Feature Description (SIFT + SURF)

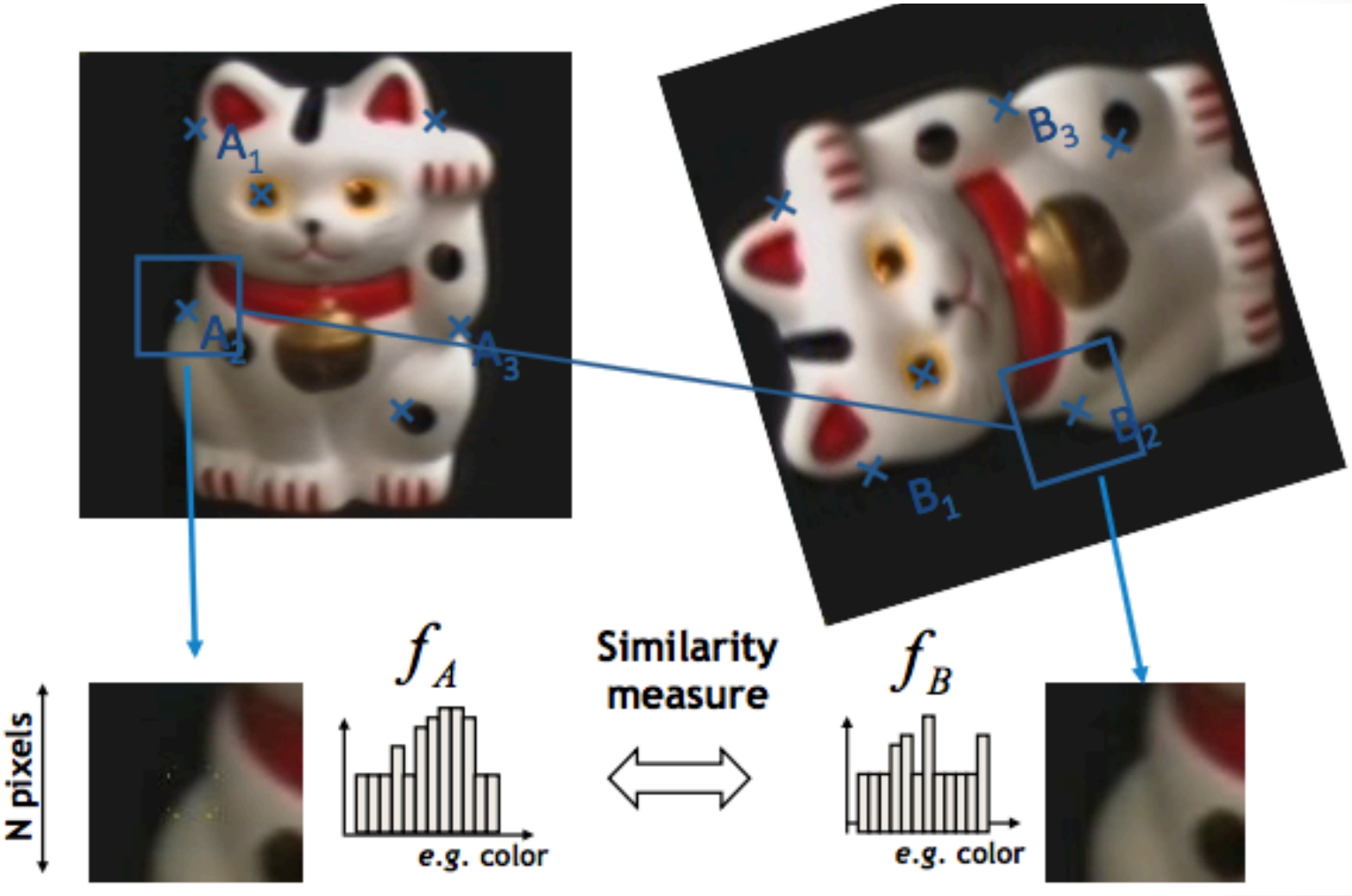
Prof. Dr. Geraldo Braz Junior

Feature Detection

1. Encontre um conjunto de **keypoints**
2. Defina uma região ao redor do keypoint
3. Normalize a região
4. Extraia características dessa região (**features**) e as normalize

- Utilize as características para realizar o **match**

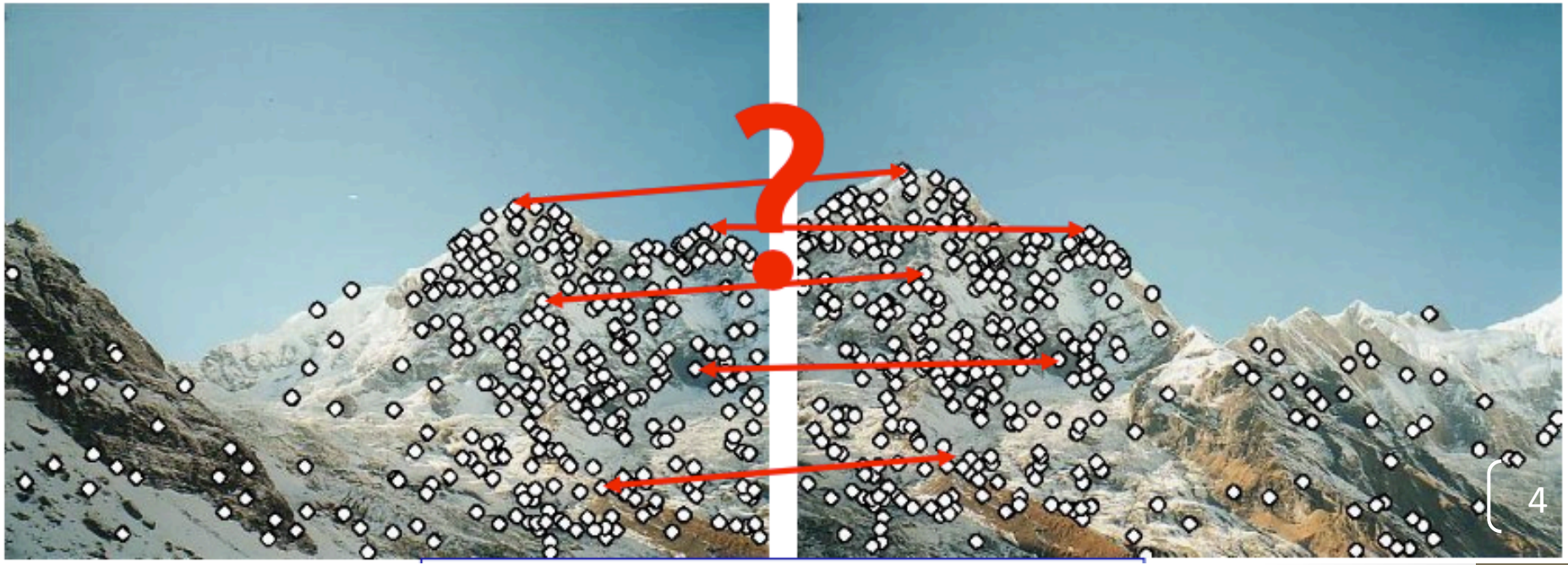
Abordagem básica



Nota de aula Fei-Fei Li

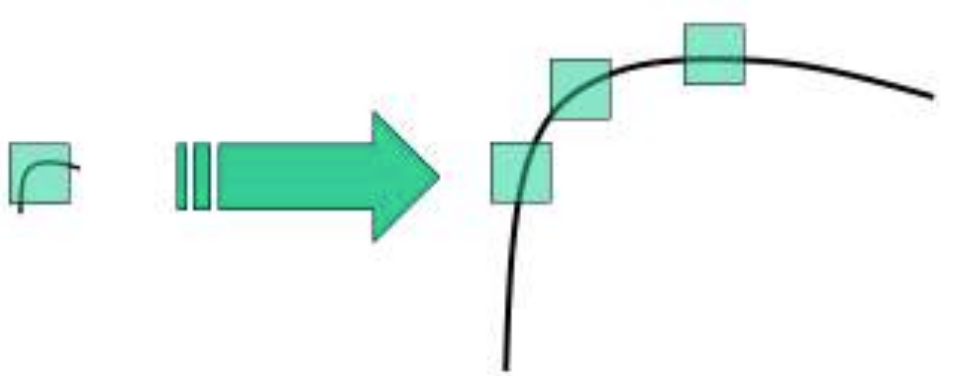
Até então

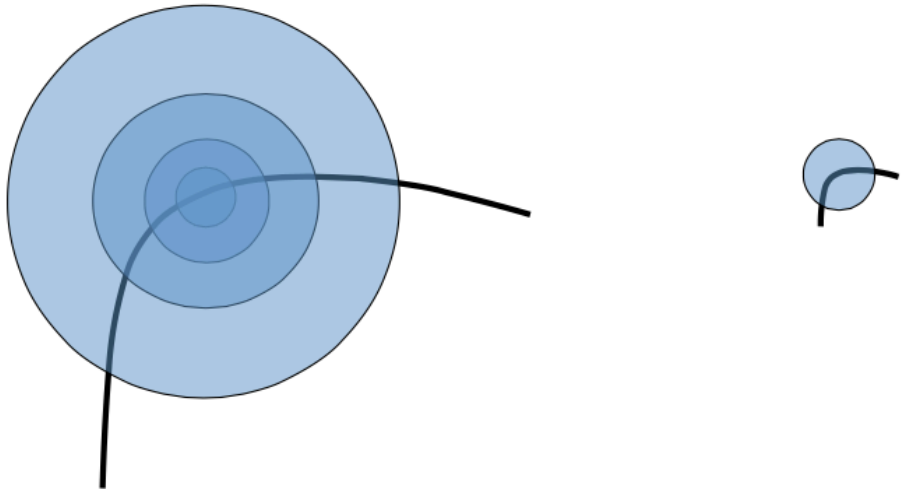
- Sabemos como detectar pontos (Harris e FAST)
- Como descrevê-los?



