Feature Detector

Prof. Dr. Geraldo Braz Junior

Problema: Matching

Matching

Matching

- Processo de encontrar uma imagem em outra
- Normalmente usado para encontrar um objeto numa imagem, mas também:
  - Reconhecimento
  - visão robótica
  - mosaic
- Abordagem
  - Local
  - Global

Porque usar abordagem local?

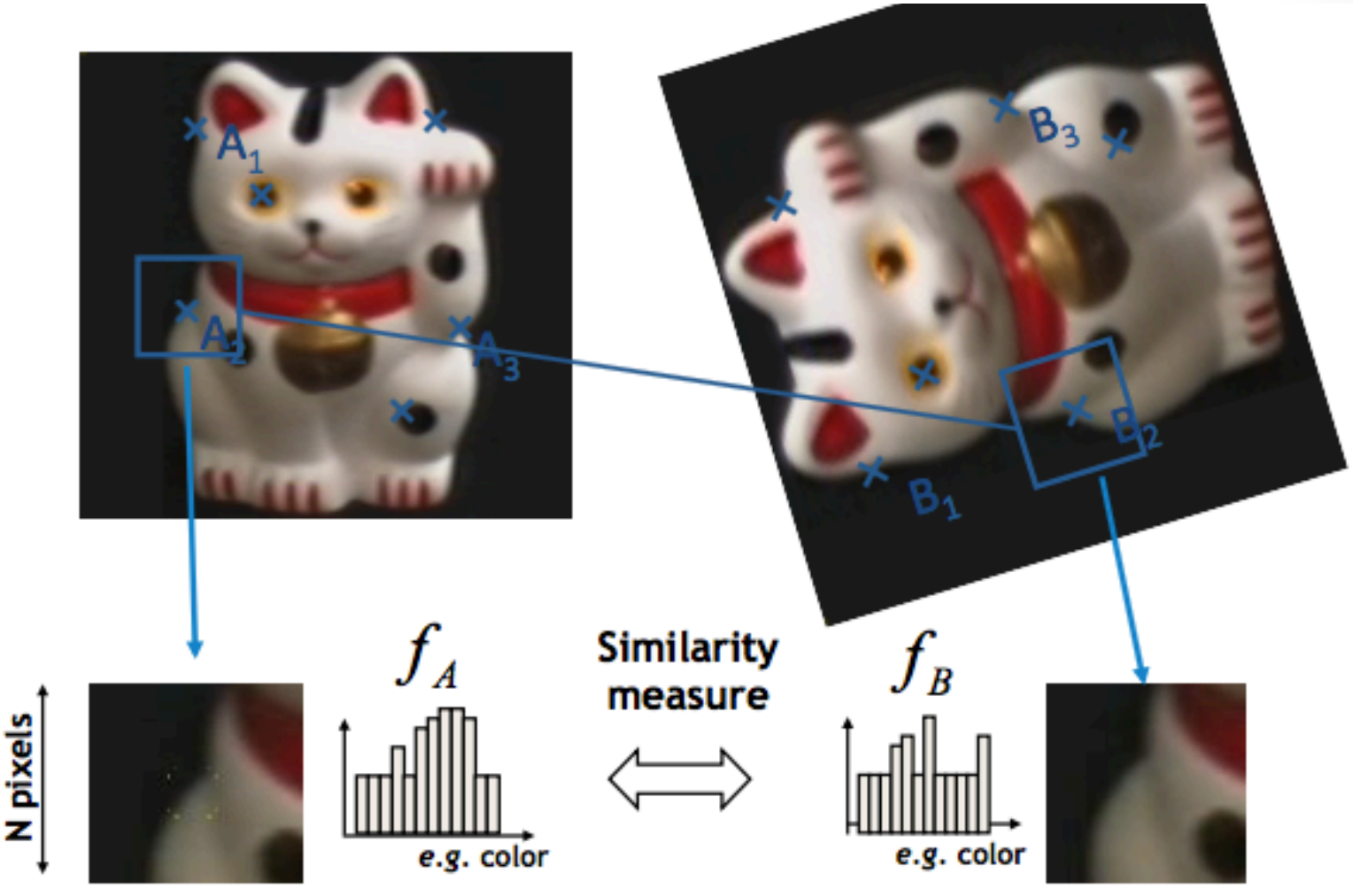
- Match em regiões localizadas permitem:
  - resolver problemas de oclusão
  - pequenas variações de textura podem ser descartadas ou tratadas
  - assim como variações de iluminação, rotação e escala

