SIFT (Scale-invariant feature transform), SURF (Speeded up robust features)

Scale- and rotation-invariant interest point detector and descriptor.

- detector (keypoints)
- descriptor
- matching

Aplicações: Camera calibration, 3D reconstruction, image registration, and object recognition.

[Lowe, 2004] David G. Lowe, “Distinctive Image Features from Scale-Invariant Keypoints,” Int. J. Computer Vision, Vol. 60, No. 2, Nov., 2004, pp. 91-110. ["artigo-texto" para seguir.]

[Bay, 2006] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool, “SURF: Speeded Up Robust Features,” ECCV 2006, Lecture Notes in Computer Science, Vol. 3951/2006, pp. 404-417.

[Mikolajczyk, 2005] K. Mikolajczyk, A performance evaluation of local descriptors, IEEE. T. Patt. Analysis Machine Intelligence 27(10) (2005) 1615-1630.

[Mikolajczyk, 2005] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas F. Schaffalitzky, T. Kadir, and L. Van Gool, “A Comparison of Affine Region Detectors,” Int. J. of Computer Vision 65(1/2), 43-72, 2005.

[Matas, 2002] J. Matas, O. Chum, M. Urban, and T. Pajdla. "Robust wide baseline stereo from maximally stable extremal regions." Proc. of British Machine Vision Conference, pages 384-396, 2002.

SIFT (Scale-invariant feature transform)

O programa SIFT original do Lowe pode ser baixado de:
http://www.cs.ubc.ca/~lowe/keypoints/siftDemoV4.zip
Há versões para Windows (siftWin32.exe) e Linux (sift). Leia README. Você deve compilar o programa match.c.

Antes de estudar a técnica, vamos rodar os programas e verificar como funcionam. Estes programas só funcionam com imagens .pgm.

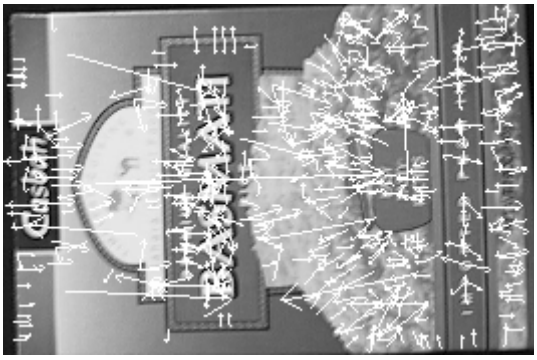
basmati.pgm


scene.pgm

Vamos efetuar casamentos entre basmati e scene. Para isso, primeiro temos que calcular keypoints das duas imagens (no Windows, troque sift por siftWin32):

```
sift -display <basmati.pgm >basmati-keys.pgm
sift -display <scene.pgm >scene-keys.pgm
```

O que gera:


basmati-keys.pgm

scene-keys.pgm

Em vez de gerar imagens com pontos-chaves, podemos gerar arquivos-textos:

```
sift <basmati.pgm >basmati-keys.txt
sift <scene.pgm >scene-keys.txt
```

Mostro abaixo o início dos dois arquivos-textos:

```
basmati-keys.txt:
579 128
87.67 55.51 22.07 0.024
 8 0 0 0 0 0 0 9 56 0 0 0 0 21 24 97 1 0 0 0
 2 97 40 20 1 0 0 0 5 9 37 52 43 10 0 0 0 0 0 7
 161 49 0 1 11 41 28 94 57 5 0 9 91 161 62 71 17 4 0 2
 80 38 73 107 52 2 0 0 0 0 0 11 161 28 7 34 18 2 0 55
 67 65 33 158 111 13 1 9 25 112 29 13 84 23 1 10 12 0 0 0
 0 0 0 0 87 2 1 10 5 0 0 4 4 15 11 46 19 0 0 0
 5 47 10 3 8 1 0 0
143.06 31.42 13.57 -0.054
 23 10 0 0 0 0 0 0 159 97 0 0 0 0 0 10 159 36 2 7
 100 22 9 16 30 8 1 7 159 27 2 8 25 1 0 0 0 0 0 6
 159 27 0 0 0 0 0 85 159 43 39 90 70 0 0 19 8 22 17 121
 122 1 0 0 15 4 0 0 0 0 0 0 159 67 0 0 1 0 0 7
 73 19 3 36 91 0 0 1 0 1 6 57 72 0 0 0 0 0 0 0
 0 0 0 0 7 1 0 0 0 0 0 0 1 0 0 0 1 0 0 0
 0 0 0 0 0 0 0 0
(...)

scene-keys.txt:
1021 128
194.29 389.65 25.42 -2.259
(...)
```

Copio do README:
The file format starts with 2 integers giving the total number of keypoints and the length of the descriptor vector for each keypoint (128). Then the location of each keypoint in the image is specified by 4 floating point numbers giving subpixel row and column location, scale, and orientation (in radians from -PI to PI). (...) Finally, the invariant descriptor vector for the keypoint is given as a list of 128 integers in range [0,255].