Problema

- Variação de escala. Um **canto** nem sempre é canto numa **escala diferente**

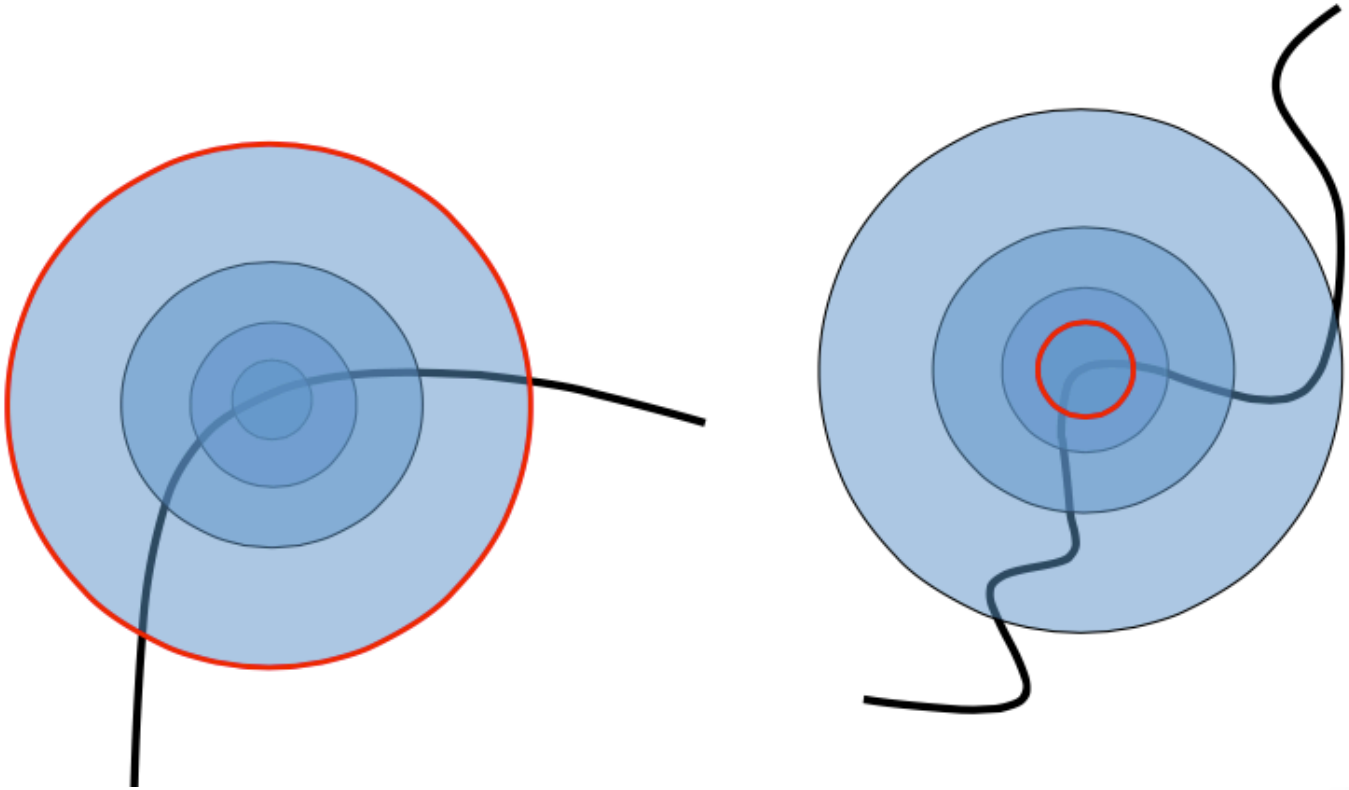




Quando a escala diminui, a janela pode identificar que o canto agora é apenas borda

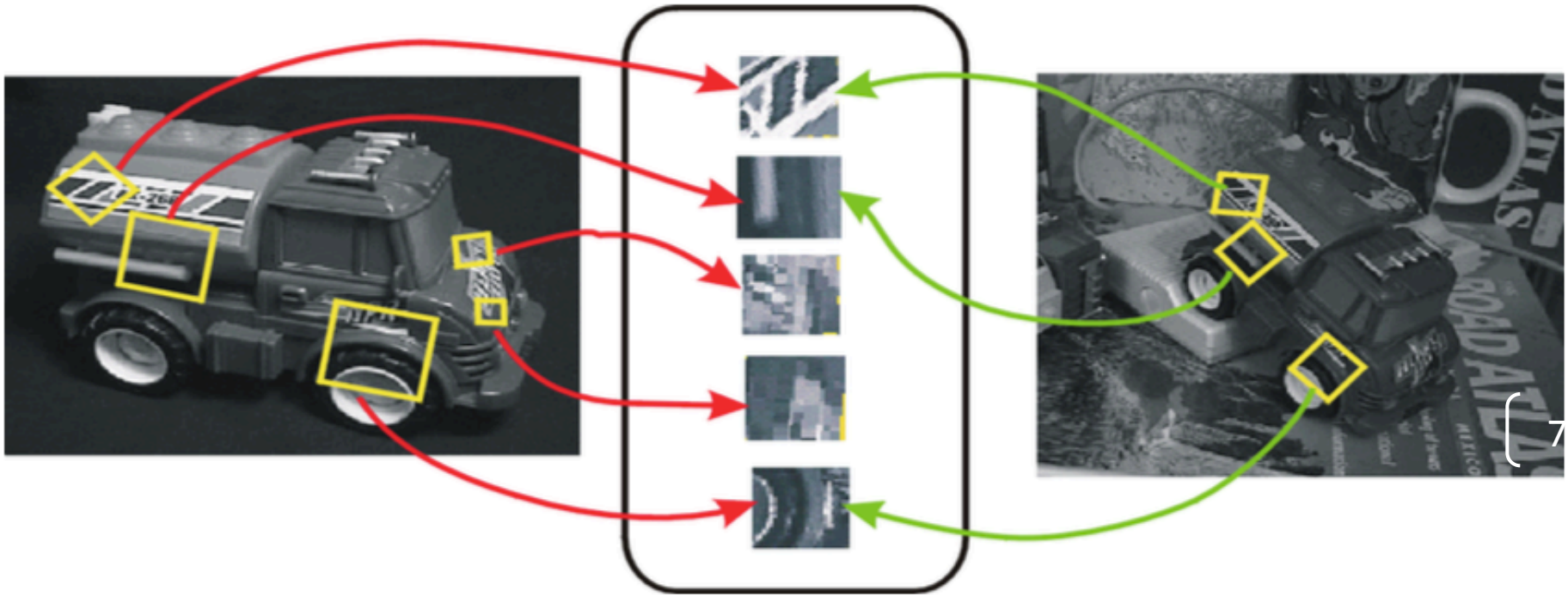
Problema

- Como achar a janela (ou círculo neste caso) do tamanho correto, independente da imagem?



O que se deseja

- Transformação de características da imagem invariantes a rotação, escala, e iluminação



Scale Invariant Feature Transform (**SIFT**)

Uma proposta

- **D. Lowe** propõe:

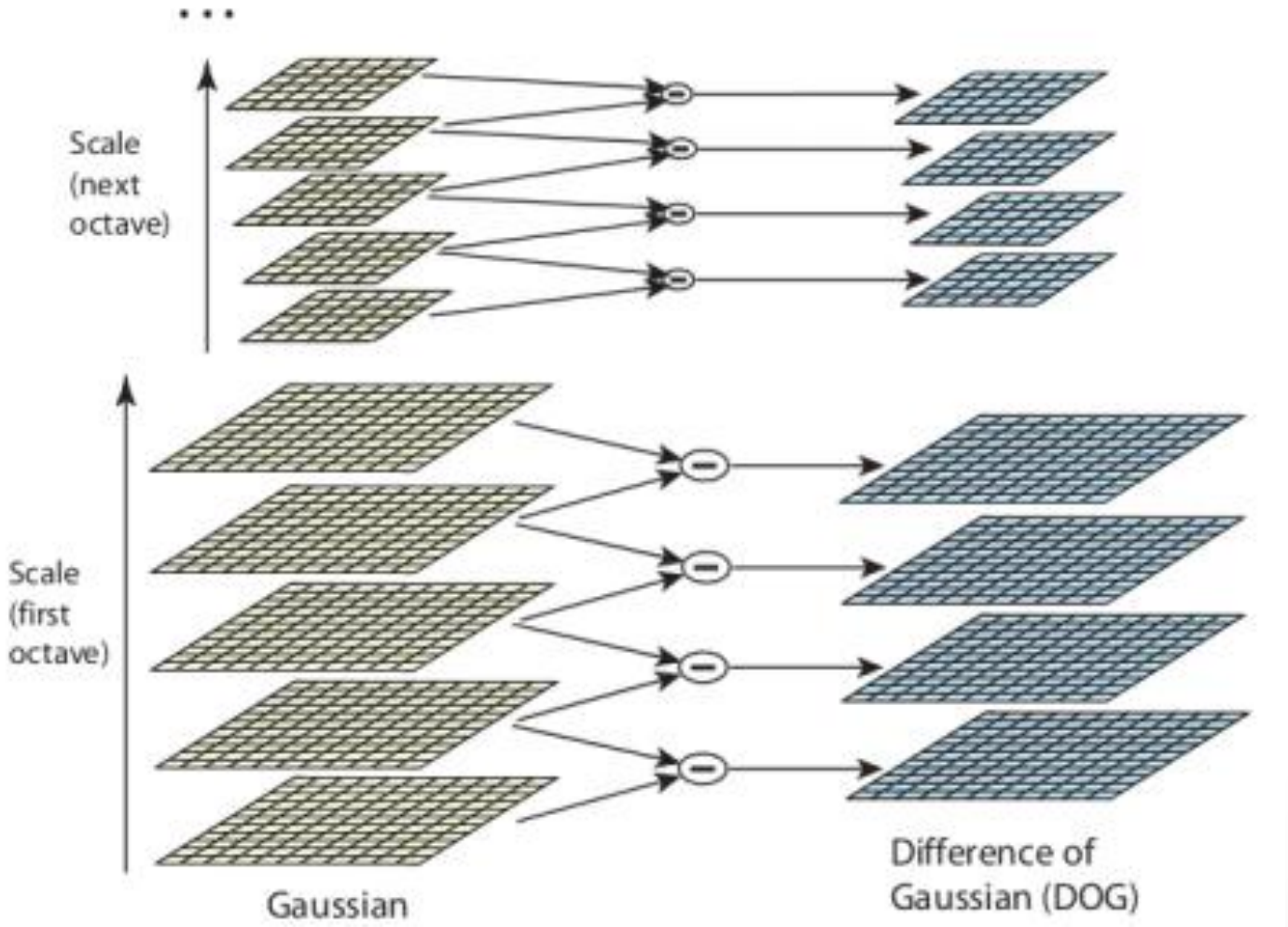
- Scale Invariant Feature Transform (**SIFT**)
  - Para extrair keypoints
  - e seus descritores

