 1000;
pixelsy = 1000;
r1 = 80;
r2 = 40;
rmax = Max[r2, r1];
ct = Complement[Tuples[{Range[0, pixelsx], Range[0, pixelsy]}],
   Tuples[{Range[rmax, pixelsx - rmax], Range[rmax, pixelsy - rmax]}] ];
c[
   0] =
  {};
```

Determinando as coordenadas de cada corpo. (aquí demora)

```
Timing[
 For[
   i = 1, i < n + 1, i++,
   a[i] = RandomChoice[Complement[Tuples[{Range[0, pixelsx], Range[0, pixelsy]}], ct]];
   theta[i] = Random[Real, {0, Pi}];
   rotm[i] = RotationMatrix[theta[i]];
   imagem = Binarize[Graphics[Table[Rotate[Disk[a[j], {r1, r2}], theta[j]], {j, 1, i}],
      PlotRange → {{0, pixelsx}, {0, pixelsy}},
      PlotRangeClipping → True, ImageSize → {pixelsx, pixelsy}]];
   ncorps = Max[Flatten[MorphologicalComponents[ColorNegate[imagem]]]];
   If[ncorps == i,
    maxx = First[ a[i]] + r1 + 1;
    minx = First[ a[i]] - r1 - 1;
    maxy = Last[ a[i]] + r2 + 1;
    miny = Last[ a[i]] - r2 - 1;
    c[i] = Tuples[{Range[minx, maxx], Range[miny, maxy]}];
    c[i] = Table[
      Round[Plus[Dot[rotm[i], Plus[Part[c[i], v], -a[i]]], a[i]]], {v, 1, Length[c[i]]}];
    ct = Union[ct, c[i]], i = i - 1
   ]; Print[i]
  ]
  Beep[]
]
```

Gerando a imagen.

```
imagem1 = Graphics[(*{EdgeForm[{Thin,Black}],FaceForm[Black],*)
  Table[Rotate[Disk[a[h], {r1, r2}], theta[h]], {h, 1, n}],
  PlotRange → {{0, pixelsx}, {0, pixelsy}},
  PlotRangeClipping → True, ImageSize → {pixelsx, pixelsy}]
```

Imagem binaria.

```
ColorNegate[Binarize[imagem1, 0.8]]
```

.Distribuição de orientação heterogênea.

.Distribuição de tamanho heterogênea.

.Distribuição espacial heterogênea (clusters).

Entrando os valores dos parámetros

```
Clear[a, c, ct, i, j, n, theta, rotm, v, vt, h,
  r1, r2, rmax, rmin, imagem1, sementesmaduras, sementes, v];
n = 1000;
pixelsx = 1000;
pixelsy = 1000;
r1max = 12;
r1min = 5;
r2max = 7;
r2min = 4;
vt = 9;
k[1] = {250, 500};
k[2] = {750, 500};
k[3] = {500, 250};
k[4] = {500, 750};
k[5] = {500, 500};
k[6] = {250, 250};
k[7] = {750, 750};
k[8] = {750, 250};
k[9] = {250, 750};
rmax = Max[r1max, r2max];
sigma = {60, 30};
ct = Complement[Tuples[{Range[0 - First[sigma], pixelsx + First[sigma]],
     Range[0 - Last[sigma], pixelsy + Last[sigma]]}],
   Tuples[{Range[rmax + 1, pixelsx - rmax - 1], Range[rmax + 1, pixelsy - rmax - 1]}] ];
c[
   0] =
  {};
```

Determinando as coordenadas de cada corpo. (aquí demora)

```
Timing[For[
   i = 1, i < n + 1, i++,
   sementes = Table[{Round[Random[NormalDistribution[First[k[v]], First[sigma]]]],
      Round[Random[NormalDistribution[Last[k[v]], Last[sigma]]]]}, {v, 1, vt}];
   sementesmaduras = Complement[sementes, ct];
   If[sementesmaduras == {}, i = i - 1,
    a[i] = RandomChoice[sementesmaduras];
    theta[i] = Random[Real, {0, Pi}];
    rotm[i] = RotationMatrix[theta[i]];
    r1[i] = RandomChoice[Range[r1min, r1max]];
    r2[i] = RandomChoice[Range[r2min, r2max]];
    imagem = Binarize[Graphics[Table[Rotate[Disk[a[j], {r1[j], r2[j]}], theta[j]],
        {j, 1, i}], PlotRange → {{0, pixelsx}, {0, pixelsy}},
       PlotRangeClipping → True, ImageSize → {pixelsx, pixelsy}]];
    ncorps = Max[Flatten[MorphologicalComponents[ColorNegate[imagem]]]];
    If[ncorps == i,
     maxx = First[ a[i]] + r1[i] + 1;
     minx = First[ a[i]] - r1[i] - 1;
     maxy = Last[ a[i]] + r2[i] + 1;
     miny = Last[ a[i]] - r2[i] - 1;
     c[i] = Tuples[{Range[minx, maxx], Range[miny, maxy]}];
     c[i] = Table[
       Round[Plus[Dot[rotm[i], Plus[Part[c[i], v], -a[i]]], a[i]]], {v, 1, Length[c[i]]}];
     ct = Union[ct, c[i]], i = i - 1
    ]
   ];
   Print[i]
  ]
  Beep[]
]
```

. Gerando a imagen.

