# Mecanismos de Detecção de Objetos – Selective Search

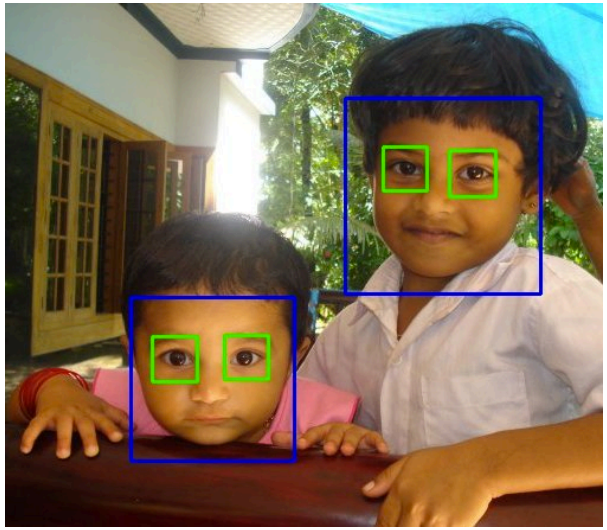
Visão Computacional

Prof. Geraldo Braz Junior

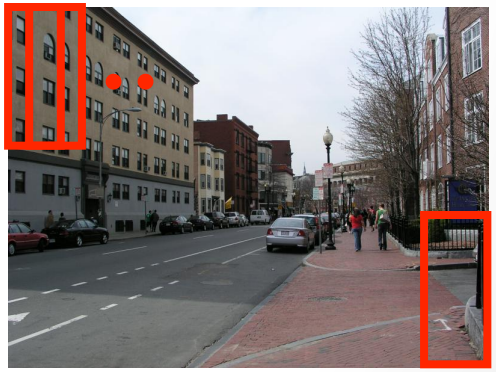
Contém material das notas de aula do CS131, CS229 CS231B

# Objetivo

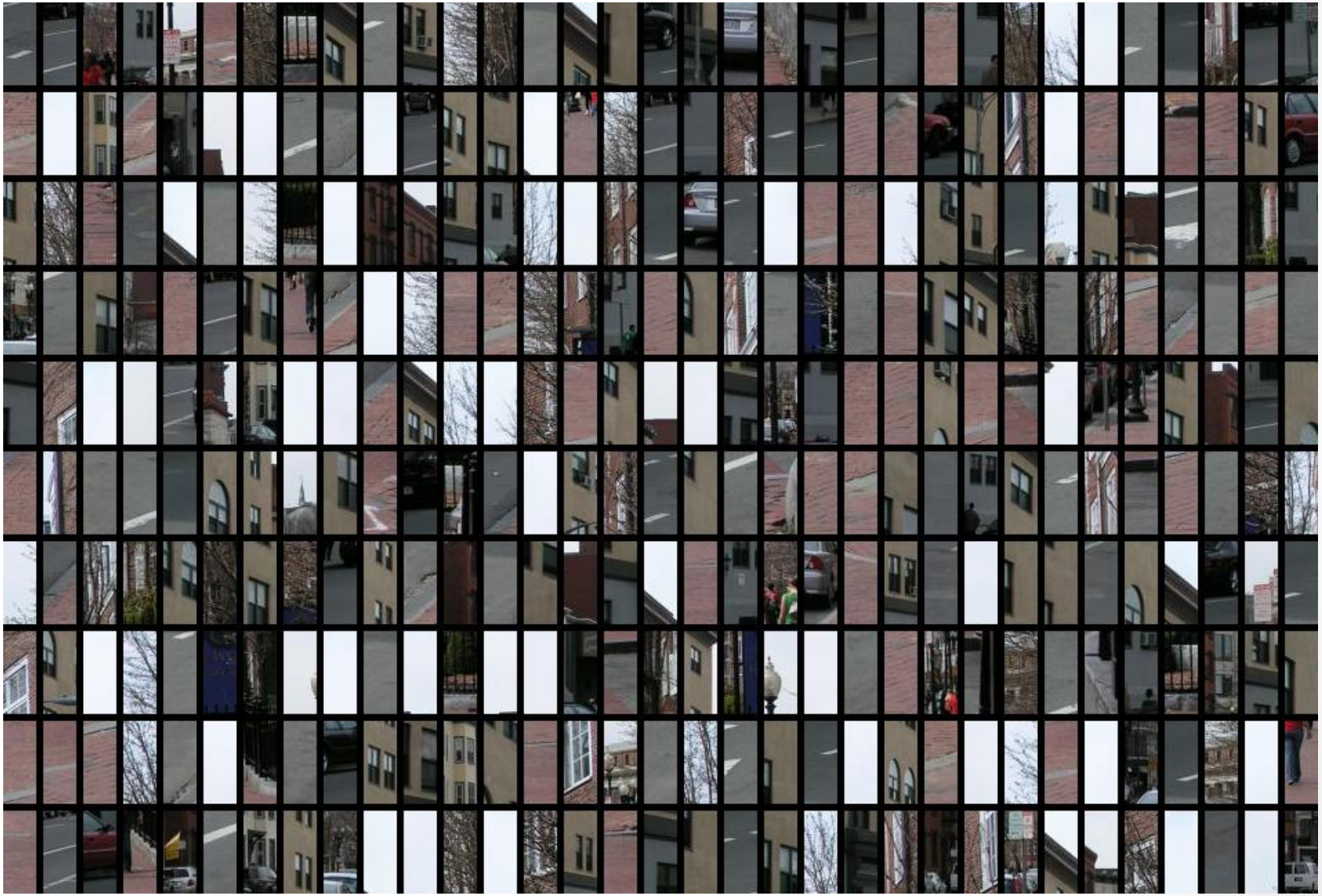
- Como detectar instancias de objeto?