D. Lowe. Distinctive Image Features from Scale-Invariant Keypoints

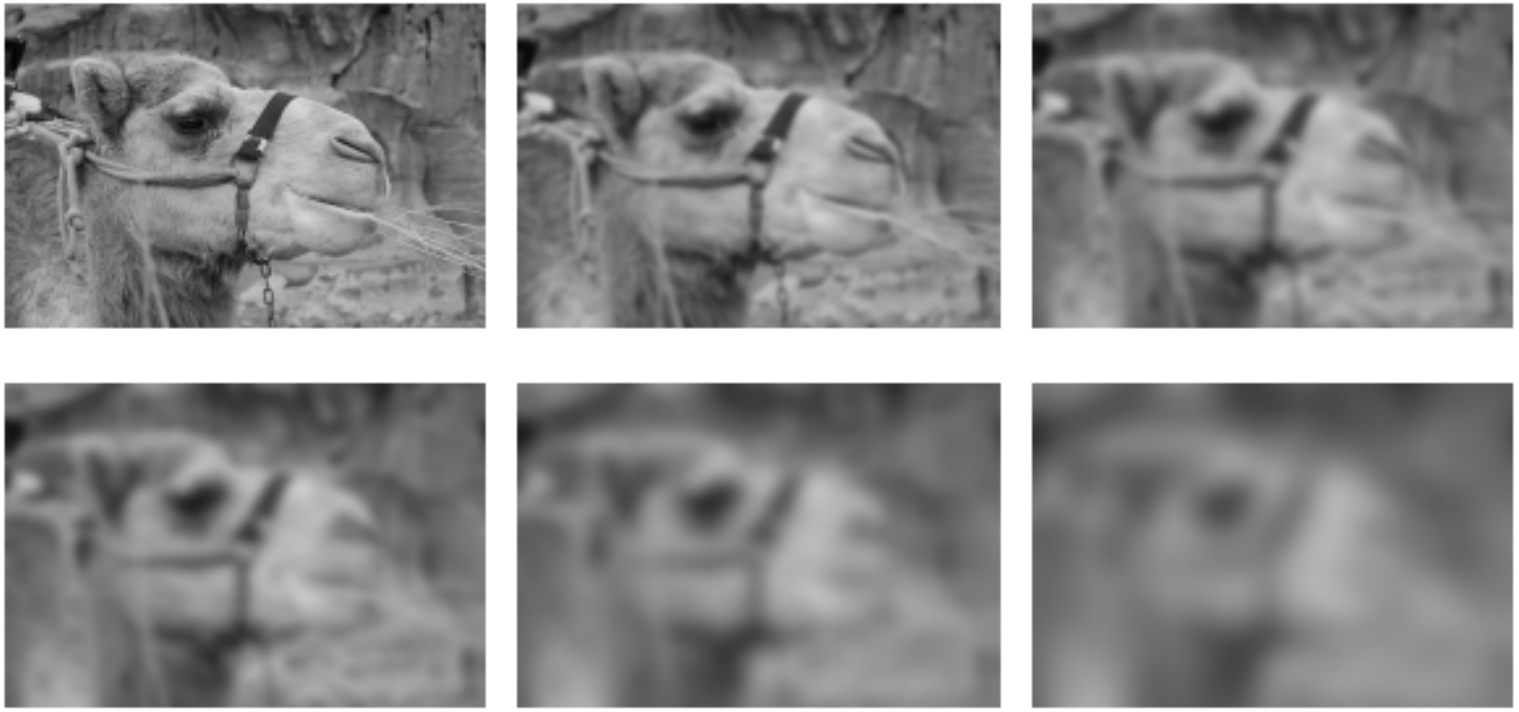
SIFT Príncipio

- Scale-Space Filtering
  - Aplica Differece of Gaussians (DoG) -> aproximação de LoG e mais rápido
  - Efeito: borramento da imagem
    - Quanto maior o borramento, existe a simulação de baixa escala
  - A diferença é calculada sobre imagens que passaram por gaussianas σ e kσ

DoG


Scale
(next
octave)
Scale
(first
octave)
Gaussian
Difference of
Gaussian (DOG)

Espaço de Escala - exemplo



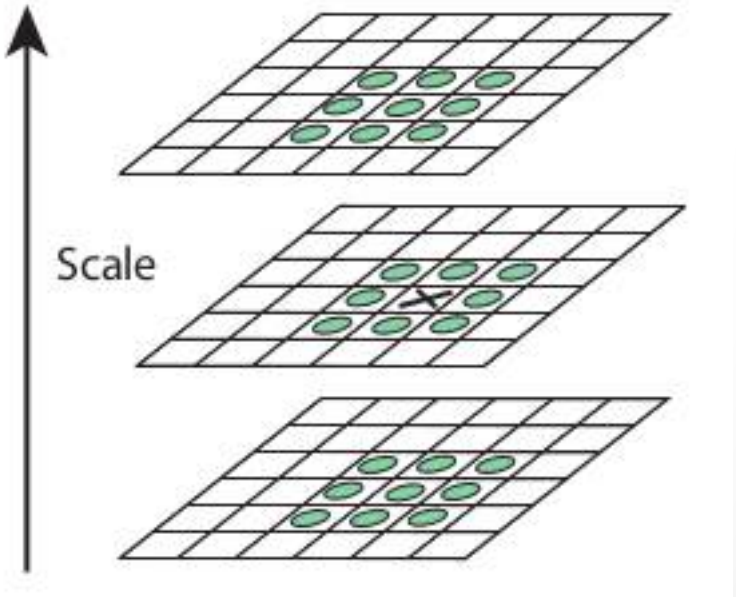
DoG

- A quantidade de imagens no Octave é informada por um constante s.
- k cresce ao quadrado ao fator $k=2^{1/8}$
- A quantidade total de imagens na pirâmide é sempre s = s+3

- No caso anterior foram usados:
- $k=2^{1/8}$, $2^{2/8}$, $2^{4/8}$ e $2^{8/8}$

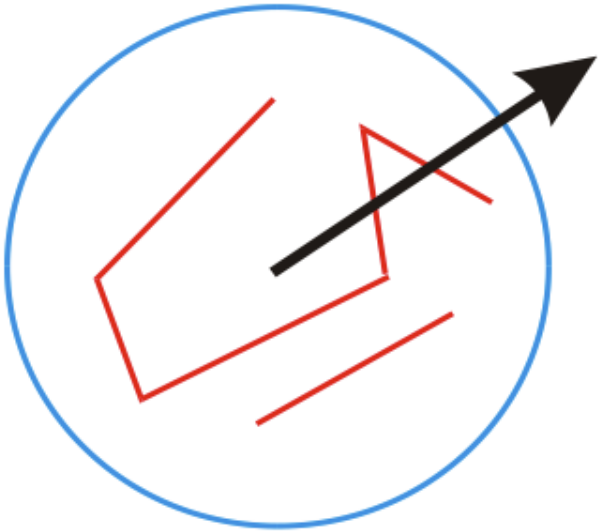
Local Extrema

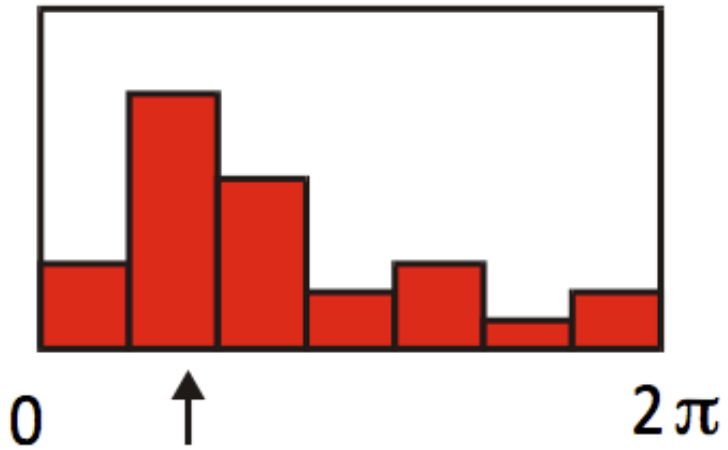
- Com a DoG construída, procura-se pelos pontos de extrema
- Esse ponto é maior que os seus 27 vizinhos de diferentes escalas
- Representa um **possível keypoint**



Invariante a Rotação

- A orientação do **possível keypoint** é obtida através da análise da orientação dos contornos próximos
- Orientação do keypoint = pico de orientação no histograma direcional ponderado pela magnitude