```
imagem1 = Graphics[(*{EdgeForm[{Thin,Black}],FaceForm[Black],*)
  Table[Rotate[Disk[a[h], {r1[h], r2[h]}], theta[h]], {h, 1, n}],
  PlotRange → {{0, pixelsx}, {0, pixelsy}},
  PlotRangeClipping → True, ImageSize → {pixelsx, pixelsy}]
```

.Imagem binaria.

```
imagbi = ColorNegate[Binarize[imagem1]]
```

.Distribuição de orientação heterogênea.

.Distribuição de tamanho heterogênea.

.Distribuição espacial heterogênea.

Entrando os valores dos parámetros

```
Clear[a, c, ct, i, j, n, theta, rotm, v, h, r1, r2, rmax, rmin];
n = 10;
pixelsx = 1000;
pixelsy = 1000;
r1max = 12;
r1min = 7;
r2max = 10;
r2min = 5;
rmax = Max[r2max, r1max];
alpha = 0;
vt = 5;
r01 = 200;
r02 = 100;
k[1] = {200, 200};
k[2] = {800, 200};
k[3] = {200, 800};
k[4] = {800, 800};
k[5] = {500, 500};
ct = Complement[Tuples[{Range[0, pixelsx], Range[0, pixelsy]}],
   Tuples[{Range[r1max, pixelsx - r1max], Range[r2max, pixelsy - r2max]}] ];
```

Determinando as coordenadas de cada corpo. (aquí demora)

```
Timing[For[
   i = 1, i < n + 1, i++,
   a[i] = RandomChoice[Complement[Tuples[{Range[0, pixelsx], Range[0, pixelsy]}], ct]];
   If[Intersection[
      Table[((Cos[alpha]^2/r01^2 + Sin[alpha]^2/r02^2) (First[a[i]] - First[k[v]])^2 - 2 Cos[alpha]
          Sin[alpha] (1/r01^2 - 1/r02^2) (First[a[i]] - First[k[v]]) * (Last[a[i]] - Last[k[v]]) +
         (Sin[alpha]^2/r01^2 + Cos[alpha]^2/r02^2) (Last[a[i]] - Last[k[v]])^2 <=
       1, {v, 1, vt}], {True}] ≠ {}, i = i - 1,
    theta[i] = Random[Real, {0, Pi}];
    rotm[i] = RotationMatrix[theta[i]];
    r1[i] = RandomChoice[Range[r1min, r1max]];
    r2[i] = RandomChoice[Range[r2min, r2max]];
    imagem = Binarize[Graphics[Table[Rotate[Disk[a[j], {r1[j], r2[j]}], theta[j]],
        {j, 1, i}], PlotRange → {{0, pixelsx}, {0, pixelsy}},
      PlotRangeClipping → True, ImageSize → {pixelsx, pixelsy}]];
    ncorps = Max[Flatten[MorphologicalComponents[ColorNegate[imagem]]]];
    If[ncorps == i,
     maxx = First[ a[i]] + r1[i] + 1;
     minx = First[ a[i]] - r1[i] - 1;
     maxy = Last[ a[i]] + r2[i] + 1;
     miny = Last[ a[i]] - r2[i] - 1;
     c[i] = Tuples[{Range[minx, maxx], Range[miny, maxy]}];
     c[i] = Table[
       Round[Plus[Dot[rotm[i], Plus[Part[c[i], v], -a[i]]], a[i]]], {v, 1, Length[c[i]]}];
     ct = Union[ct, c[i]], i = i - 1
    ]
   ]; Print[i]
  ]
  Beep[]
]
```

. Gerando a imagen.

```
imagem1 = Graphics[(*{EdgeForm[{Thin,Black}],FaceForm[Black],*)
  Table[Rotate[Disk[a[h], {r1[h], r2[h]}], theta[h]], {h, 1, n}],
  PlotRange → {{0, pixelsx}, {0, pixelsy}},
  PlotRangeClipping → True, ImageSize → {pixelsx, pixelsy}]
```

.Imagem binaria.

```
ColorNegate[Binarize[imagem1, 0.8]]
```

.Distribuição de orientação heterogênea.

.Distribuição de tamanho heterogênea.

.Distribuição espacial homogênea.

Entrando os valores dos parámetros

```
Clear[a, c, ct, i, j, n, theta, rotm, v, h, r1, r2, rmax, rmin, imagem, ncorps];
n = 1000;
pixelsx = 1000;
pixelsy = 1000;
r1max = 10;
r1min = 7;
r2max = 6;
r2min = 4;
rmax = Max[r2max, r1max];
ct = Complement[Tuples[{Range[0, pixelsx], Range[0, pixelsy]}],
   Tuples[{Range[rmax, pixelsx - rmax], Range[rmax, pixelsy - rmax]}] ];
c[
   0] =
  {};
```

Determinando as coordenadas de cada corpo. (aquí demora)