Abordagem básica: Passos

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

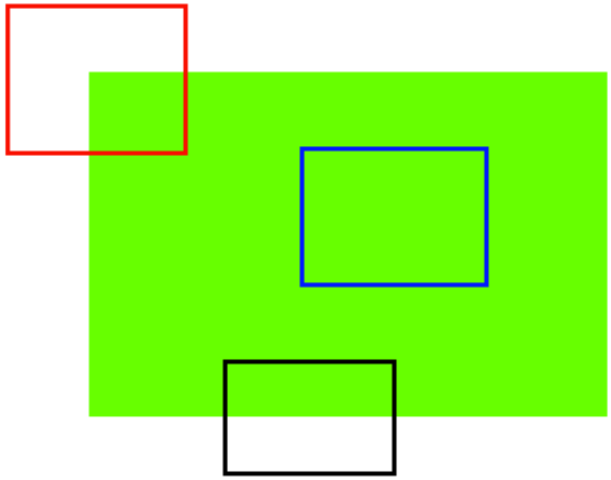
Keypoints

- Pontos Chave e **Importantes**!
- Independente da imagem, o algoritmo extrator deve extrair os keypoints sempre da mesma maneira
  - Consistência
  - Repetição

Características para o método

- Repetição (independência de imagem)
- Invariante: translação, escala e rotação
- Robusto
- Eficiente
- Discriminativo
- Quantitativo
- Local

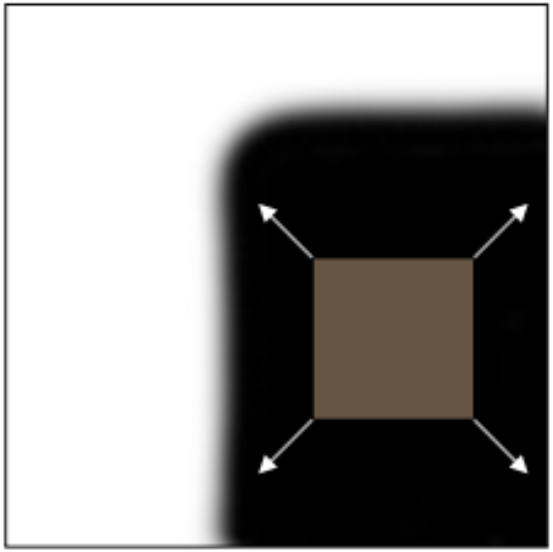
O que são bons candidatos a keypoints?



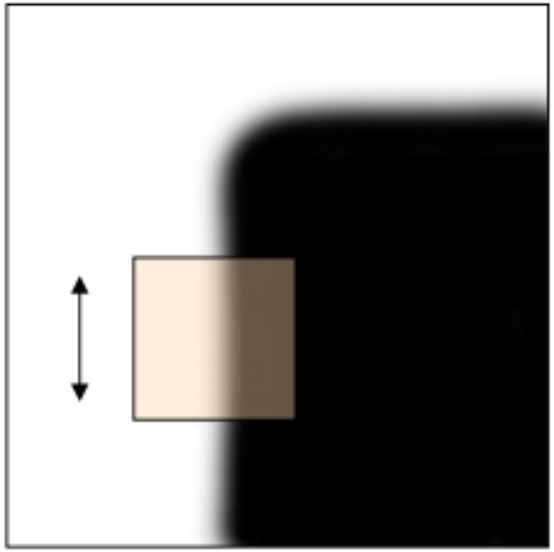
Bons candidatos

- A keypoints:
  - **Cantos**!
  - Dentro de um canto, o gradiente do ponto tem duas ou mais direções dominantes
  - São repetíveis e discriminantes
- Feature Detector:
  - Harris
  - FAST
  - SIFT
  - SURF