Invariante a Rotação

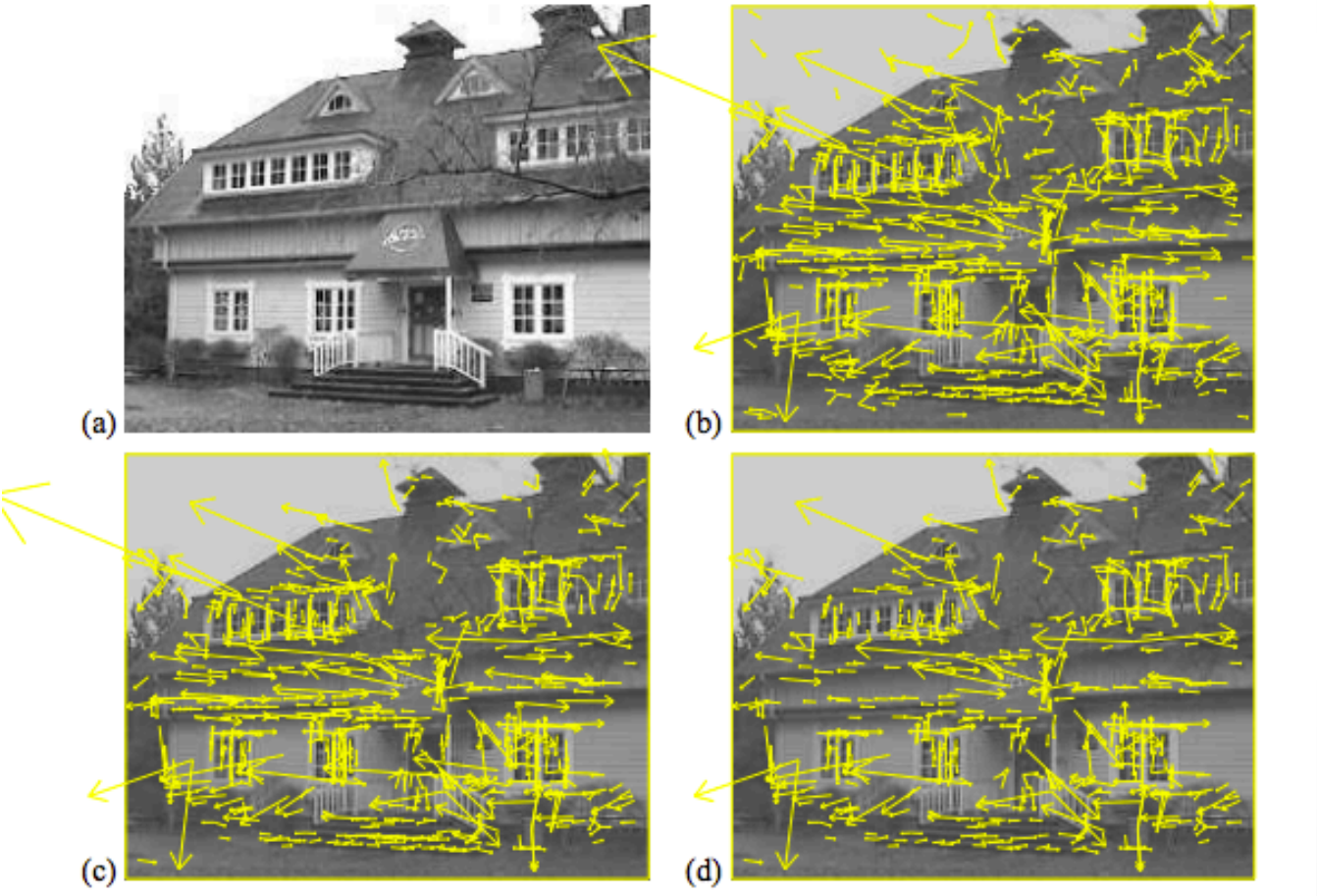
- Todas as características serão descritas baseadas nessa orientação

- Quando um ponto aparecer em outra imagem poderá ser comparado de acordo com a orientação que aparece

Localização dos Keypoints

- Verifica os pontos de extremo localizados
- Dois tipo são removidos:
  1. Pontos de baixo contraste: threshold 0.03
  2. Pontos de borda (DoG tem alta resposta para borda): ratio of principal curvatures (mesmo do Harris – autovalores da hessiana), valor 10

Localização dos Keypoints



c) após redução por contraste
d) após redução por ratio of principal curvatures

Descritor SIFT

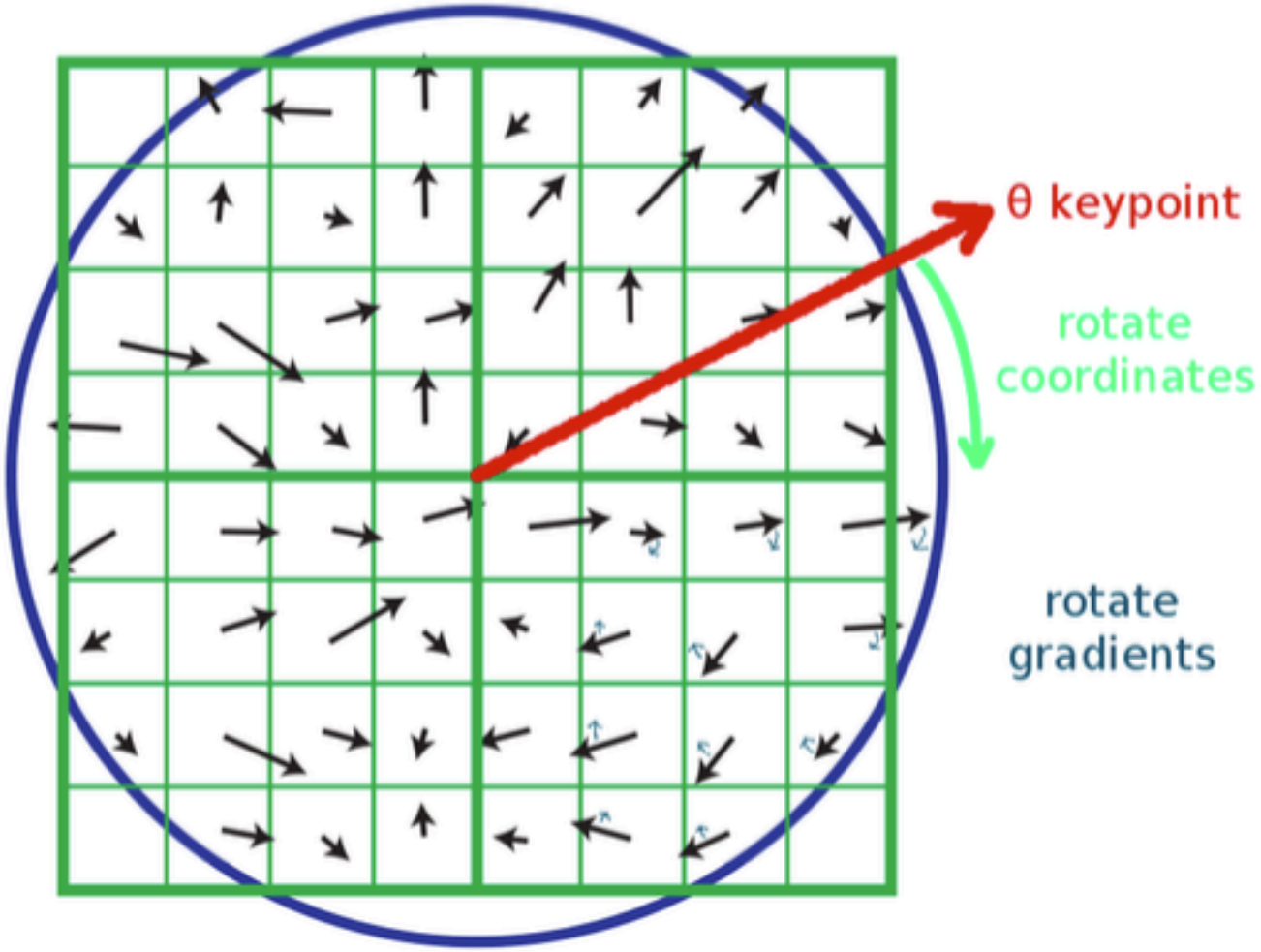
- O descritor consiste num **vetor de características** obtidas ao redor do keypoint localizado
- Para que seja corretamente interpretado deve ser calculado:
  1. Na escala na qual foi obtido o ponto
  2. Obedecer as regras de invariante a rotação

Descritor SIFT

- Consiste num **vetor de orientações** (gradientes) – histograma direcional
- Calculado ao redor do keypoint, numa janela 4x4

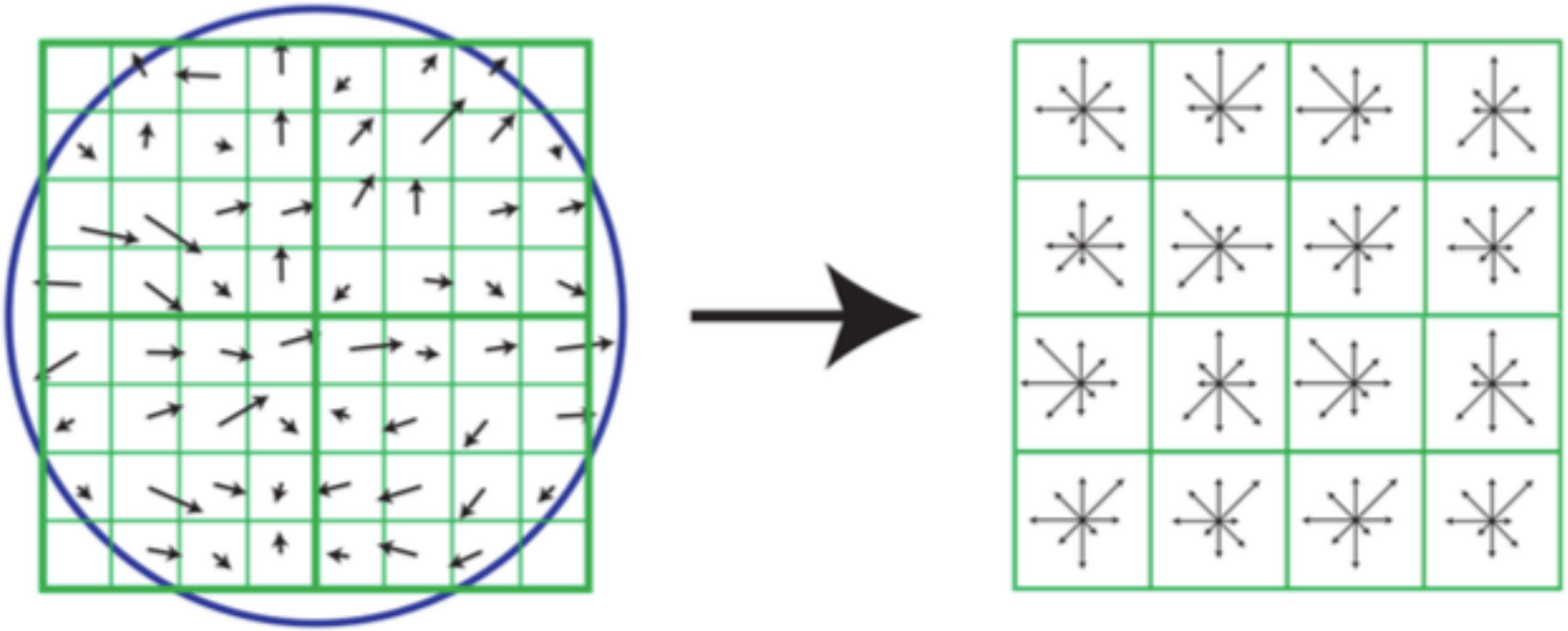
- Todas as **orientações** são **rotacionadas** na **direção** do keypoint para anular a rotação externa do objeto

Descritor SIFT



Descitor SIFT

- Melhores resultados com histograma orientado 4x4 de 8bins
- Total: 128 = 4 x 4 x 8 = características



O que fazer com o vetor?