Agora, vamos fazer o casamento:

```
match -im1 basmati.pgm -k1 basmati-keys.txt -im2 scene.pgm -k2 scene-keys.txt > basmati-scene.pgm
```

basmati-scene.pgm

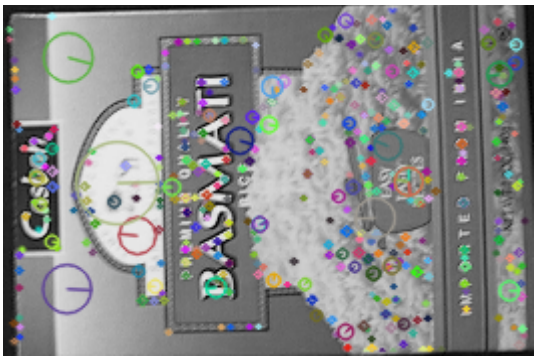
Achou 37 casamentos. 1 casamento errado. 36 casamentos corretos.

Vamos fazer a mesma coisa, usando funções do OpenCV (nova maneira de chamar, v2.4):

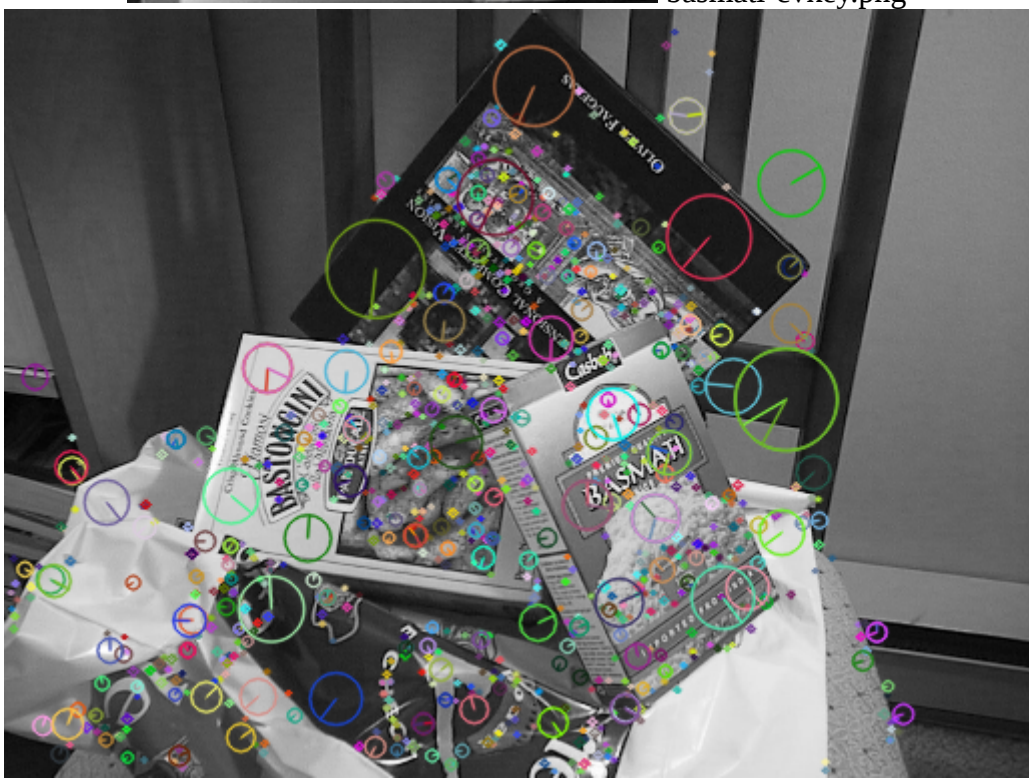
```
// cvsift1.cpp: faz associacao sift - grad2017
// linkar com opencv2: compila cvsift1 -c
// http://docs.opencv.org/2.4/modules/nonfree/doc/feature_detection.html
#include <cekeikon.h>
#include <opencv2/nonfree/nonfree.hpp>

void ratioTest(const vector< vector<DMatch> >& nnmatches, vector<DMatch>& matches, double
ratio=0.8) {
  for (unsigned i=0; i<nnmatches.size(); i++) {
    const vector<DMatch>& m=nnmatches[i];
    if (m.size()!=2) erro("Erro ratioTest inesperado");
    if (m[1].distance>epsilon) {
      double r=m[0].distance/m[1].distance;
      if (r<=ratio) matches.push_back(m[0]);
    }
  }
}

int main() {
  Mat_<COR> sai;

  Mat_<GRY> a1; le(a1,"basmati.pgm");
  SIFT sift1;
  vector<KeyPoint> k1;
  Mat_<FLT> d1;
  sift1(a1,Mat(),k1,d1,false);
  xprint(k1.size());
  xprint(d1.size());
  drawKeypoints(a1,k1,sai,Scalar(-1,-1,-1,-1),DrawMatchesFlags::DRAW_RICH_KEYPOINTS);
  imp(sai,"basmati-cvkey.png");

  Mat_<GRY> a2; le(a2,"scene.pgm");
  SIFT sift2;
  vector<KeyPoint> k2;
  Mat_<FLT> d2;
  sift2(a2,Mat(),k2,d2,false);
  xprint(k2.size());
  xprint(d2.size());
  drawKeypoints(a2,k2,sai,Scalar(-1,-1,-1,-1),DrawMatchesFlags::DRAW_RICH_KEYPOINTS);
  imp(sai,"scene-cvkey.png");

  vector< vector<DMatch> > nnmatches;
  BFMatcher matcher(NORM_L2);
  //FlannBasedMatcher matcher;
  matcher.knnMatch(d1,d2,nnmatches,2);
  xprint(nnmatches.size());

  vector<DMatch> matches;
  ratioTest(nnmatches,matches); // ordenado por ratio
  xprint(matches.size());

  drawMatches(a1,k1,a2,k2,matches,sai);
  imp(sai,"basmati-scene-cvmatch.png");
}
```

```
k1.size() = 489
d1.size() = [128 x 489]
k2.size() = 969
d2.size() = [128 x 969]
nnmatches.size() = 489
matches.size() = 64
```