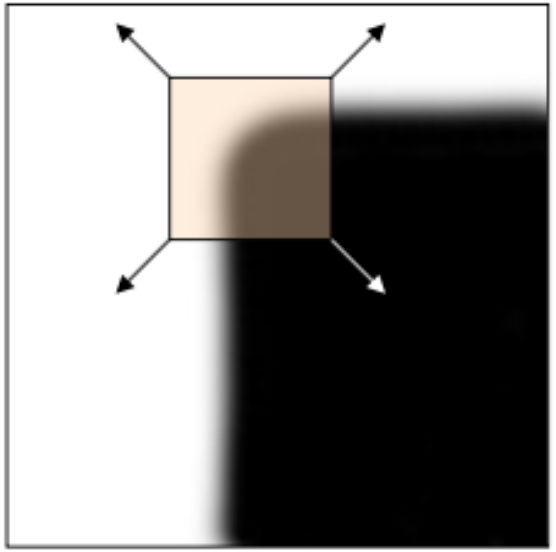
Distinção



Interior: sem variações em todas as direções



borda: variações na direção da borda



canto: mudanças bruscas em todas as direções

Harris Detector

- Ideia da janela
- Maximizar

$$E(u,v) = \sum_{x,y} w(x,y)\left[I(x+u,y+v) - I(x,y)\right]^2$$

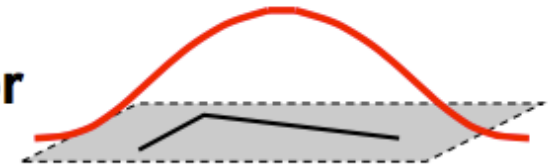
Window function

Shifted intensity

Intensity

Window function $w(x,y)$ =

or



1 in window, 0 outside

Gaussian

Harris Detector

- Pode ser aproximada por:

$$E(u,v) \approx [u \;\; v] \; M \begin{bmatrix} u \\ v \end{bmatrix}$$

- onde M significa

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

- I_x e I_y são os gradientes em x e y

- Essa é uma matriz de covariância

Harris Detector

$$M = \begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} = \sum \begin{bmatrix} I_x \\ I_y \end{bmatrix} [I_x \; I_y]$$

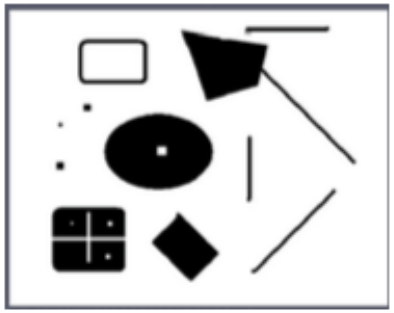
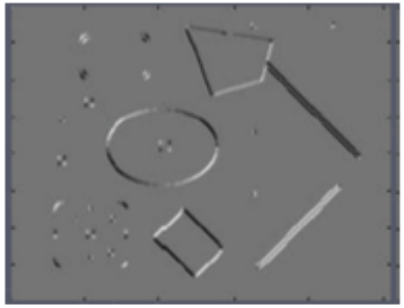


Image I

I_x

I_y



$I_x I_y$

Pode-se usar Sobel para calcular o gradiente

Harris Detector

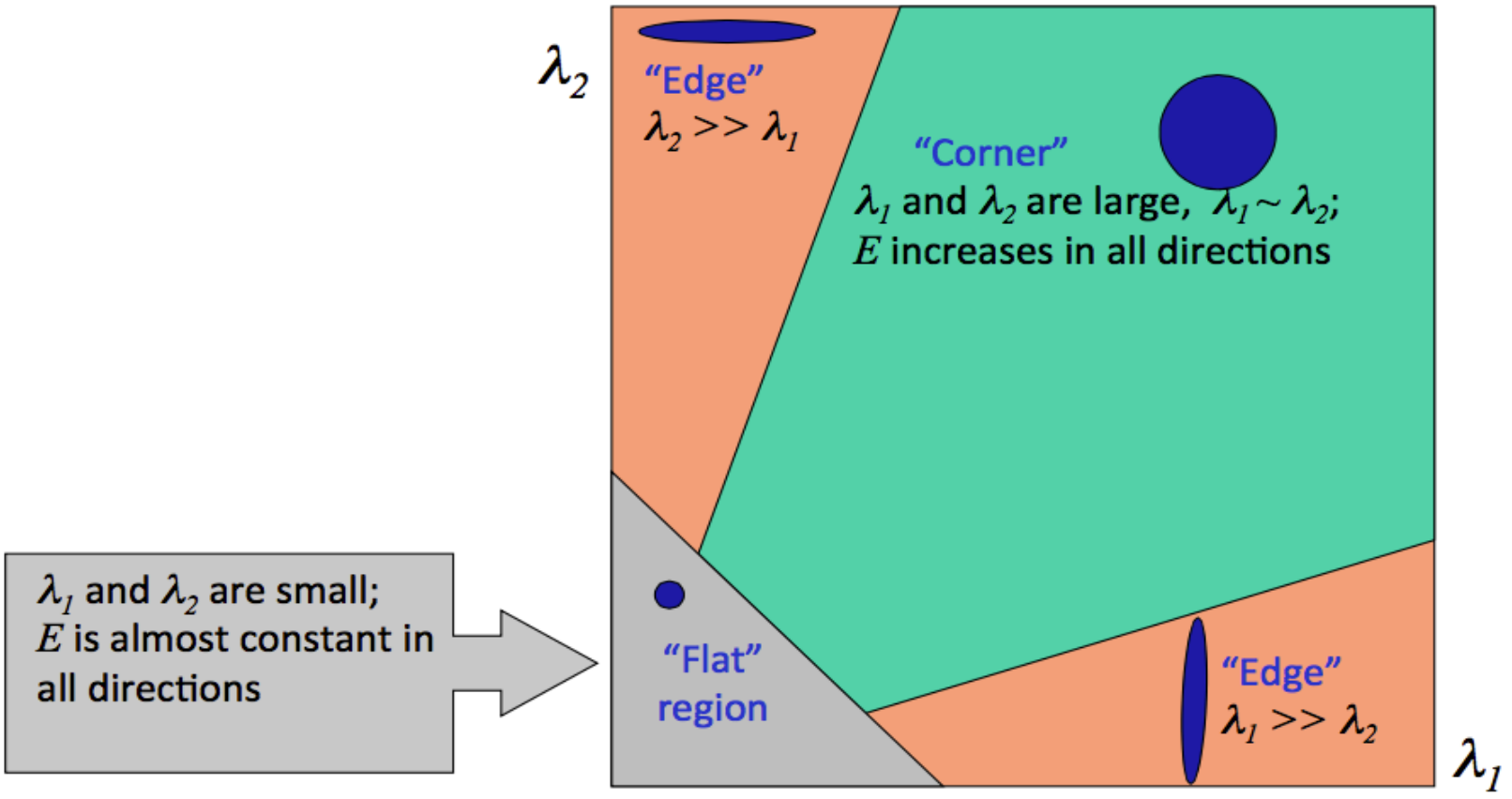
- A equação anterior gera um score de cada janela testada.
- Para saber se a janela possui um canto:

$$R = \det(M) - k(trace(M))^2$$

- onde:
  - $\det(M) = \lambda_1\lambda_2$
  - $trace(M)=\lambda_{1+}\lambda_2$
  - k é uma constante que varia entre 0.04 e 0.06
  - λ_1 e λ_2 são os autovalores de M

Harris Detector

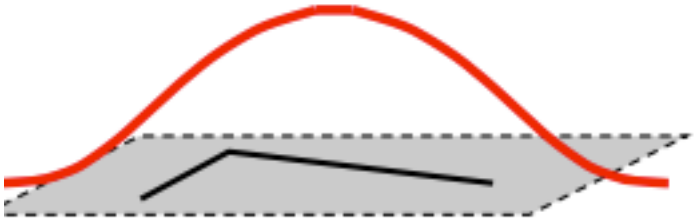
- Os valores dos autovalores decidem se a determinada janela possui ou não um canto:
  - quando R é grande (que acontece quando λ_1 e λ_2 são grandes) então a janela possui um canto



Harris Detector

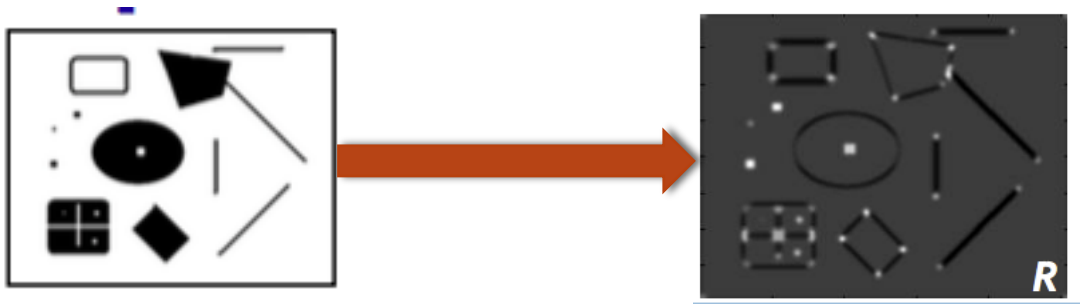
- Função Janela
  - Utiliza-se Gaussiana para tornar o resultado invariante a rotação
  - A função Gaussiana já aplica uma soma ponderada aos termos