- Matching!!

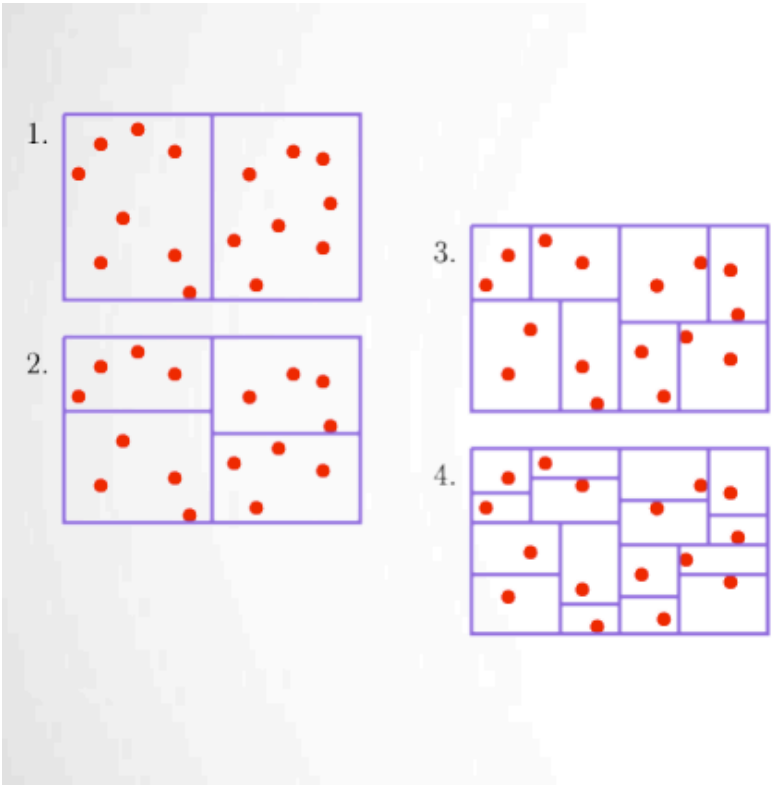
- Pode comparar o vetor localizado num ponto de uma imagem com o vetor de outro ponto em outra imagem
- Se forem semelhantes = match!

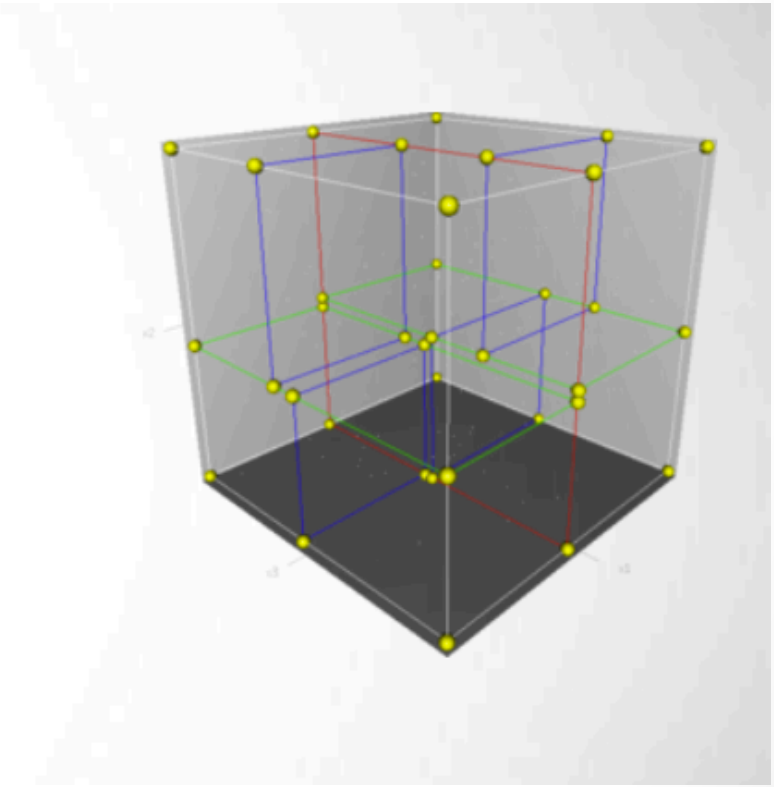
- Similaridade pode ser obtida com distancia euclidiana

Matching

- Normalmente com distancia euclidiana
- Resultados guardados numa KD-Tree (128 dimensões)
  - KNN – para classificação
  - Baseada no ponto que possui maior similaridade
- Normal no SIFT buscar ainda dois vizinhos a similaridade