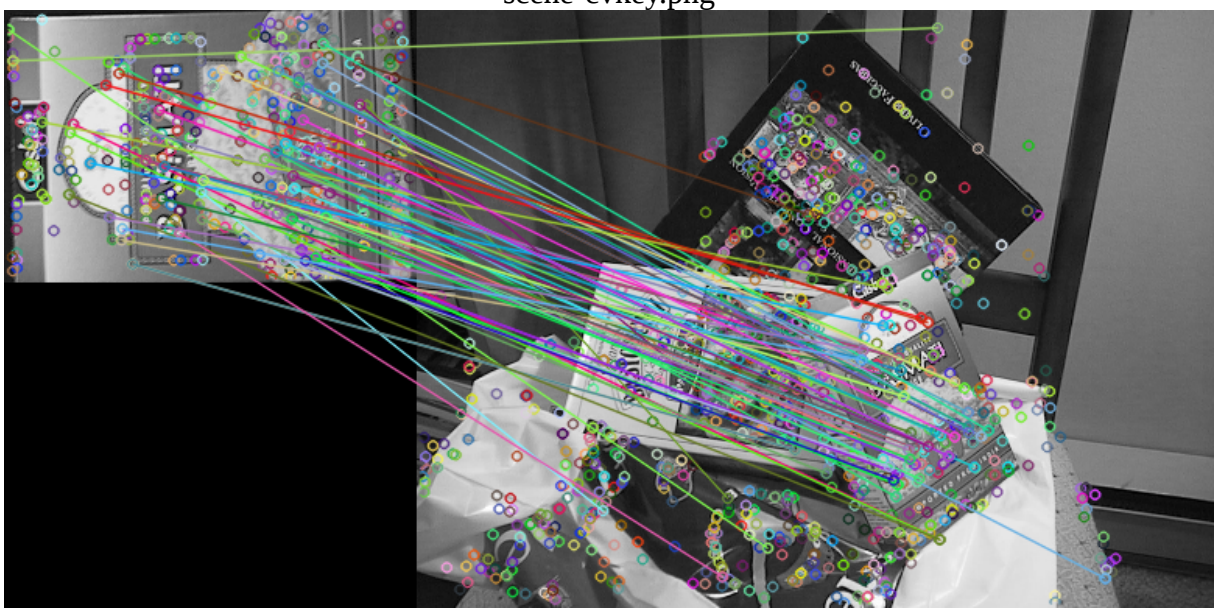
[
2) Colorido, usando a implementação do Geusebroek:
>csift
3) Usando funções do OpenCV 2.4.10
>descript
Problemas para buscar modelos simples com poucas texturas e padrões repetidos.
]