```
Timing[For[
   i = 1, i < n + 1, i++,
   a[i] = RandomChoice[Complement[Tuples[{Range[0, pixelsx], Range[0, pixelsy]}], ct]];
   theta[i] = Random[Real, {0, Pi}];
   rotm[i] = RotationMatrix[theta[i]];
   r1[i] = RandomChoice[Range[r1min, r1max]];
   r2[i] = RandomChoice[Range[r2min, r2max]];
   imagem =
    Binarize[Graphics[Table[Rotate[Disk[a[j], {r1[j], r2[j]}], theta[j]], {j, 1, i}],
      PlotRange → {{0, pixelsx}, {0, pixelsy}}, PlotRangeClipping → True,
      ImageSize → {pixelsx, pixelsy}]];
   ncorps = Max[Flatten[MorphologicalComponents[ColorNegate[imagem]]]];
   If[ncorps == i,
     maxx = First[ a[i]] + r1[i] + 1;
     minx = First[ a[i]] - r1[i] - 1;
     maxy = Last[ a[i]] + r2[i] + 1;
     miny = Last[ a[i]] - r2[i] - 1;
     c[i] = Tuples[{Range[minx, maxx], Range[miny, maxy]}];
     c[i] = Table[
       Round[Plus[Dot[rotm[i], Plus[Part[c[i], v], -a[i]]], a[i]]], {v, 1, Length[c[i]]}];
     ct = Union[ct, c[i]], i = i - 1
    ]
    Print[i]
  ]
  Beep[]
]
```

. Gerando a imagen.

```
imagem1 = Graphics[(*{EdgeForm[{Thin,Black}],FaceForm[Black],*)
  Table[Rotate[Disk[a[h], {r1[h], r2[h]}], theta[h]], {h, 1, n}],
  PlotRange → {{0, pixelsx}, {0, pixelsy}},
  PlotRangeClipping → True, ImageSize → {pixelsx, pixelsy}]
```

.Imagem binaria.

```
imagembi = ColorNegate[Binarize[imagem1]]
```

.Orientação normalizada.

.Distribuição de tamanho heterogénea.

.Distribuição espacial heterogênea (E.Z.).

Entrando os valores dos parámetros

```
Clear[a, c, ct, i, j, n, cita, rotm, v, h, r1, r2, rmax, rmin];
n = 10;
pixelsx = 1000;
pixelsy = 1000;
r1max = 8;
r1min = 3;
r2max = 4;
r2min = 1.5;
rmax = Max[r2max, r1max];
ct = Complement[Tuples[{Range[0, pixelsx], Range[0, pixelsy]}],
   Tuples[{Range[rmax, pixelsx - rmax], Range[rmax, pixelsy - rmax]}] ];
c[0] = {};
alpha = 0;
vt = 5;
r01 = 200;
r02 = 100;
k[1] = {200, 200};
k[2] = {800, 200};
k[3] = {200, 800};
k[4] = {800, 800};
k[5] = {500, 500};
```

Determinando as coordenadas de cada corpo (aquí demora).

```
Timing[
 For[
  i = 1, i < n + 1, i++,
  r1[i] = RandomChoice[Range[r1min, r1max]];
  r2[i] = RandomChoice[Range[r2min, r2max]];
  a[i] = RandomChoice[Complement[Tuples[{Range[0, pixelsx], Range[0, pixelsy]}], ct]];
  If[
   Intersection[Table[(Cos[alpha]^2/r01^2 + Sin[alpha]^2/r02^2) (First[a[i]] - First[k[v]])^2 - 2 Cos[alpha]
         Sin[alpha] (1/r01^2 - 1/r02^2) (First[a[i]] - First[k[v]]) * (Last[a[i]] - Last[k[v]]) +
        (Sin[alpha]^2/r01^2 + Cos[alpha]^2/r02^2) (Last[a[i]] - Last[k[v]])^2 <=
      1, {v, 1, vt}], {True}] ≠ {}, i = i - 1,
   maxx = First[ a[i]] + r1[i] + 1;
   minx = First[ a[i]] - r1[i] - 1;
   maxy = Last[ a[i]] + r2[i] + 1;
   miny = Last[ a[i]] - r2[i] - 1;
   c[i] = Tuples[{Range[minx, maxx], Range[miny, maxy]}];
   If[Intersection[c[i], ct] ≠ {}, i = i - 1, ct = Union[ct, c[i]]]
  ]
 ];
 Beep[]
]
```

. Gerando a imagen.

```
imagem1 = Graphics[(*{EdgeForm[{Thin,Black}],FaceForm[Black],*)
  Table[Disk[a[h], {r1[h], r2[h]}], {h, 1, n}], PlotRange → {{0, pixelsx}, {0, pixelsy}},
  PlotRangeClipping → True, ImageSize → {pixelsx, pixelsy}]
```

.Imagem binaria.

```
ColorNegate[Binarize[imagem1, 0.8]]
```

.Orientação normalizada.

.Distribuição de tamanho heterogênea.

.Distribuição espacial heterogênea.

Entrando os valores dos parámetros

```
Clear[a, c, ct, i, j, n, theta, rotm, v, h, r1, r2, rmax, rmin];
n = 10;
pixelsx = 1000;
pixelsy = 1000;
r1max = 8;
r1min = 3;
r2max = 4;
r2min = 1.5;
rmax = Max[r2max, r1max];
vt = 5;
k[1] = {200, 200};
k[2] = {800, 200};
k[3] = {200, 800};
k[4] = {800, 800};
k[5] = {500, 500};
sigma = {70, 30};
ct = Complement[Tuples[{Range[0 - First[sigma], pixelsx + First[sigma]],
      Range[0 - Last[sigma], pixelsy + Last[sigma]]}],
   Tuples[{Range[rmax + 1, pixelsx - rmax - 1], Range[rmax + 1, pixelsy - rmax - 1]}] ];
c[
   0] =
  {};
```

Determinando as coordenadas de cada corpo (aquí demora).