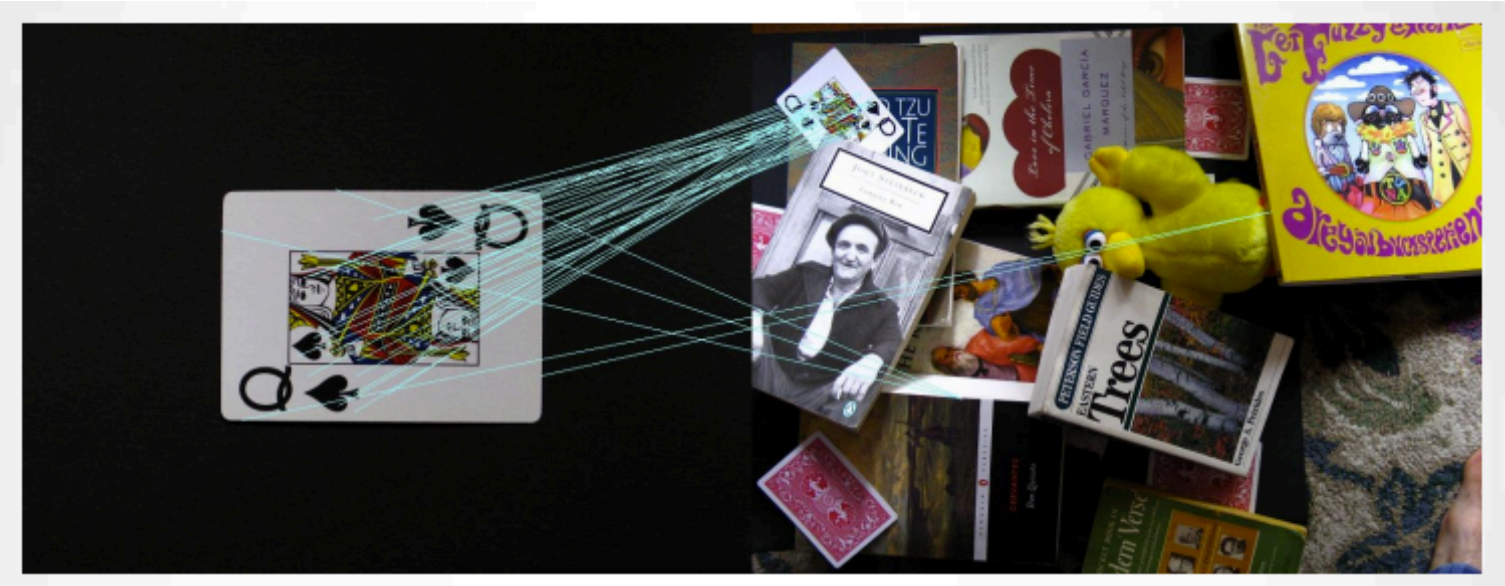
Espaço de Matching - vizinhança



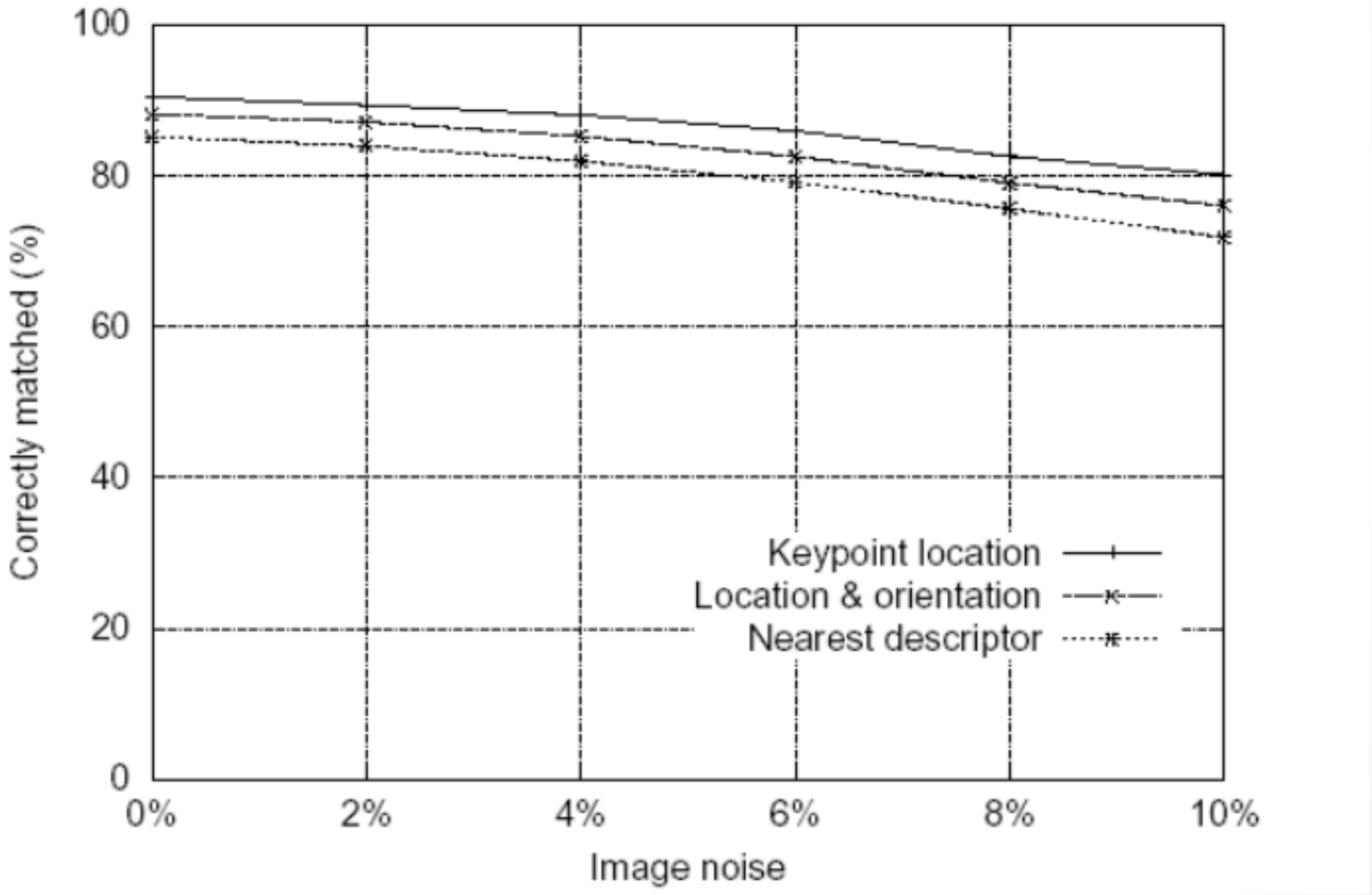


Processo

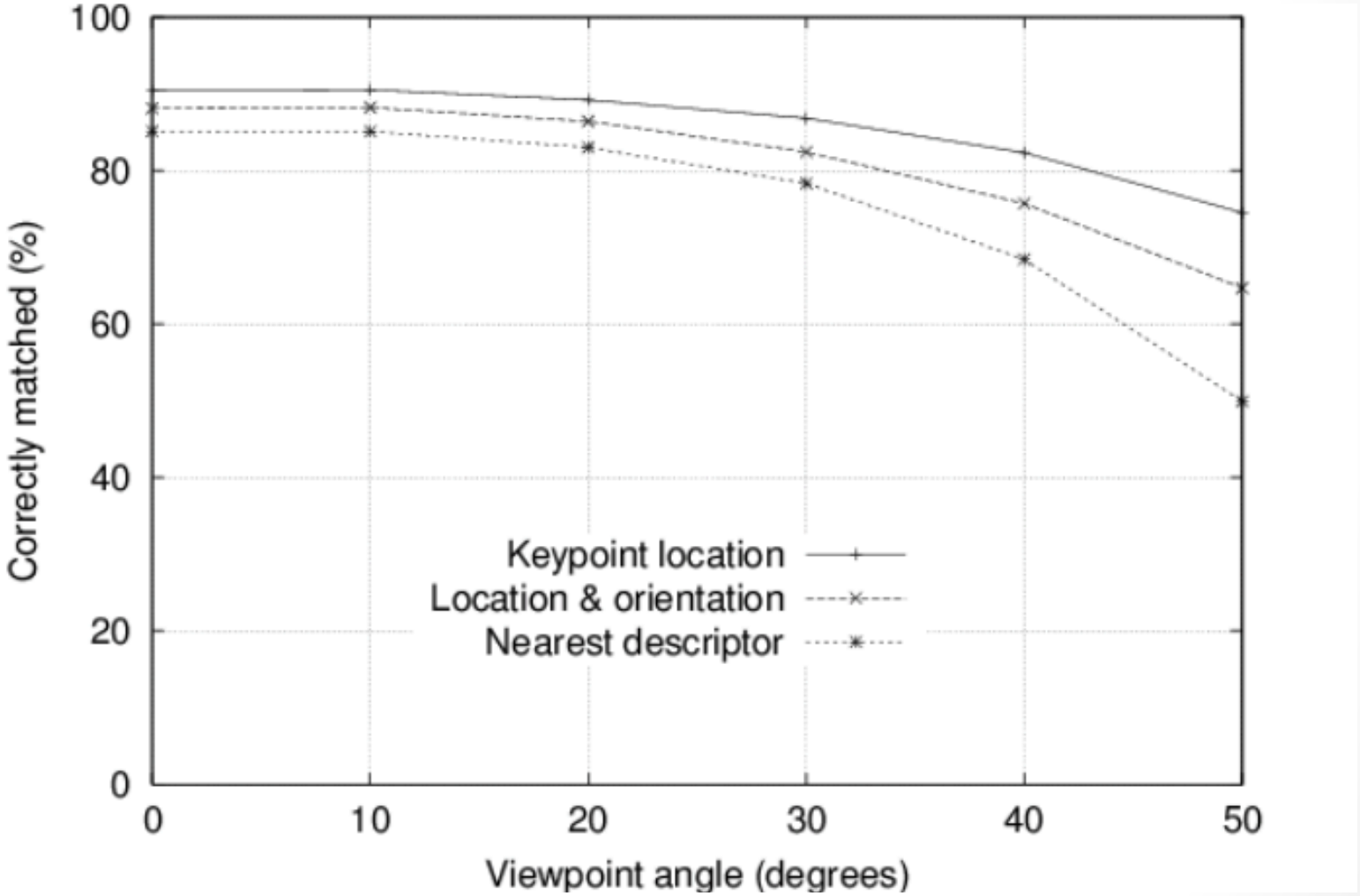
1. Constroi kd-tree do template
2. Busca os 2 vizinhos mais próximos para cada ponto da imagem
3. Seleciona o melhor casamento (obtiver a maior similaridades)