basmati-cvkey.png



scene-cvkey.png

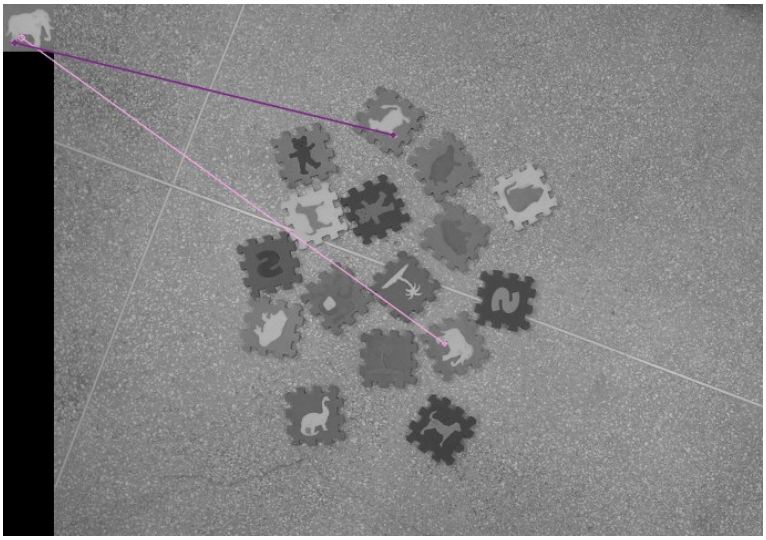

basmati-scene-cvmatch.png

Veja na apostila "espescala" sobre como SIFT acha os keypoints.

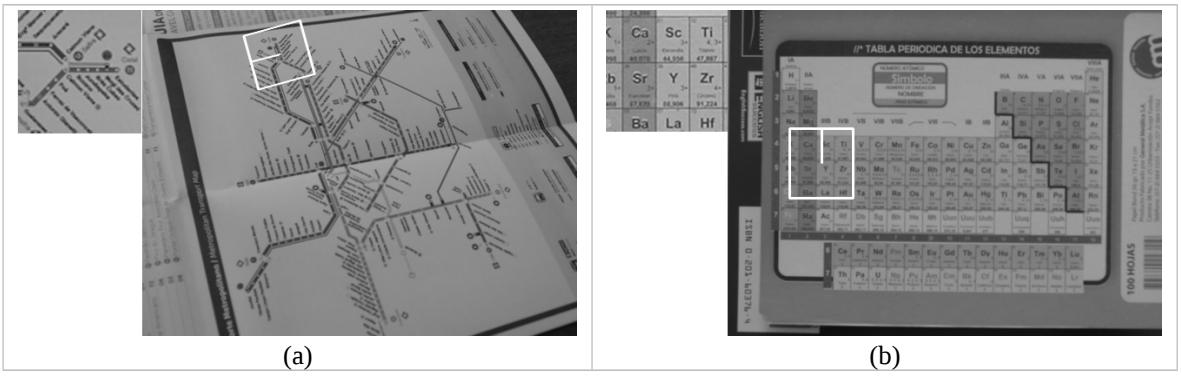
Veja o artigo do Lowe para a descrição completa do SIFT.
[Lowe, 2004] David G. Lowe, “Distinctive Image Features from Scale-Invariant Keypoints,” Int. J. Computer Vision, Vol. 60, No. 2, Nov., 2004, pp. 91-110.

SIFT falha quando (exemplos extraídos de [Guillermo2015]):

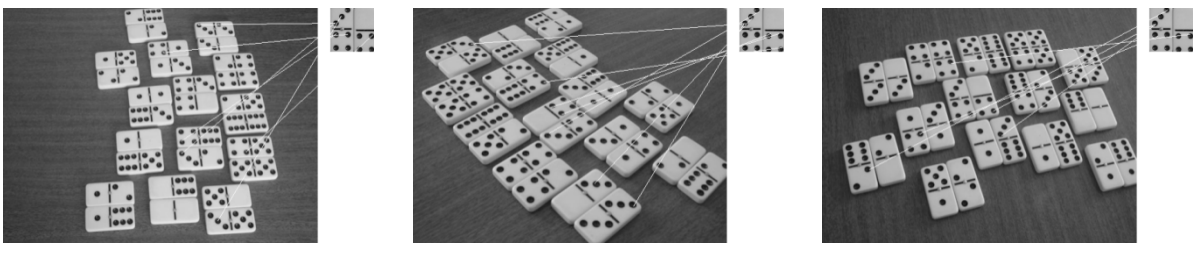
1) Há pouca textura.



2) Padrões que se repetem.



(a) (b)



3) Mudança de brilho e contraste.

É possível fazer template matching invariante por rotação e escala usando SIFT, com a ajuda de transformada generalizada de Hough. Veja apostila da transformada de Hough.