```
Timing[
 For[
  i = 1, i < n + 1, i++,
  sementes = Table[{Round[Random[NormalDistribution[First[k[v]], First[sigma]]]],
     Round[Random[NormalDistribution[Last[k[v]], Last[sigma]]]]}, {v, 1, vt}];
  sementesmaduras = Complement[sementes, ct];
  If[sementesmaduras == {}, i = i - 1,
   a[i] = RandomChoice[sementesmaduras];
   r1[i] = RandomChoice[Range[r1min, r1max]];
   r2[i] = RandomChoice[Range[r2min, r2max]];
   imagem = Binarize[Graphics[
      Table[Disk[a[j], {r1[j], r2[j]}], {j, 1, i}], PlotRange → {{0, pixelsx}, {0, pixelsy}},
      PlotRangeClipping → True, ImageSize → {pixelsx, pixelsy}]];
   ncorps = Max[Flatten[MorphologicalComponents[ColorNegate[imagem]]]];
   If[ncorps == i,
    maxx = First[ a[i]] + r1[i] + 1;
    minx = First[ a[i]] - r1[i] - 1;
    maxy = Last[ a[i]] + r2[i] + 1;
    miny = Last[ a[i]] - r2[i] - 1;
    c[i] = Tuples[{Range[minx, maxx], Range[miny, maxy]}];
    ct = Union[ct, c[i]], i = i - 1]
  ]; Print[i]
 ];
 Beep[]
]
```

```
{14.352, Null}
```

. Gerando a imagen.

```
imagem1 = Graphics[(*{EdgeForm[{Thin,Black}],FaceForm[Black],*)
  Table[Disk[a[h], {r1[h], r2[h]}], {h, 1, n}], PlotRange → {{0, pixelsx}, {0, pixelsy}},
  PlotRangeClipping → True, ImageSize → {pixelsx, pixelsy}]
```

.Imagem binaria.

```
ColorNegate[Binarize[imagem1, 0.8]]
```

.Orientação normalizada.

.Distribuição de tamanho heterogénea.

.Distribuição espacial homogênea.

Entrando os valores dos parámetros

```
Clear[a, c, ct, i, j, n, cita, rotm, v, h, r1, r2, rmax, rmin];
n = 10;
pixelsx = 1000;
pixelsy = 1000;
r1max = 80;
r1min = 30;
r2max = 40;
r2min = 15;
rmax = Max[r2max, r1max];
ct = Complement[Tuples[{Range[0, pixelsx], Range[0, pixelsy]}],
   Tuples[{Range[rmax, pixelsx - rmax], Range[rmax, pixelsy - rmax]}] ];
c[
   0] =
  {};
```

Determinando as coordenadas de cada corpo (aquí demora).

```
Timing[
 For[
  i = 1, i < n + 1, i++,
  r1[i] = RandomChoice[Range[r1min, r1max]];
  r2[i] = RandomChoice[Range[r2min, r2max]];
  a[i] = RandomChoice[Complement[Tuples[{Range[0, pixelsx], Range[0, pixelsy]}], ct]];
  imagem = Binarize[Graphics[
     Table[Disk[a[j], {r1[j], r2[j]}], {j, 1, i}], PlotRange → {{0, pixelsx}, {0, pixelsy}},
     PlotRangeClipping → True,  ImageSize → {pixelsx, pixelsy}]];
  ncorps = Max[Flatten[MorphologicalComponents[ColorNegate[imagem]]]];
  If[ncorps == i,
   maxx = First[ a[i]] + r1[i] + 1;
   minx = First[ a[i]] - r1[i] - 1;
   maxy = Last[ a[i]] + r2[i] + 1;
   miny = Last[ a[i]] - r2[i] - 1;
   c[i] = Tuples[{Range[minx, maxx], Range[miny, maxy]}];
   ct = Union[ct, c[i]], i = i - 1
  ]; Print[i]
 ];
 Beep[]
]
```

. Gerando a imagen.

```
imagem1 = Graphics[(*{EdgeForm[{Thin,Black}],FaceForm[Black],*)
  Table[Disk[a[h], {r1[h], r2[h]}], {h, 1, n}], PlotRange → {{0, pixelsx}, {0, pixelsy}},
  PlotRangeClipping → True, ImageSize → {pixelsx, pixelsy}]
```

.Imagem binaria.