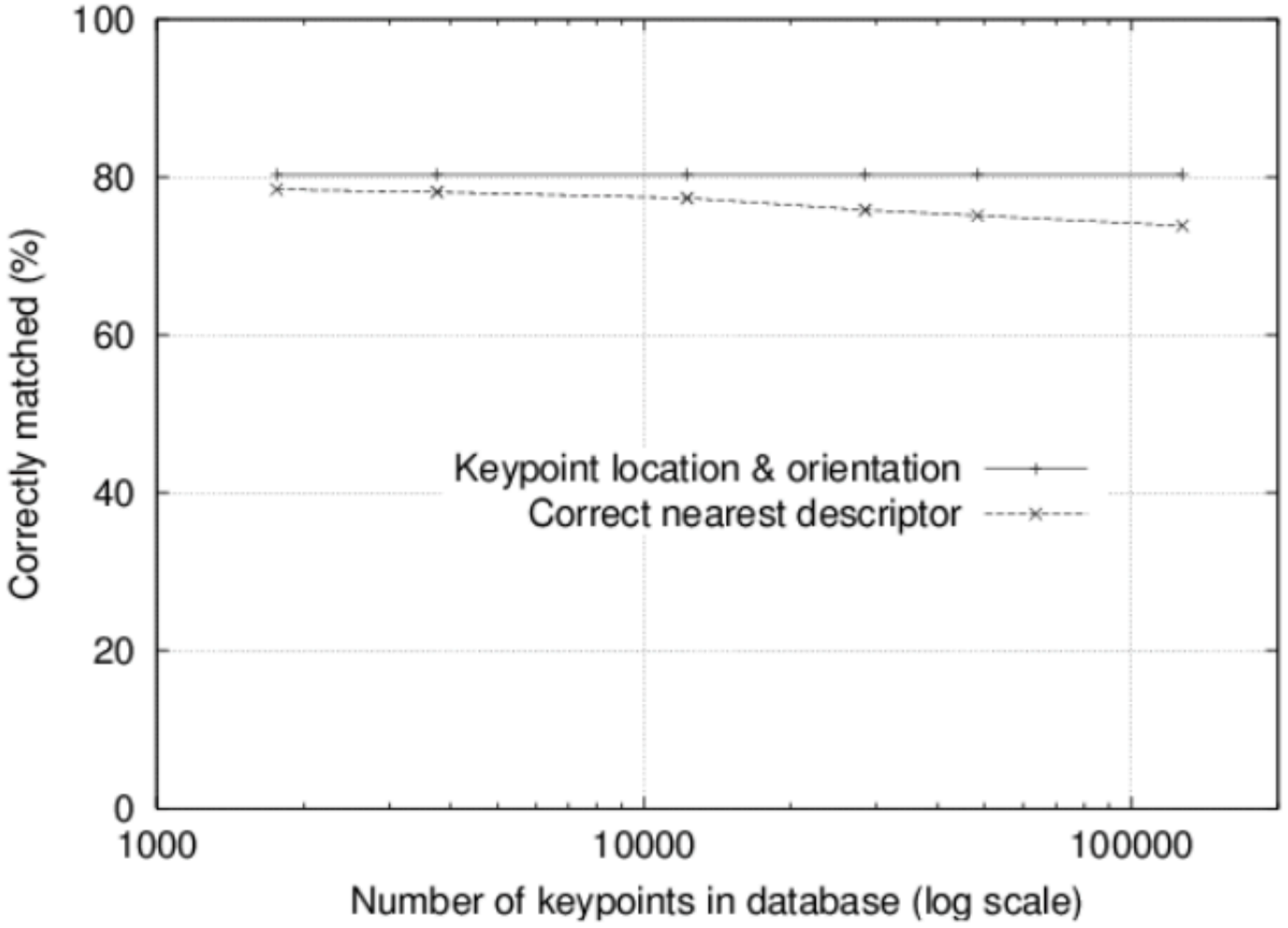
Estabilidade



Estabilidade

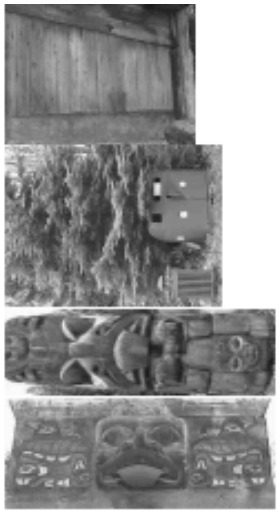


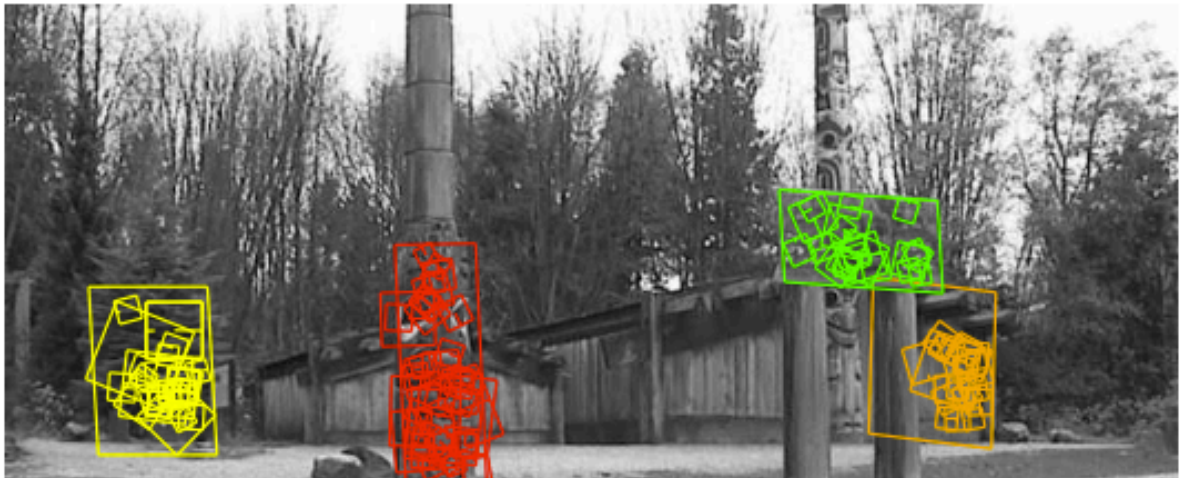
Estabilidade



Exemplos

Exemplos





Aplicações: placas de transito

Aplicações: mosaico

No opencv

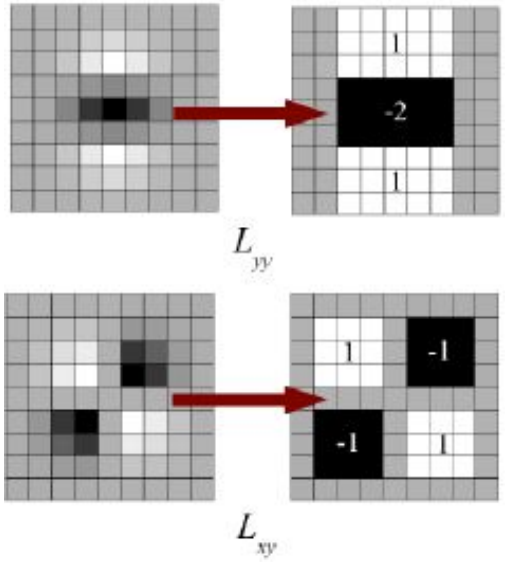
```
Ptr<Feature2D> f2d = xfeatures2d::SIFT::create();

std::vector<KeyPoint> keypoints_1, keypoints_2;
f2d->detect( src, keypoints_1 );
f2d->detect( proc, keypoints_2 );
Mat descriptors_1, descriptors_2;
f2d->compute( src, keypoints_1, descriptors_1 );
f2d->compute( proc, keypoints_2, descriptors_2 );

BFMatcher matcher(NORM_L2);
std::vector< DMatch > matches;
matcher.match( descriptors_1, descriptors_2, matches );
drawMatches( src, keypoints1, proc, keypoints2, matches, drawImg);
```

SURF

Speed-Up Robust Features

A ideia

- SIFT relativamente lento
- No SIFT:
  - Aproxima-se Laplaciano de Gaussiana com DoG
- No SURF
  - A aproximaçãoé feita com BoxFilter



Bay, H., Tuytelaars, T. and Van Gool, L, published another paper, "SURF: Speeded Up Robust Features"

Obtendo os pontos de interesse

- Baseado em Integral Images

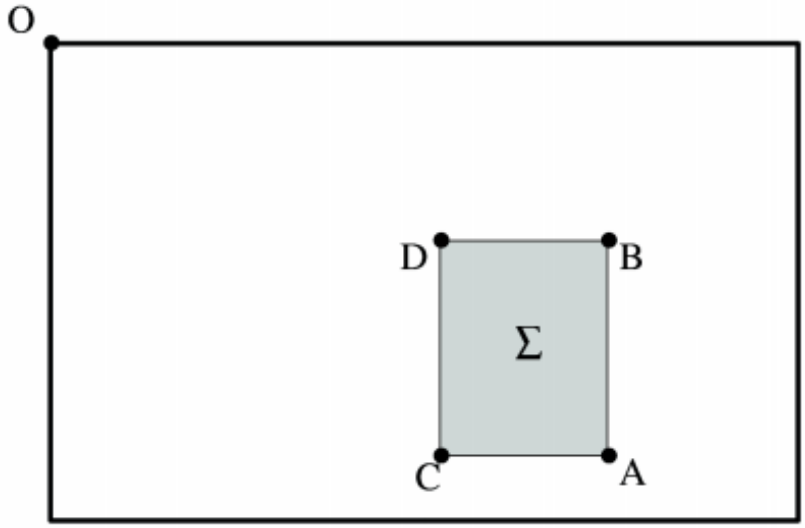
$$I_\Sigma(\mathbf{x}) = \sum_{i=0}^{i \le x} \sum_{j=0}^{j \le y} I(i, j) \qquad \mathbf{x} = (x, y)^\top$$

1	2	2	4	1
3	4	1	5	2
2	3	3	2	4
4	1	5	4	6
6	3	2	1	3

input image

0	0	0	0	0	0
0	1	3	5	9	10
0	4	10	13	22	25
0	6	15	21	32	39
0	10	20	31	46	59
0	16	29	42	58	74

integral image



Σ=A-B-C+D

Obtendo os pontos de interesse

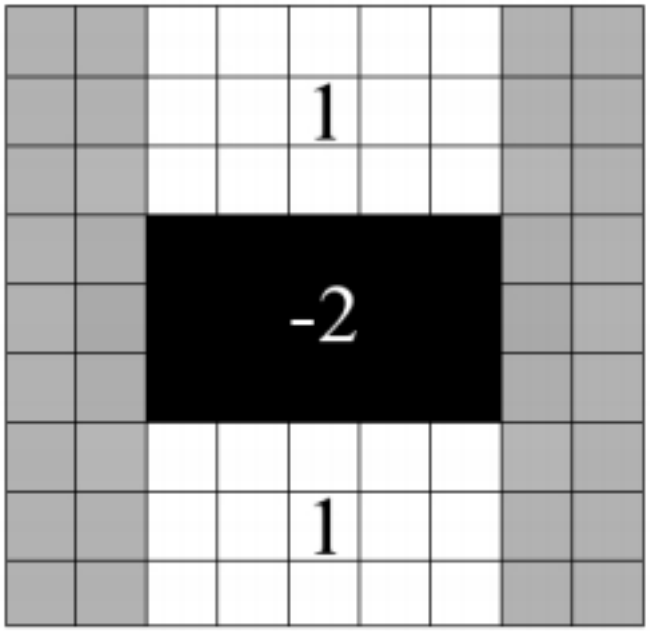
- Sobre imagens integral

$$\mathcal{H}(\mathbf{x}, \sigma) = \begin{bmatrix} L_{xx}(\mathbf{x}, \sigma) & L_{xy}(\mathbf{x}, \sigma) \\ L_{xy}(\mathbf{x}, \sigma) & L_{yy}(\mathbf{x}, \sigma) \end{bmatrix}$$

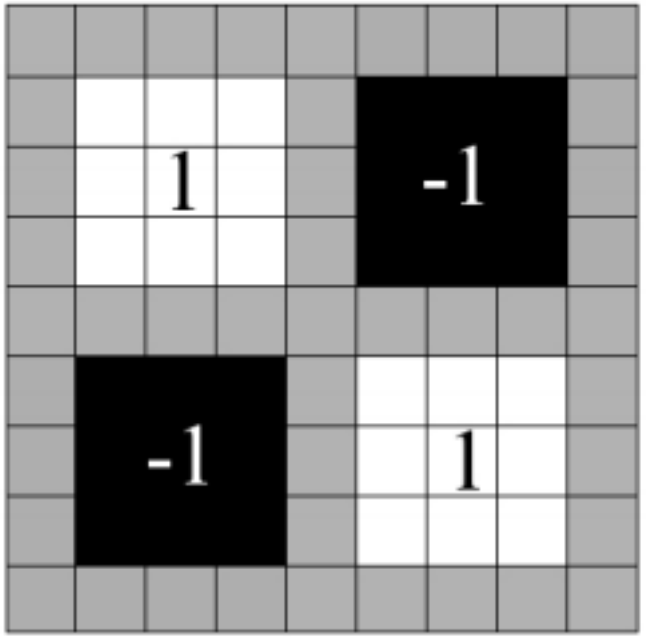
- Onde $L_{\{x|y|xx|yy\}}(x,\sigma)$ são a convolução da imagem com a gaussiana de segunda ordem
- a hesssiana informa: **altos valores** (máximos – hills ou mínimos - valleys)