$$M = g(\sigma) * \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$



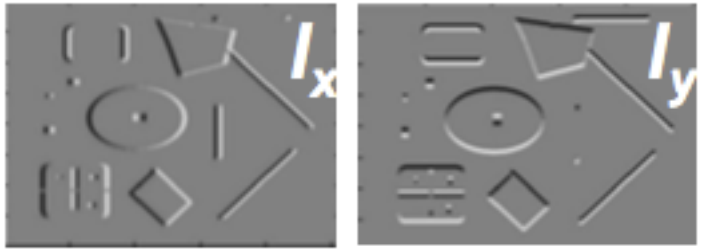
Gaussian

Passo a Passo



- Threshold do máximo
ou
- non-maximum supression

1. Image derivatives

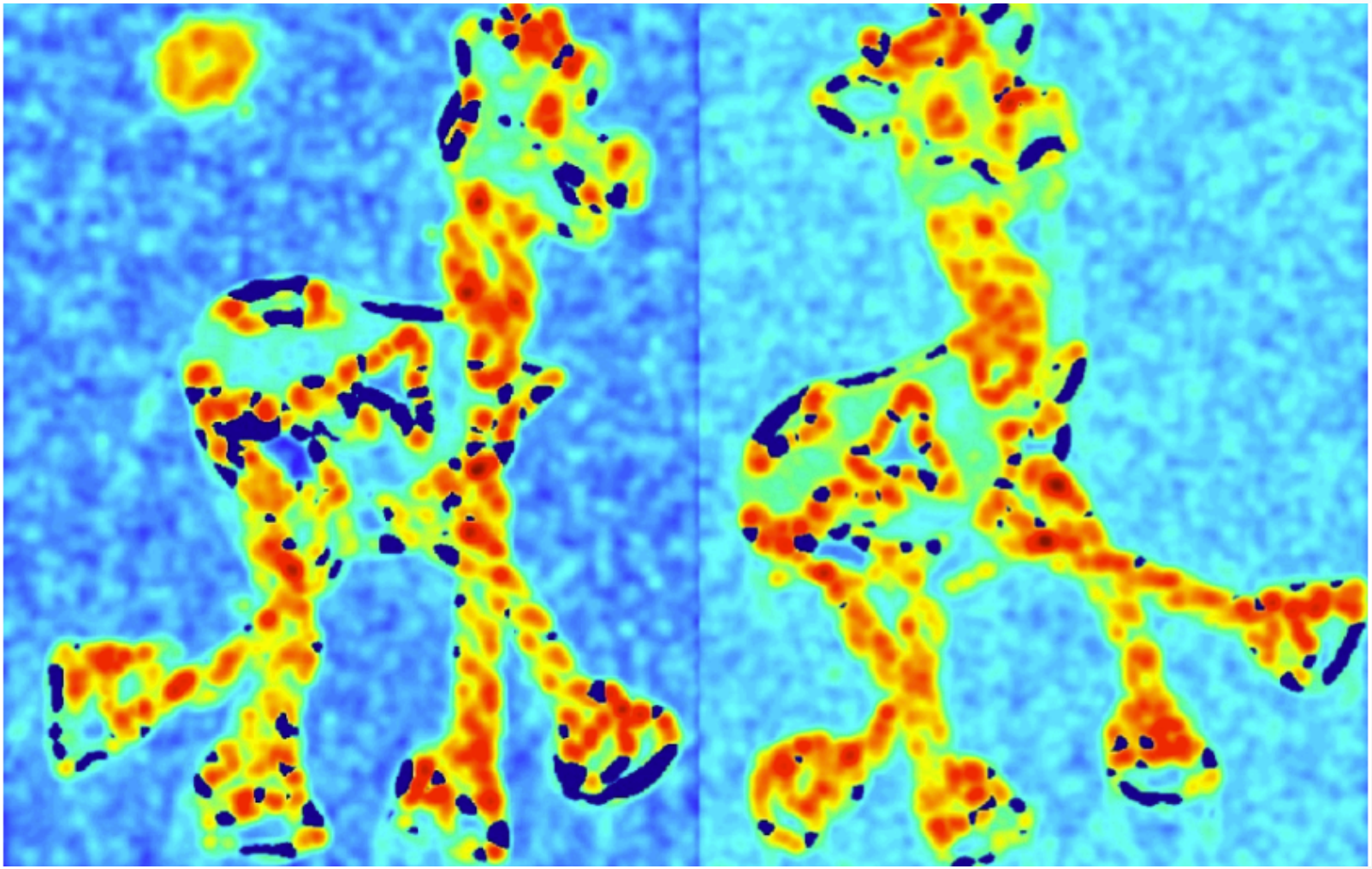


2. Square of derivatives

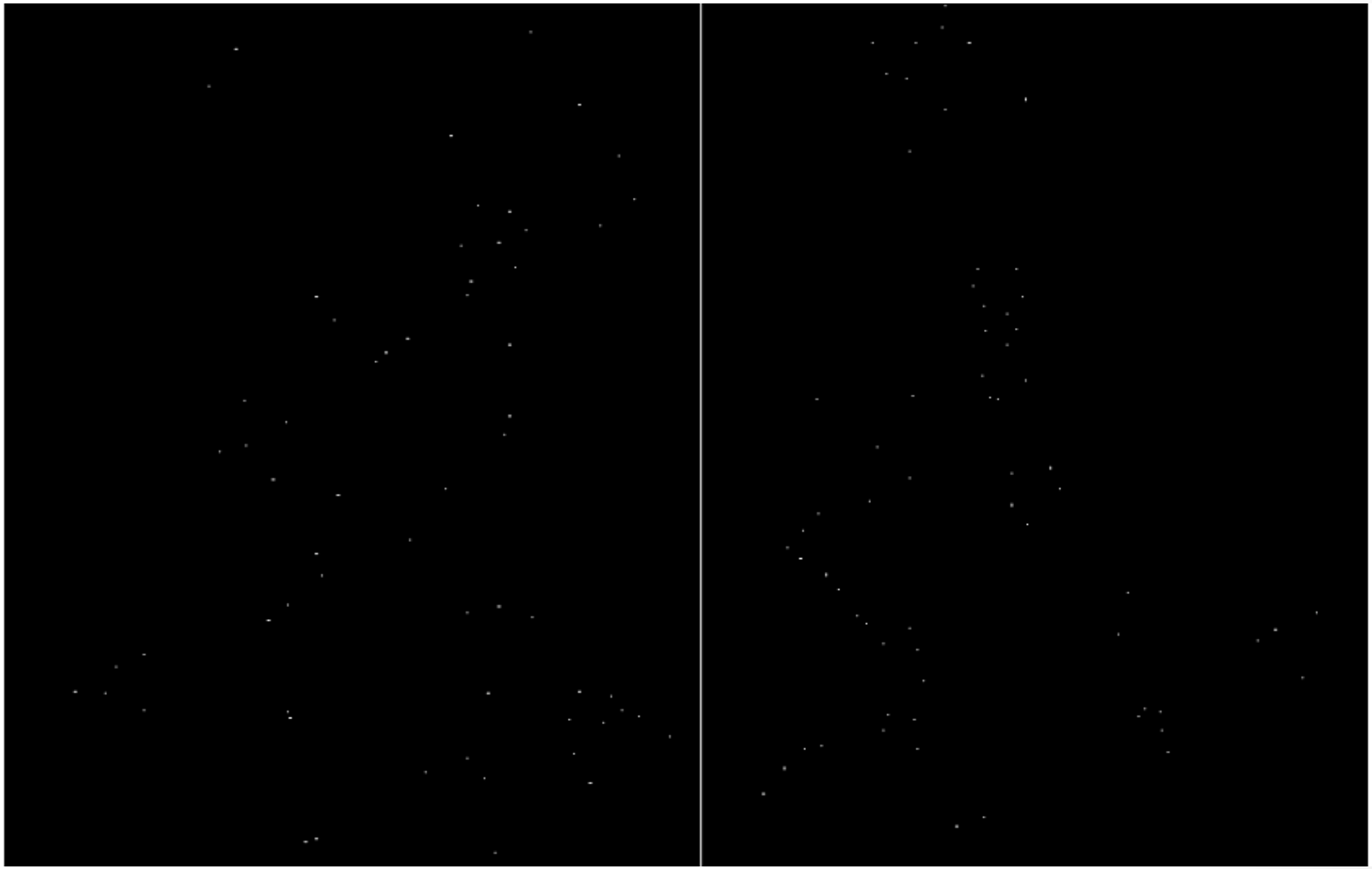
3. Gaussian filter $g(\sigma_I)$

Um exemplo

Respostas de R (maior azul escuro)



Recorte dos picos (threshold)