```
ColorNegate[Binarize[imagem1, 0.8]]
```

.Orientação normalizada.

.Distribuição de tamanho homogênea.

.Distribuição espacial heterogênea (clusters).

Entrando os valores dos parámetros

```
Clear[a, c, ct, i, j, n, k, vt, imagem1, sementesmaduras, sementes, v, ncorps];
pixelsx = 1000;
pixelsy = 1000;
n = 300;
r1 = 11;
r2 = 11;
vt = 9;
k[1] = {150, 500};
k[2] = {500, 150};
k[3] = {850, 500};
k[4] = {500, 850};
k[5] = {500, 500};
k[6] = {150, 150};
k[7] = {850, 850};
k[8] = {850, 150};
k[9] = {150, 850};
rmax = Max[r1, r2];
sigma = {50, 50};
ct = Complement[Tuples[{Range[0 - First[sigma], pixelsx + First[sigma]],
      Range[0 - Last[sigma], pixelsy + Last[sigma]]}],
   Tuples[{Range[r1 + 1, pixelsx - r1 - 1], Range[r2 + 1, pixelsy - r2 - 1]}] ];
c[
   0] =
  {};
```

Determinando as coordenadas de cada corpo (aquí demora).

```
Timing[
 For[
  i = 1, i < n + 1, i++,
  sementes = Table[{Round[Random[NormalDistribution[First[k[v]], First[sigma]]]],
     Round[Random[NormalDistribution[Last[k[v]], Last[sigma]]]]}, {v, 1, vt}];
  sementesmaduras = Complement[sementes, ct];
  If[sementesmaduras == {}, i = i - 1,
   a[i] = RandomChoice[sementesmaduras];
   imagem = Binarize[Graphics[
      Table[Disk[a[j], {r1, r2}], {j, 1, i}], PlotRange → {{0, pixelsx}, {0, pixelsy}},
      PlotRangeClipping → True, ImageSize → {pixelsx, pixelsy}]];
   ncorps = Max[Flatten[MorphologicalComponents[ColorNegate[imagem]]]];
   If[ncorps == i,
    maxx = First[ a[i]] + r1 + 1;
    minx = First[ a[i]] - r1 - 1;
    maxy = Last[ a[i]] + r2 + 1;
    miny = Last[ a[i]] - r2 - 1;
    c[i] = Tuples[{Range[minx, maxx], Range[miny, maxy]}];
    ct = Union[ct, c[i]], i = i - 1
   ]
  ]; Print[i]
 ];
 Beep[]
]
as = Table[a[i], {i, 1, n}];
```

```
{234.017, Null}
```

```
as = Table[a[i], {i, 1, n}]
```

. Gerando a imagen.

```
imagem1 = Graphics[(*{EdgeForm[{Thin,Black}],FaceForm[Black],*)
  Table[Disk[a[h], {r1, r2}], {h, 1, n}], PlotRange → {{0, pixelsx}, {0, pixelsy}},
  PlotRangeClipping → True, ImageSize → {pixelsx, pixelsy}]
```

.Imagem binaria.

```
imagembi = ColorNegate[Binarize[imagem1]]
```

.Controlando o número de corpos.

```
Max[Flatten[MorphologicalComponents[imagembi]]]
```

.Orientação normalizada.

.Distribuição de tamanho homogênea.

.Distribuição espacial homogênea.

Entrando os valores dos parámetros

```
Clear[a, c, ct, i, j, n, ncorps];
pixelsx = 1000;
pixelsy = 1000;
n = 1000;
r1 = 6;
r2 = 6;
ct = Complement[Tuples[{Range[0, pixelsx], Range[0, pixelsy]}],
   Tuples[{Range[r1 + 2, pixelsx - r1 - 2], Range[r2 + 2, pixelsy - r2 - 2]}] ];
c[
   0] =
  {};
```

Determinando as coordenadas de cada corpo (aquí demora).

```
Timing[
 For[
   i = 1, i < n + 1, i++,
   a[i] = RandomChoice[Complement[Tuples[{Range[0, pixelsx], Range[0, pixelsy]}], ct]];
   imagem = Binarize[Graphics[
      Table[Disk[a[j], {r1, r2}], {j, 1, i}], PlotRange → {{0, pixelsx}, {0, pixelsy}},
      PlotRangeClipping → True, ImageSize → {pixelsx, pixelsy}]];
   ncorps = Max[Flatten[MorphologicalComponents[ColorNegate[imagem]]]];
   If[ncorps == i,
    maxx = First[ a[i]] + r1 + 1;
    minx = First[ a[i]] - r1 - 1;
    maxy = Last[ a[i]] + r2 + 1;
    miny = Last[ a[i]] - r2 - 1;
    c[i] = Tuples[{Range[minx, maxx], Range[miny, maxy]}];
    ct = Union[ct, c[i]], i = i - 1
   ]; Print[i]
  ]
  Beep[]
]
```

. Gerando a imagen.

```
imagem1 = Graphics[(*{EdgeForm[{Thin,Black}],FaceForm[Black],*)
  Table[Disk[a[h], {r1, r2}], {h, 1, n}], PlotRange → {{0, pixelsx}, {0, pixelsy}},
  PlotRangeClipping → True, ImageSize → {pixelsx, pixelsy}]
```

.Imagem binaria.

```
ColorNegate[Binarize[imagem1]]
```

.Distribuição de orientação homogênea.

.Distribuição de tamanho homogênea.

.Distribuição espacial homogênea.

Entrando os valores dos parámetros.