Calculando Hough generalizado após SIFT para localizar objetos:

```
>sift Find_obj olho.tga lennag.tga olho.ppm
```

 olho.tga

 lennag.tga

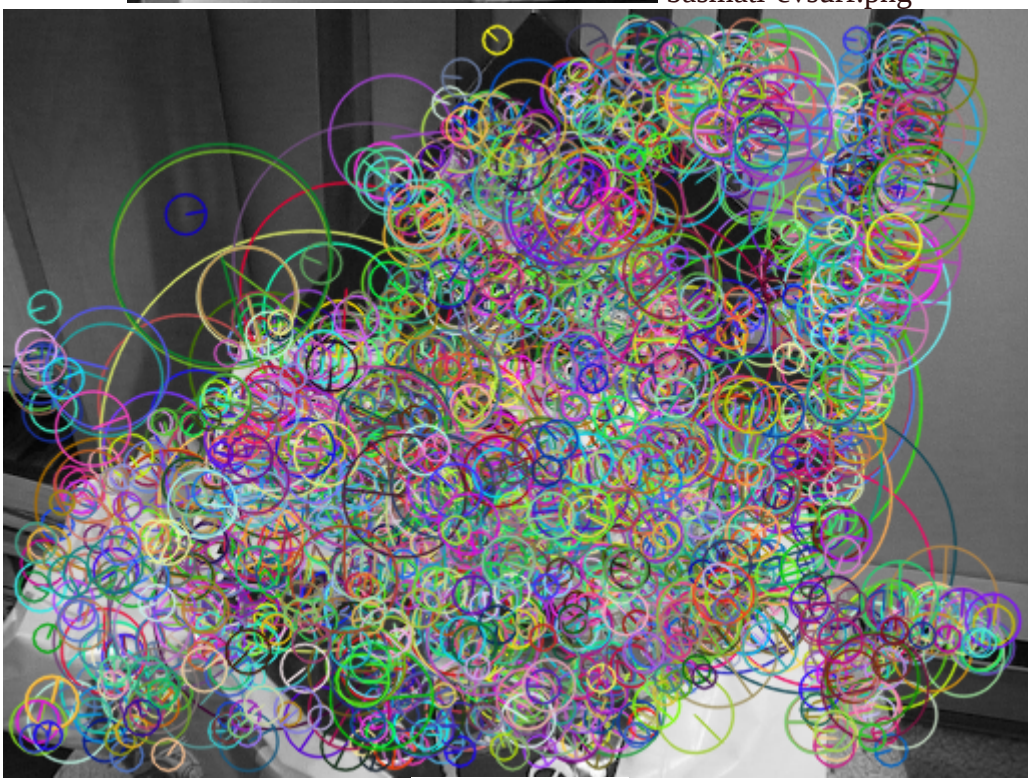
olho.ppm

SURF:

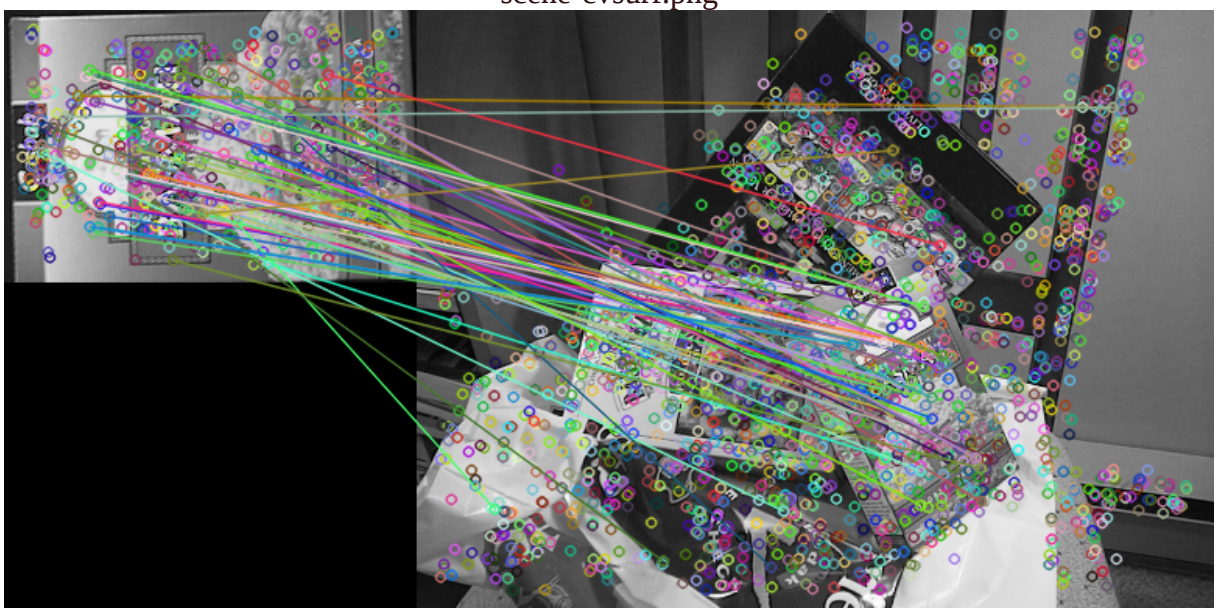
SURF é outro método similar a SIFT. É mais rápido que SIFT. Mas é menos acurado.

```
// cvsurf1.cpp: faz associacao surf - grad2017
// linkar com opencv2: compila cvsurf1 -c
#include <cekeikon.h>
#include <opencv2/nonfree/nonfree.hpp>

void ratioTest(const vector< vector<DMatch> >& nnmatches, vector<DMatch>& matches, double
ratio=0.8) {
  for (unsigned i=0; i<nnmatches.size(); i++) {
    const vector<DMatch>& m=nnmatches[i];
    if (m.size()!=2) erro("Erro ratioTest inesperado");
    if (m[1].distance>epsilon) {
      double r=m[0].distance/m[1].distance;
      if (r<=ratio) matches.push_back(m[0]);
    }
  }
}

int main() {
  Mat_<COR> sai;

  Mat_<GRY> a1; le(a1,"basmati.pgm");
  SURF surf1;
  vector<KeyPoint> k1;
  Mat_<FLT> d1;
  surf1(a1,Mat(),k1,d1);
  xprint(k1.size());
  xprint(d1.size());
  drawKeypoints(a1,k1,sai,Scalar(-1,-1,-1,-1),DrawMatchesFlags::DRAW_RICH_KEYPOINTS);
  imp(sai,"basmati-cvsurf.png");

  Mat_<GRY> a2; le(a2,"scene.pgm");
  SURF surf2;
  vector<KeyPoint> k2;
  Mat_<FLT> d2;
  surf2(a2,Mat(),k2,d2);
  xprint(k2.size());
  xprint(d2.size());
  drawKeypoints(a2,k2,sai,Scalar(-1,-1,-1,-1),DrawMatchesFlags::DRAW_RICH_KEYPOINTS);
  imp(sai,"scene-cvsurf.png");

  vector< vector<DMatch> > nnmatches;
  BFMatcher matcher(NORM_L2);
  //FlannBasedMatcher matcher;
  matcher.knnMatch(d1,d2,nnmatches,2);
  xprint(nnmatches.size());

  vector<DMatch> matches;
  ratioTest(nnmatches,matches); // ordenado por ratio
  xprint(matches.size());

  drawMatches(a1,k1,a2,k2,matches,sai);
  imp(sai,"basmati-scene-cvsurf.png");
}

k1.size() = 607
d1.size() = [64 x 607]
k2.size() = 1822
d2.size() = [64 x 1822]
nnmatches.size() = 607
matches.size() = 55
```

basmati-cvsurf.png



scene-cvsurf.png



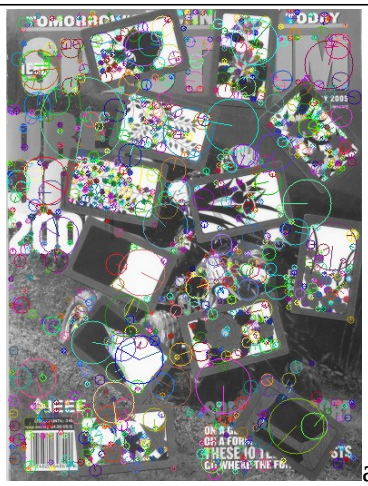
basmati-scene-cvsurf.png

```
@REM 2016
descript keysift a1.pgm a1-key.xml 0.5 1
descript keymyshow a1.pgm a1-key.xml a1-my.ppm
descript keysift a2.pgm a2-key.xml 0.5 1
descript keymyshow a2.pgm a2-key.xml a2-my.ppm
descript dessift a1.pgm a1-des.xml a1-key.xml
descript dessift a2.pgm a2-des.xml a2-key.xml
descript matratio a1-des.xml a2-des.xml match.txt ratio=0.08
descript matmyshow a1.pgm a2.pgm match.txt sai.png
```


a1


a2

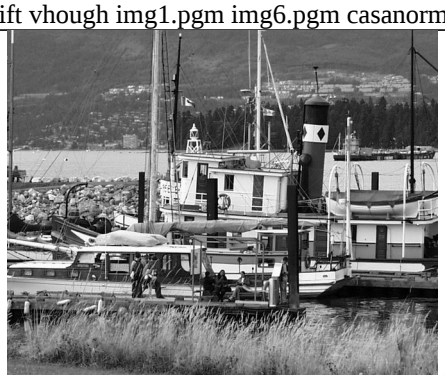

a1-my


a2-my


sai

```
:: Usando SIFT de Lowe/Proeikon - 2011
sift sift img1.pgm img1.key
sift sift img6.pgm img6.key
sift showkey img1.pgm img1.key img1k.ppm
sift showkey img6.pgm img6.key img6k.ppm
sift match img1.key img6.key result.res
sift showmat img1.pgm img6.pgm result.res result.ppm
sift normcasa img1.pgm result.res casanorm.txt
```