Cantos detectados

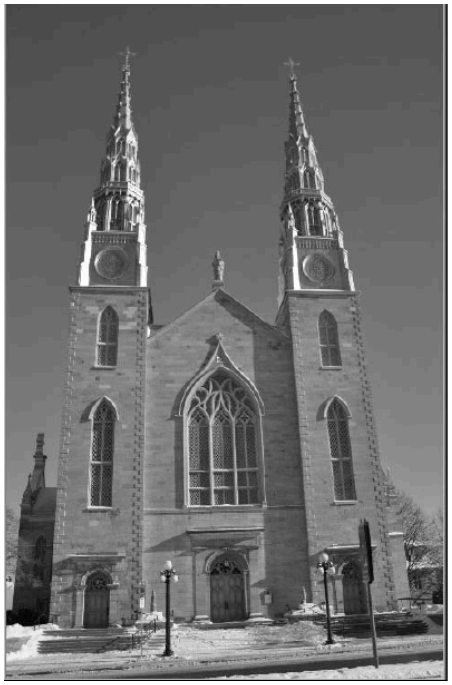
No opencv

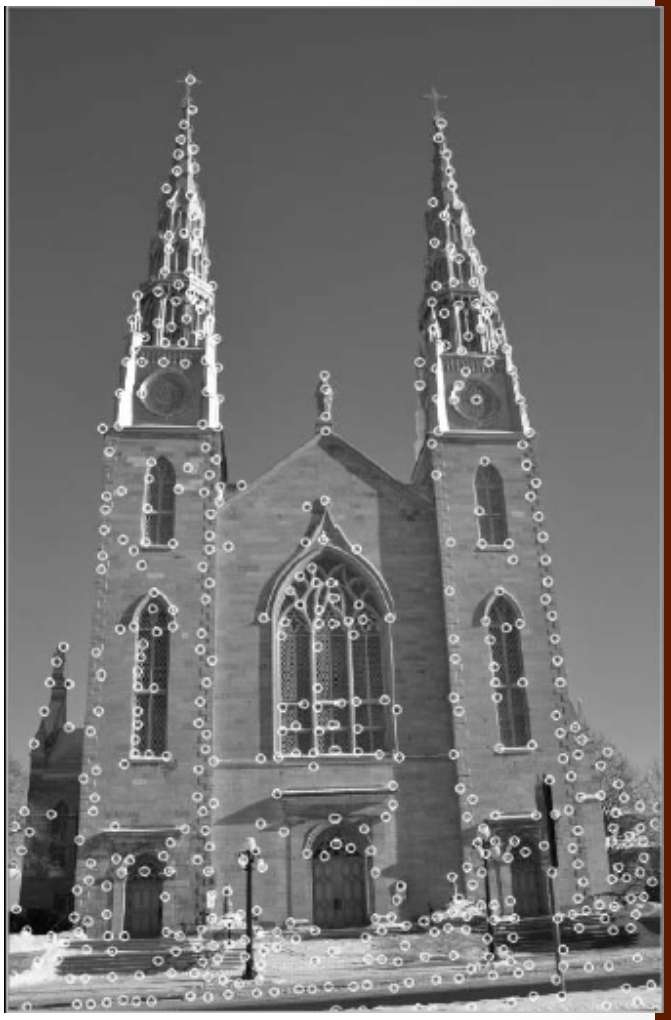
- Use a função cornerHarris(). Entrada:
  - imagem
  - tamanho da janela
  - tamanho da janela do Sobel para o gradiente
  - k: parâmetro da equação de harris

No opencv

```
cornerHarris( src, dst, 3, 3, 0.04);
dilate(dst,dst,cv::Mat());
double minStrength, maxStrength;
minMaxLoc(dst,&minStrength,&maxStrength);
threshold(dst, dst, 0.01*maxStrength, 255, BINARY_TRESH_INV)
```

Good Features to Track





```
vector<KeyPoint> keypointsD;
Ptr<GFTTDetector> detector =
GFTTDetector::create(500,0.01,10);
detector->detect(src,keypointsD,Mat());
drawKeypoints(src, keypointsD, src);
```

Propriedades Harris Detector

- Invariante:
  - translação
  - rotação
- Não atende a escala

- Custo elevado

Outro detector: FAST

- Problemas com Harry
  - Custo computacional

- Uma alternativa:
  - FAST: Features from Accelerated Segment Test

  - Detecção rápida de pontos
  - Decisão de aceitar ou não um ponto baseado em algumas comparações