```
Clear[a, c, ct, i, j, n, theta, rotm, v, h, r1, r2, rmax, rmin];
pixelsx = 1000;
pixelsy = 1000;
n = 8;
r1 = 40;
r2 = 80;
rmax = Max[r1, r2];
theta = 1;
rotm = RotationMatrix[cita];
ct = Complement[Tuples[{Range[0, pixelsx], Range[0, pixelsy]}],
   Tuples[{Range[rmax, pixelsx - rmax], Range[rmax, pixelsy - rmax]}] ];
c[
   0] =
  {};
```

Determinando as coordenadas de cada corpo (aquí demora).

```
Timing[
 For[
  i = 1, i < n + 1, i++,
  a[i] = RandomChoice[Complement[Tuples[{Range[0, pixelsx], Range[0, pixelsy]}], ct]];
  imagem = Binarize[Graphics[Table[Rotate[Disk[a[j], {r1, r2}], theta], {j, 1, i}],
     PlotRange → {{0, pixelsx}, {0, pixelsy}},
     PlotRangeClipping → True, ImageSize → {pixelsx, pixelsy}]];
  ncorps = Max[Flatten[MorphologicalComponents[ColorNegate[imagem]]]];
  If[ncorps == i,
   maxx = First[ a[i]] + r1 + 1;
   minx = First[ a[i]] - r1 - 1;
   maxy = Last[ a[i]] + r2 + 1;
   miny = Last[ a[i]] - r2 - 1;
   c[i] = Tuples[{Range[minx, maxx], Range[miny, maxy]}];
   c[i] =
    Table[Round[Plus[Dot[rotm, Plus[Part[c[i], v], -a[i]]], a[i]]], {v, 1, Length[c[i]]}];
   ct = Union[ct, c[i]], i = i - 1
  ]; Print[i]
 ];
 Beep[]
]
```

. Gerando a imagen.

```
imagem1 = Graphics[(*{EdgeForm[{Thin,Black}],FaceForm[Black],*)
  Table[Rotate[Disk[a[h], {r1, r2}], theta], {h, 1, n}],
  PlotRange → {{0, pixelsx}, {0, pixelsy}},
  PlotRangeClipping → True, ImageSize → {pixelsx, pixelsy}]
```

.Imagem binaria.

```
ColorNegate[Binarize[imagem1, 0.8]]
```

.Distribuição de orientação homogênea.

.Distribuição de tamanho heterogênea.

.Distribuição espacial homogênea.

Entrando os valores dos parámetros

```
Clear[a, c, ct, i, j, n, theta, rotm, v, h, r1, r2, rmax, rmin];
pixelsx = 1000;
pixelsy = 1000;
n = 10;
r1max = 80;
r1min = 30;
r2max = 40;
r2min = 15;
rmax = Max[r2max, r1max];
theta = 1;
rotm = RotationMatrix[theta];
ct = Complement[Tuples[{Range[0, pixelsx], Range[0, pixelsy]}],
   Tuples[{Range[rmax, pixelsx - rmax], Range[rmax, pixelsy - rmax]}] ];
c[
   0] =
  {};
```

Determinando as coordenadas de cada corpo (aquí demora).

```
Timing[
 For[
   i = 1, i < n + 1, i++,
   a[i] = RandomChoice[Complement[Tuples[{Range[0, pixelsx], Range[0, pixelsy]}], ct]];
   r1[i] = RandomChoice[Range[r1min, r1max]];
   r2[i] = RandomChoice[Range[r2min, r2max]];
   imagem = Binarize[Graphics[Table[Rotate[Disk[a[j], {r1[j], r2[j]}], theta], {j, 1, i}],
      PlotRange → {{0, pixelsx}, {0, pixelsy}},
      PlotRangeClipping → True, ImageSize → {pixelsx, pixelsy}]];
   ncorps = Max[Flatten[MorphologicalComponents[ColorNegate[imagem]]]];
   If[ncorps == i,
    maxx = First[ a[i]] + r1[i] + 1;
    minx = First[ a[i]] - r1[i] - 1;
    maxy = Last[ a[i]] + r2[i] + 1;
    miny = Last[ a[i]] - r2[i] - 1;
    c[i] = Tuples[{Range[minx, maxx], Range[miny, maxy]}];
    c[i] = Table[
      Round[Plus[Dot[rotm, Plus[Part[c[i], v], -a[i]]], a[i]]], {v, 1, Length[c[i]]}];
    ct = Union[ct, c[i]], i = i - 1
   ]; Print[i]
  ]
  Beep[]
]
```

$\{12.948, \text{Null}^2\}$

. Gerando a imagen.

```
imagem1 = Graphics[(*{EdgeForm[{Thin,Black}],FaceForm[Black],*)
  Table[Rotate[Disk[a[h], {r1[h], r2[h]}], theta], {h, 1, n}],
  PlotRange → {{0, pixelsx}, {0, pixelsy}},
  PlotRangeClipping → True, ImageSize → {pixelsx, pixelsy}]
```

.Imagem binaria.

```
ColorNegate[Binarize[imagem1, 0.8]]
```

8
Apêndice 2 - Código computacional para a determinação do Covdmean de ensembles reais e virtuais

Caracterização da distribuição espacial: Cálculo do Cov(dmean)

Carrega a imagem a analizar, determina suas dimenções e aplica a transformada de distancia.