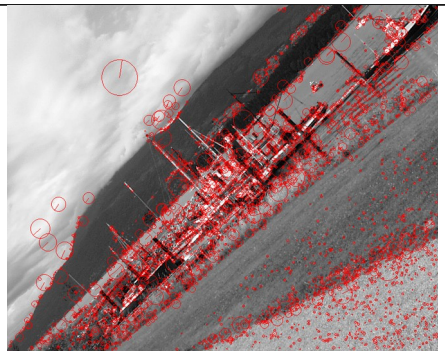
sift vhough img1.pgm img6.pgm casanorm.txt p.ppm



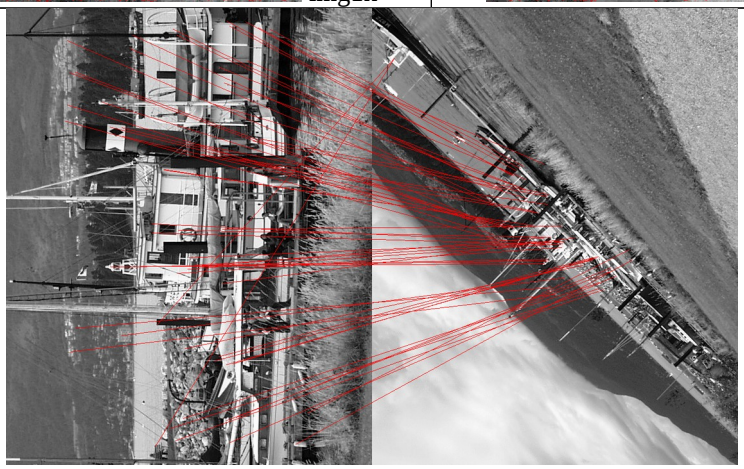
img1.pgm

img6.pgm

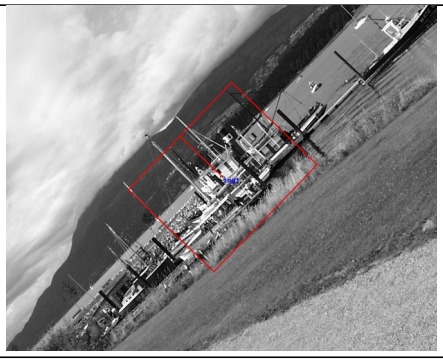
img1k



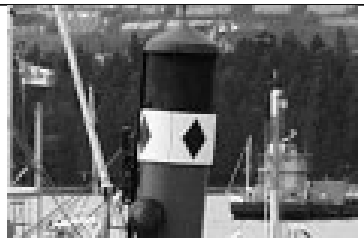
img6k



result.ppm



p.ppm



chamine.pgm

chamine.ppm

result.ppm

p.ppm

```
sift sift chamine.pgm chamine.key
sift sift img6.pgm img6.key
sift showkey chamine.pgm chamine.key chamine.ppm
sift match chamine.key img6.key result.res
sift showmat chamine.pgm img6.pgm result.res result.ppm
sift normcasa chamine.pgm result.res casanorm.txt
sift vhough chamine.pgm img6.pgm casanorm.txt p.ppm
```

Para fazer template matching usando SIFT/SURF, utiliza-se a transformada de Hough generalizada.