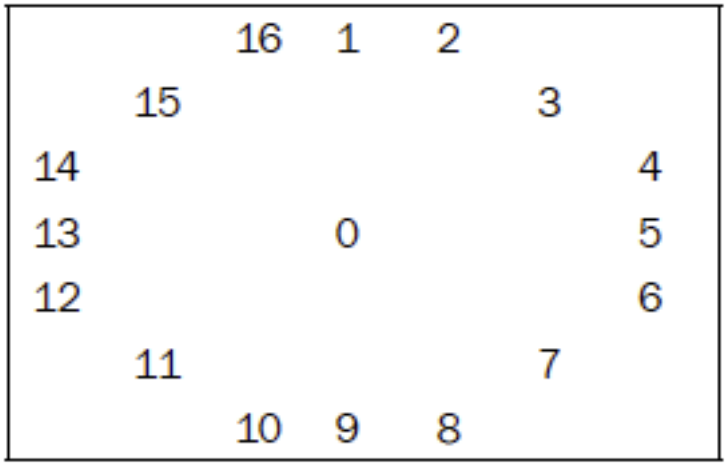
FAST

- Idéia: o que define um canto?
- A resposta é dada em relação aos vizinhos de um ponto analisado
  - Caso, num círculo de pontos onde o centro é o candidato, houver:
  1. um ARCO contínuo de pontos
  2. de tamanho maior que ¾ do círculo
  3. de pontos que diferem significativamente do candidato

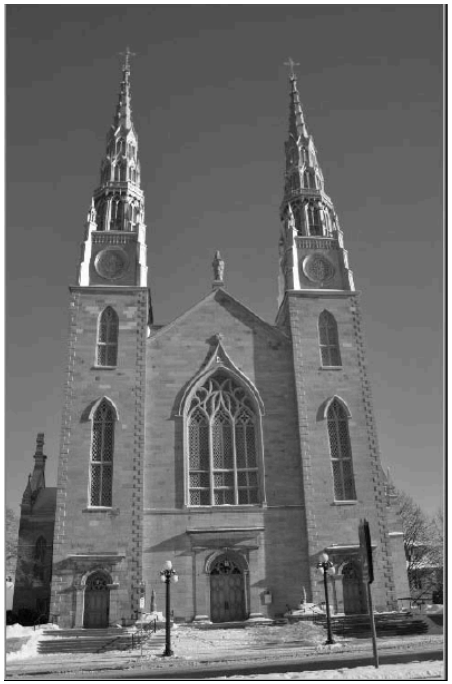
- Se obedecer 1-3, este é um keypoint

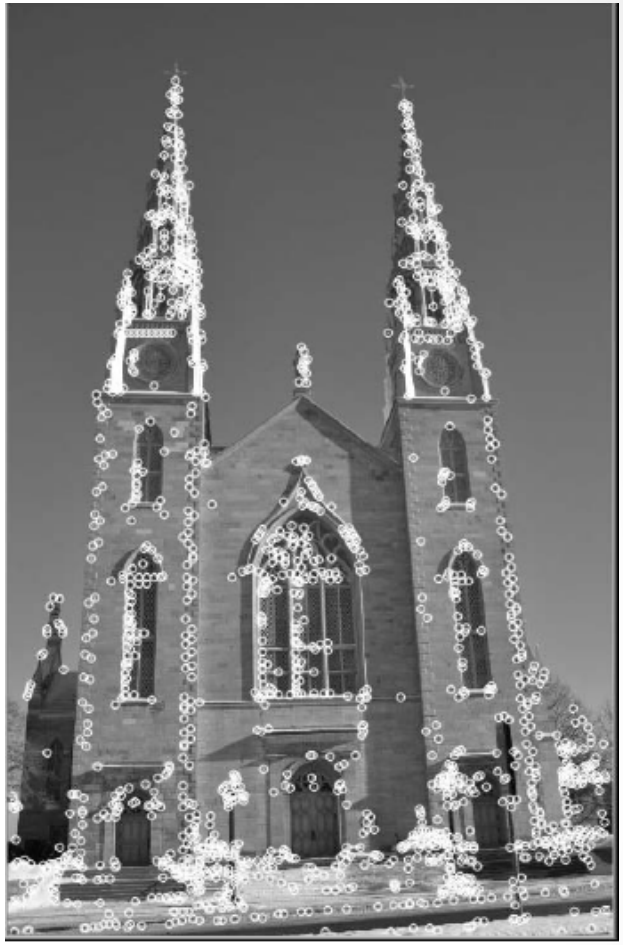
FAST

- Eliminação rápida de pontos:
  - teste rapidamente os **4 pontos nas direções cardeais (N – S –L – O)**
  - Se pelo 3 deles não diferem do candidato, o candidato é descartado
- Raio do círculo = 3 (testado como eficiente)



FAST no opencv





```
vector<KeyPoint> keypointsD;
Ptr<FastFeatureDetector> detector =
FastFeatureDetector::create();
detector->detect(src,keypointsD,Mat());
drawKeypoints(src, keypointsD, src);
```

Leitura extra

- Manual opencv: http://docs.opencv.org/3.0-beta/modules/features2d/doc/features2d.html