Obtendo os pontos de interesse

- Essa gaussiana de segunda ordem é aproximada por box filter



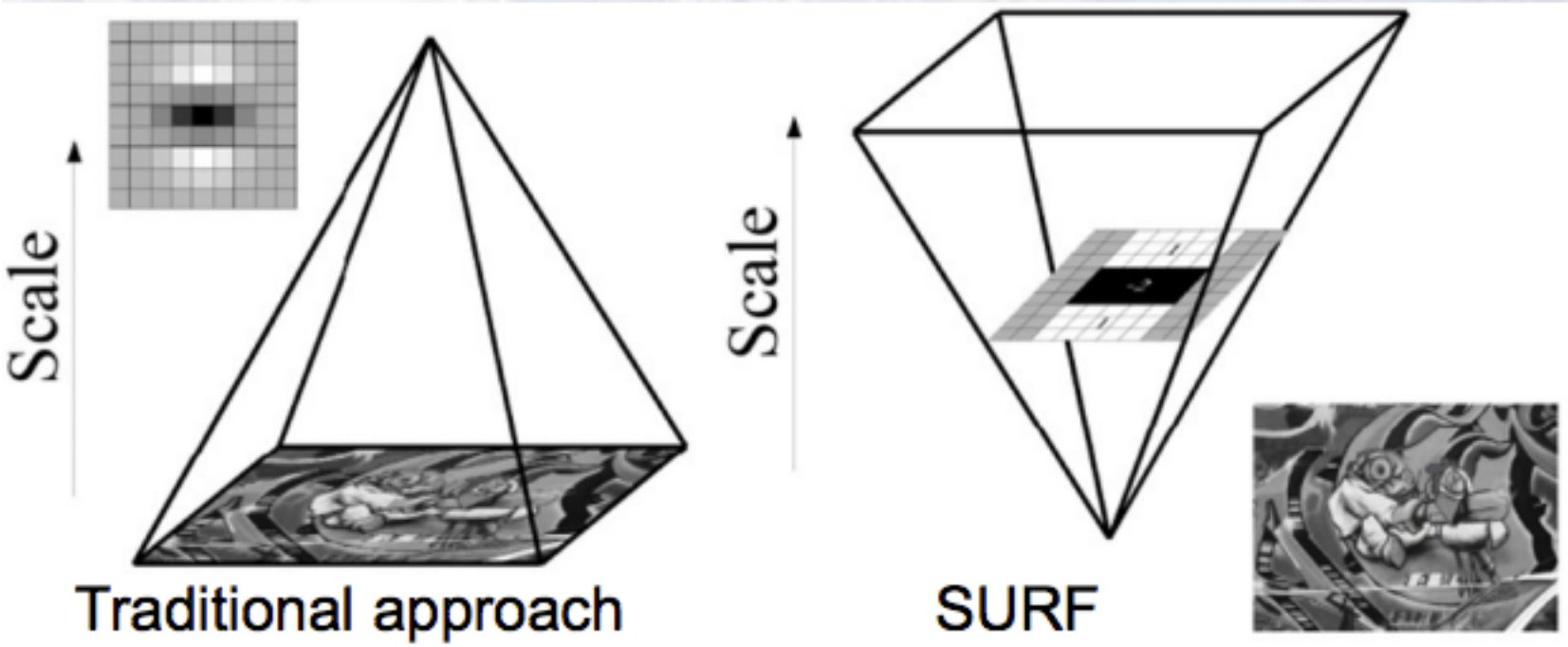
Direção Y



Direção XY

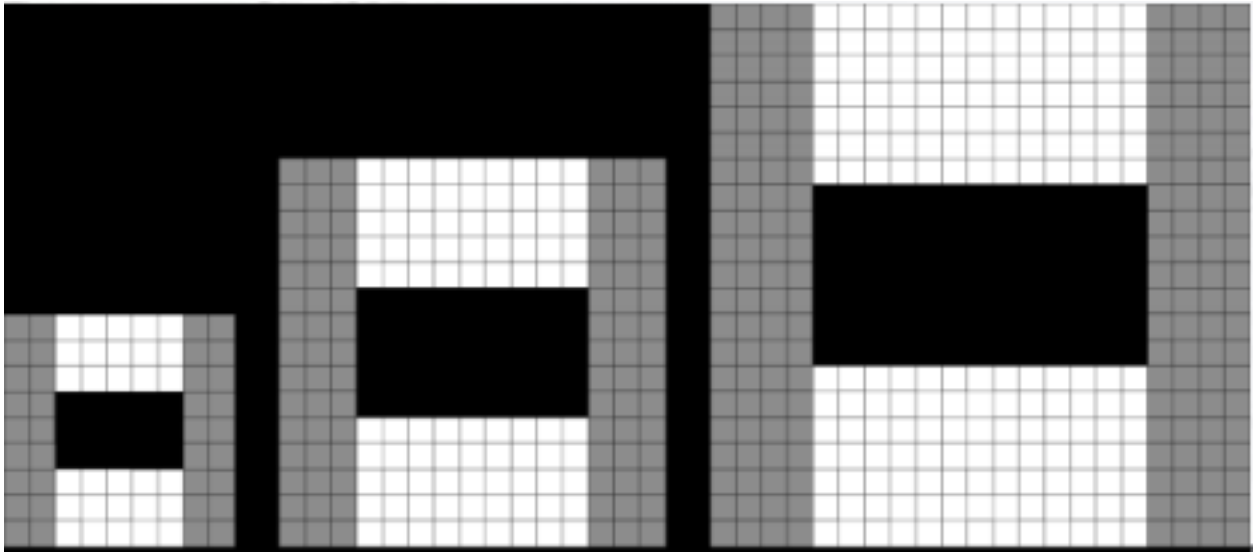
Obtendo os pontos de interesse

- Multiplas escalas, escalando o box filter



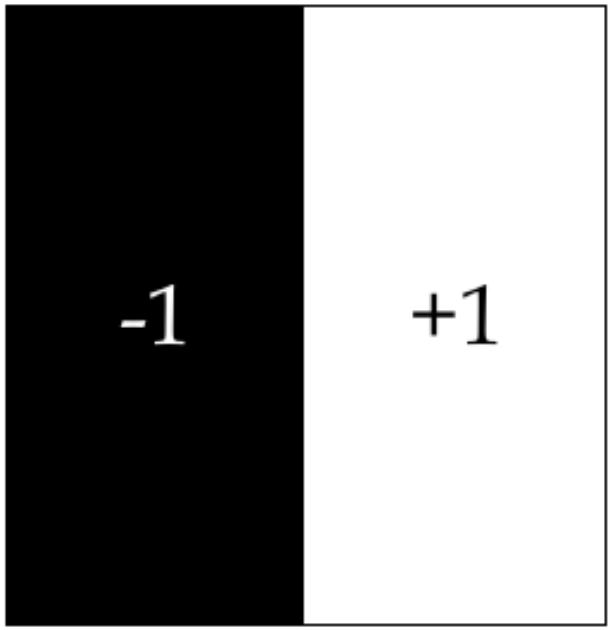
Obtendo os pontos de interesse

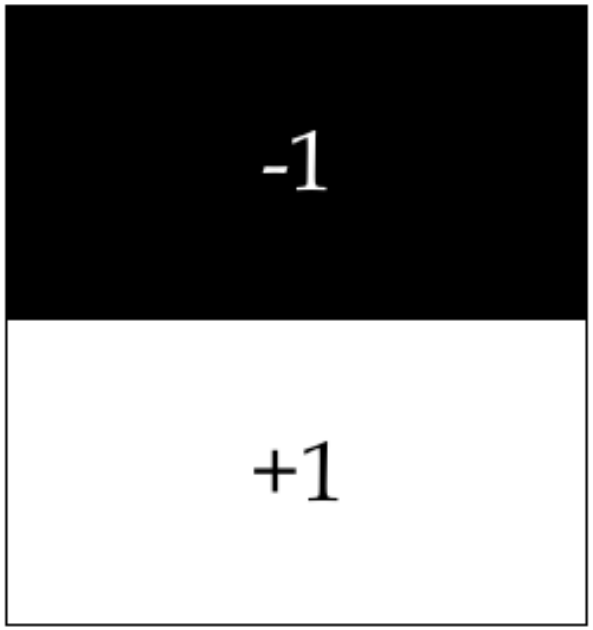
- Os filtros são organizados em octaves
- Começam com dimensão 9x9 (equivale a σ=1.2)
- Incrementos são na ordem de 6



Descritores

- Contêm basicamente a resposta a primeira ordem de Haar Waveltes
  - Calculados em x e y (acha gradientes)
  - Novamente sobre as imagens integral





Descritores

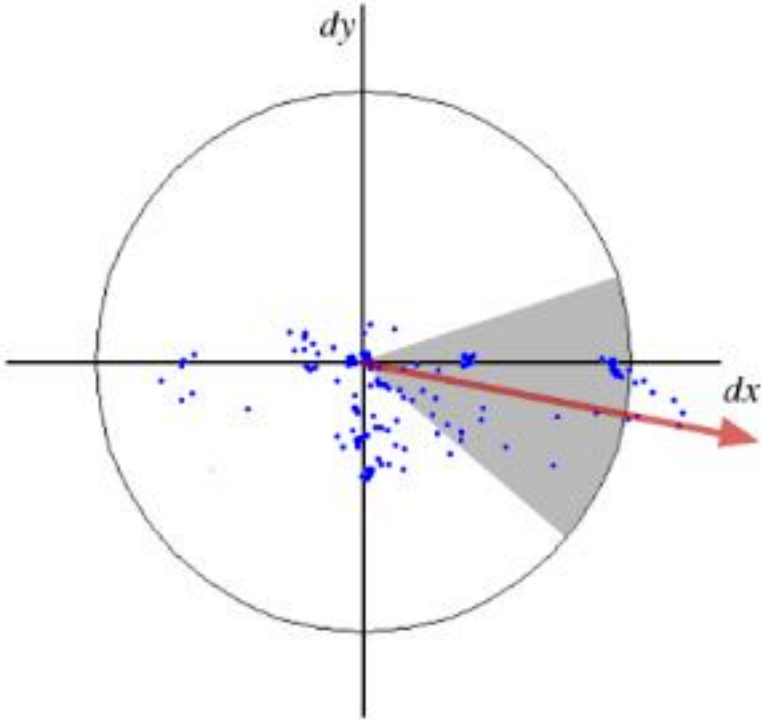
- O tamanho da janela onde será extraído a informação é sempre 20 vezes o tamanho da escala (20σ)
- Essa região é divida em 4x4
- Para cada região, Haar wavelet é calculada
- As respostas são somadas

$$\left[\sum dx \quad \sum dy \quad \sum |dx| \quad \sum |dy|\right]$$

- Totalizando 64 características : 4x4 = 16 * 4 = 64
- *Existe uma implementação que separa o sinal do gradiente totalizando 128 características

Orientação

- As respostas do haar são levadas em consideração
- Numa janela de tamanho 6σ ao redor do ponto
- A direção predominante (num intervalo π/3) será escolhida



No opencv

```
Ptr<SURF> detector = SURF::create();

std::vector<KeyPoint> keypoints_1, keypoints_2;
Mat descriptors_1, descriptors_2;
detector->detectAndCompute( src,  Mat(), keypoints_1, descriptors_1 );
detector->detectAndCompute( proc, Mat(), keypoints_2, descriptors_2 );

BFMatcher matcher(NORM_L2);
std::vector< DMatch > matches;
matcher.match( descriptors_1, descriptors_2, matches );
drawMatches( src, keypoints1, proc, keypoints2, matches, drawImg);
```

Outras boas alternativas

- ORB (FAST + BRIEF): oriented BRIEF (Binary robust independent elementary features)
- KAZE
- BRISK: Binary Robust Invariant Scalable Keypoints
- FREAK: Fast Retina Keypoint