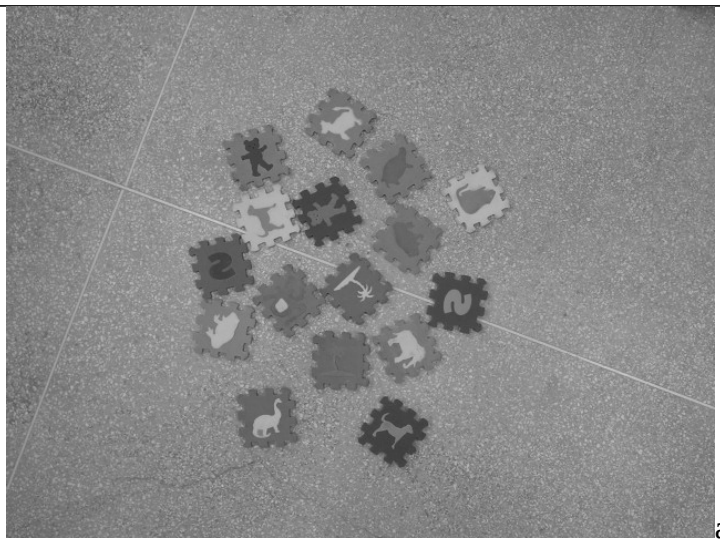
```
:: Usando SIFT de OpenCV/Cekeikon
descript keysift img1.pgm img1k.xml 0.06 10
descript keysift img6.pgm img6k.xml 0.06 10
descript keymyshow img1.pgm img1k.xml img1k.ppm
descript keymyshow img6.pgm img6k.xml img6k.ppm
descript dessift img1.pgm img1d.xml img1k.xml
descript dessift img6.pgm img6d.xml img6k.xml
descript matratio img1d.xml img6d.xml match.txt ratio=0.6
descript matshow img1.pgm img6.pgm match.txt sai.png
descript matrate match.txt H1to6p
```

img1k

img6k

sai.png

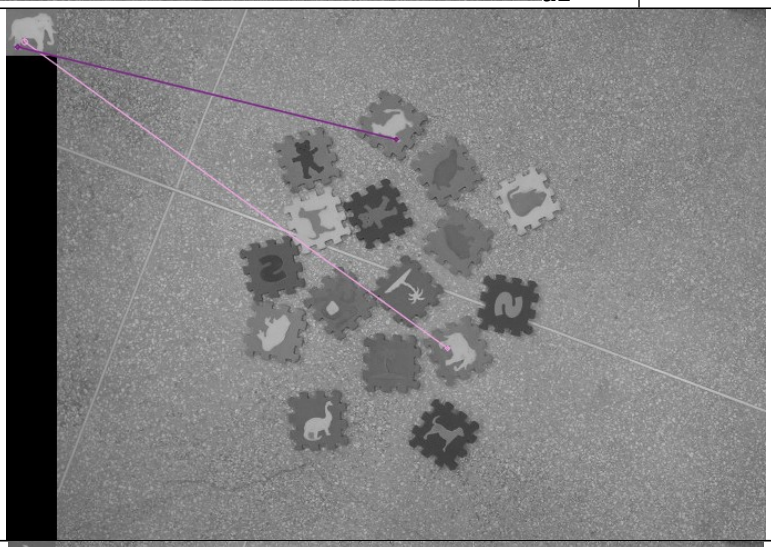
```
H1to6p:
   2.9992872e-01   2.2821975e-01   2.2930182e+02
  -2.3832758e-01   2.4564042e-01   3.6767399e+02
   9.9064973e-05  -5.8498673e-05   1.0000000e+00
```

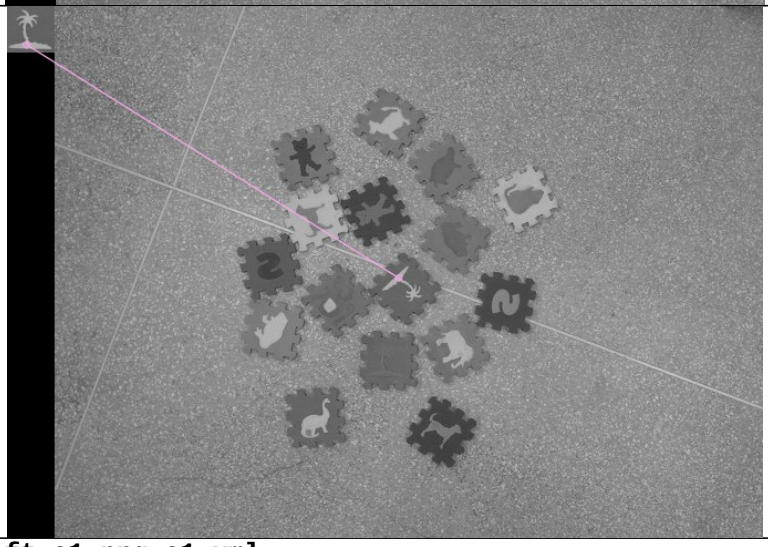


a1

q01

q02





```
descript dessift a1.png a1.xml
descript dessift q01.png q01.xml
descript dessift q02.png q02.xml
descript MatRatio q01.xml a1.xml matchq01a1.txt
descript MatShow q01.png a1.png matchq01a1.txt matq01a1.png
descript MatRatio q02.xml a1.xml matchq02a1.txt
descript MatShow q02.png a1.png matchq02a1.txt matq02a1.png
```