```
Clear[acs1, ac1, dmean, acsex, er01, er1, er2, acs,
 imagem2, imagem3, imagem4, imagem5, imagem6, imagem7, obj];
er0 = ColorNegate[Import["F:\Nova Tese/Mathemática/imagens/avanzadas/300het5.bmp"]];
sizex = Length[Transpose[ ImageData[er0]]];
sizey = Length[ImageData[er0]];
er1 = ImageAdjust[LaplacianGaussianFilter[DistanceTransform[er0], 1.5]]
```

Segmenta a imagem da transformada.

```
er2 = Binarize[er1, 0.64]
```

Controla o parámetro de segmentação da seção anterior (condição: n[imagtransf] = n[imag])

```
n = Max[Flatten[MorphologicalComponents[er2]]]
```

Determina os elementos das bordas.

```
iborda =
 Union[MorphologicalComponents[er2][[4]], Reverse[MorphologicalComponents[er2]][[4]],
  Transpose[MorphologicalComponents[er2]][[4]],
  Reverse[Transpose[MorphologicalComponents[er2]]][[4]]]
u =
  0;
```

Determinando o dmean para cada corpo (aquí demora).

```
Timing[
 For [i = 1, i < n + 1, i++,
  If[Intersection[{i}, iborda] == {},
   obj[i] = Reverse[Position[Reverse[MorphologicalComponents[er2]], i], 2];
   imagem2 =
    Binarize[ColorNegate[Graphics[Point[obj[i]], PlotRange → {{0, sizex}, {0, sizey}},
       PlotRangeClipping → True, ImageSize → {sizex, sizey}]]];
   imagem3 = Dilation[imagem2, 3];
   imagem4 = GeodesicDilation[Binarize[imagem3], er2];
   imagem5 = GeodesicDilation[imagem4, ColorNegate[er0]];
   imagem6 = MorphologicalPerimeter[imagem5];
   imagem7 = GeodesicDilation[imagem2, imagem6];
   ac1 = Reverse[Position[Reverse[MorphologicalComponents[imagem7]], 1], 2];
   acs = Table[Reverse[Position[Reverse[MorphologicalComponents[imagem6]], h], 2],
     {h, 1, Max[Flatten[MorphologicalComponents[imagem6]]]}];
   acsex = Complement[acs, {ac1}];
   u = u + 1;
   dmean[u] =
    N[Mean[Table[Min[Table[EuclideanDistance[Nearest[ac1, acsex[[j]][[i]]][[1]],
          acsex[[j]][[i]]], {i, 1, Length[acsex[[j]]]}]], {j, 1, Length[acsex]}]]];
   Print[i]
  ]
 ]]
Beep[]
```

Calcula o valor do Cov(dmean).

```
Covdmean =
 StandardDeviation[Table[(dmean[i]), {i, 1, u}]] / Mean[Table[(dmean[i]), {i, 1, u}]]
```

9
Apêndice 3 - Códigos computacionais para a determinação do Covdmean local e para a construção de mapas de Covdmean

Mapas de Cov (dmean)

Carrega a imagem e determina suas dimenções

```
er0 = ColorNegate[Import["F:\Nova Tese/Mathemática/imagens/avanzadas/img11.bmp"]];
sizetx = Length[Transpose[ ImageData[er0]]];
sizety = Length[ImageData[er0]];
```

Introduze os valores que definem a características da quadratura do mapa

```
mapax = 2;
mapay = 2;
```

Determina o centro e as dimenções de todos os quadrantes

```
sizex = IntegerPart[sizetx / mapax];
sizey = IntegerPart[sizety / mapay];
pontos = Tuples[{Range[IntegerPart[sizex / 2], IntegerPart[sizetx – sizex / 2], sizex],
   Range[IntegerPart[sizey / 2], IntegerPart[sizety – sizey / 2], sizey]}]
```

Determina o Cov (dmean) para cada quadrante do mapa

```
For[j = 1, j < Length[pontos] + 1, j++,
 ponto = pontos[[j]];
 tamanhox = sizex;
 tamanhoy = sizey;
re = ImageData[er0, "Byte"][[
   ponto[[1]] + 1 - IntegerPart[tamanhoy / 2] ;; ponto[[1]] - 1 + IntegerPart[tamanhoy / 2],
   ponto[[2]] + 1 - IntegerPart[tamanhox / 2] ;; ponto[[2]] - 1 + IntegerPart[tamanhoy / 2]]];
 er0p = Image[re];
 sizexn = Length[Transpose[ ImageData[er0p]]];
 sizeyn = Length[ImageData[er0p]];
 n0 = Max[Flatten[MorphologicalComponents[ColorNegate[er0p]]]];
 er1 = ImageAdjust[LaplacianGaussianFilter[DistanceTransform[er0p], 1.5]];
 er2 = Binarize[er1, 0.55];
 n = Max[Flatten[MorphologicalComponents[er2]]];
 iborda =
  Union[MorphologicalComponents[er2][[4]], Reverse[MorphologicalComponents[er2]][[4]],
   Transpose[MorphologicalComponents[er2]][[4]],
   Reverse[Transpose[MorphologicalComponents[er2]]][[4]]];
 u = 0;
 Timing[
  For [i = 1, i < n + 1, i++,
   If[Intersection[{i}, iborda] == {},
    obj[i] = Reverse[Position[Reverse[MorphologicalComponents[er2]], i], 2];
    imagem2 =
     Binarize[ColorNegate[Graphics[Point[obj[i]], PlotRange → {{0, sizexn}, {0, sizeyn}},
        PlotRangeClipping → True,  ImageSize → {sizexn, sizeyn}]]];
    imagem3 = Dilation[imagem2, 3];
    imagem4 = GeodesicDilation[Binarize[imagem3], er2];
    imagem5 = GeodesicDilation[imagem4, ColorNegate[er0p]];
    imagem6 = MorphologicalPerimeter[imagem5];
    imagem7 = GeodesicDilation[imagem2, imagem6];
    ac1 = Reverse[Position[Reverse[MorphologicalComponents[imagem7]], 1], 2];
    acs = Table[Reverse[Position[Reverse[MorphologicalComponents[imagem6]], h], 2],
      {h, 1, Max[Flatten[MorphologicalComponents[imagem6]]]}];
    acsex = Complement[acs, {ac1}];
    u = u + 1;
    dmean[u] =
     N[Mean[Table[Min[Table[EuclideanDistance[Nearest[ac1, acsex[[j]][[i]]][[1]],
            acsex[[j]][[i]]], {i, 1, Length[acsex[[j]]]}]], {j, 1, Length[acsex]}]]];
    Print[i]
   ]
  ]];
 Beep[];
 covdmean[j] =
  StandardDeviation[Table[(dmean[i]), {i, 1, u}]] / Mean[Table[(dmean[i]), {i, 1, u}]];
 Print[covdmean[j]]
]
```

Construe o mapa de Cov (dmean)

```
mapaCov =
 MatrixForm[Transpose[Partition[Table[covdmean[i], {i, 1, Length[pontos]}], mapay]]]
```

Determinação do Cov(dmean) local

Carrega a imagem

```
er0 = ColorNegate[Import["F:\Nova Tese/Mathemática/imagens/avanzadas/img11.bmp"]];
```

Introduce o valor do pixel central e o tamanho do campo desejado

```
ponto = {700, 500};
tamanhox = 100;
tamanhoy = 130;
re = ImageData[er0, "Byte"][[
   ponto[[1]] - IntegerPart[tamanhoy / 2] ;; ponto[[1]] - 1 + IntegerPart[tamanhoy / 2],
   ponto[[2]] - IntegerPart[tamanhox / 2] ;; ponto[[2]] - 1 + IntegerPart[tamanhox / 2]]];
```

Criar a imagem do campo desejado

```
er0 = Image[re]
tamanhoxn = Length[Transpose[ ImageData[er0]]];
tamanhoyn = Length[ImageData[er0]];
```

Determina o número de corpos no campo

```
n0 = Max[Flatten[MorphologicalComponents[ColorNegate[er0]]]]
```

```
15
```

Aplica a transforma de distancia na imagem do campo

```
er1 = ImageAdjust[LaplacianGaussianFilter[DistanceTransform[er0], 1.5]]
```

Segmenta a imagem da transformada.

```
er2 = Binarize[er1, 0.55]
```

Controla o parámetro de segmentação da seção anterior (condição: n[imagtransf] = n[imag])

```
n = Max[Flatten[MorphologicalComponents[er2]]]
```

Determina os elementos das bordas.

```
iborda =
 Union[MorphologicalComponents[er2][[4]], Reverse[MorphologicalComponents[er2]][[4]],
  Transpose[MorphologicalComponents[er2]][[4]],
  Reverse[Transpose[MorphologicalComponents[er2]]][[4]]]
u =
  0;
```

Determinando o dmean para cada corpo (aquí demora).

```
Timing[
 For [i = 1, i < n + 1, i++,
  If[Intersection[{i}, iborda] == {},
   obj[i] = Reverse[Position[Reverse[MorphologicalComponents[er2]], i], 2];
   imagem2 = Binarize[
     ColorNegate[Graphics[Point[obj[i]], PlotRange → {{0, tamanhoxn}, {0, tamanhoyn}},
       PlotRangeClipping → True, ImageSize → {tamanhoxn, tamanhoyn}]]];
   imagem3 = Dilation[imagem2, 3];
   imagem4 = GeodesicDilation[Binarize[imagem3], er2];
   imagem5 = GeodesicDilation[imagem4, ColorNegate[er0]];
   imagem6 = MorphologicalPerimeter[imagem5];
   imagem7 = GeodesicDilation[imagem2, imagem6];
   ac1 = Reverse[Position[Reverse[MorphologicalComponents[imagem7]], 1], 2];
   acs = Table[Reverse[Position[Reverse[MorphologicalComponents[imagem6]], h], 2],
     {h, 1, Max[Flatten[MorphologicalComponents[imagem6]]]}];
   acsex = Complement[acs, {ac1}];
   u = u + 1;
   dmean[u] =
    N[Mean[Table[Min[Table[EuclideanDistance[Nearest[ac1, acsex[[j]][[i]]][[1]],
           acsex[[j]][[i]]], {i, 1, Length[acsex[[j]]]}]], {j, 1, Length[acsex]}]];
   Print[i]
  ]
 ]]
Beep[]
```

Calcula o valor do Cov(dmean) para o campo local seleccionado.

```
Covdmean =
 StandardDeviation[Table[(dmean[i]), {i, 1, u}]] / Mean[Table[(dmean[i]), {i, 1, u}]]
```