# Uma abordagem: janela deslizante



# Quais são as janelas geradas?

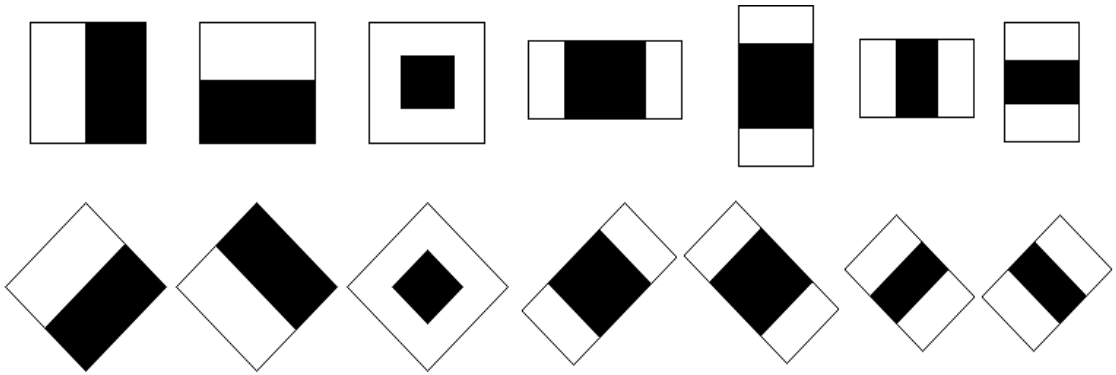


# Como reconhecer as janelas?

- Template Matching
- Shape
- **Cascade of Classifiers (+ Boosting)**
  - Proposto por Viola and Jones: "Rapid Object Detection using a Boosted Cascade of Simple Features" 2001

# Haar Features

- Diferença da soma de “brancos” com “pretos”
- O filtro deve ser posicionado na imagem, numa escala específica
- E depois obtido a métrica



# **Boosting** para seleção

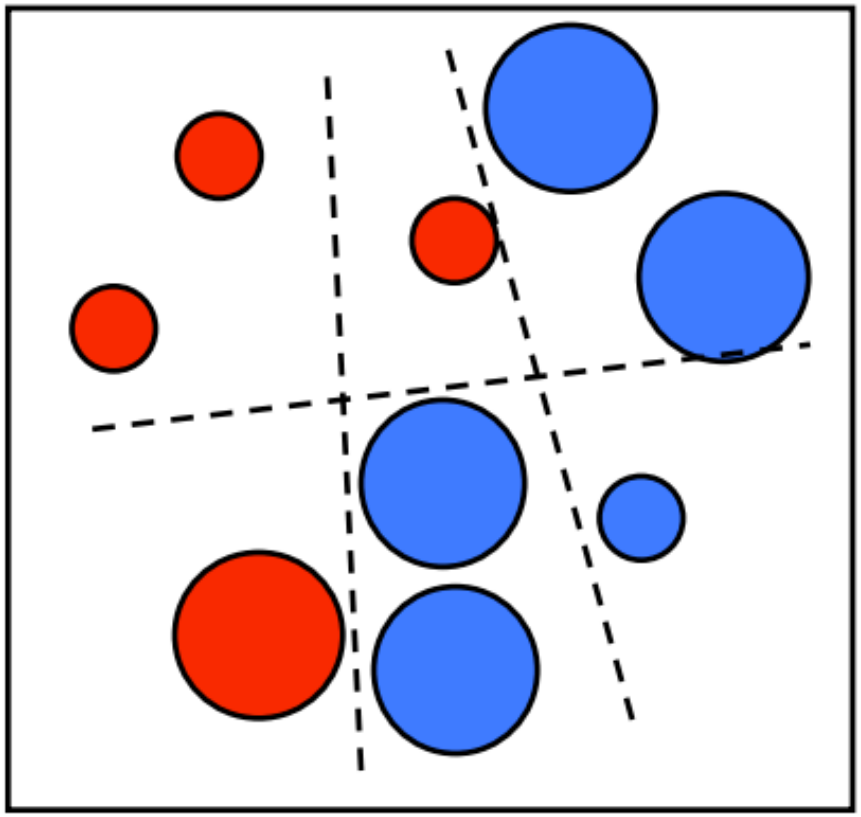
- Esquema de classificação (**ensemble**) que combina vários classificadores fracos para gerar um classificador robusto

- Fraco = usar apenas 1 feature

- Boosting rounds
  - selecionar um novo classificador fraco, que tem resultado melhor do que seus antecessores

**Final classifier is a weighted combination of the weak classifiers**

$$(\mathbf{x}) = \text{sign}\left(\sum_{j=1}^{M} \alpha_j h_j(\mathbf{x})\right)$$



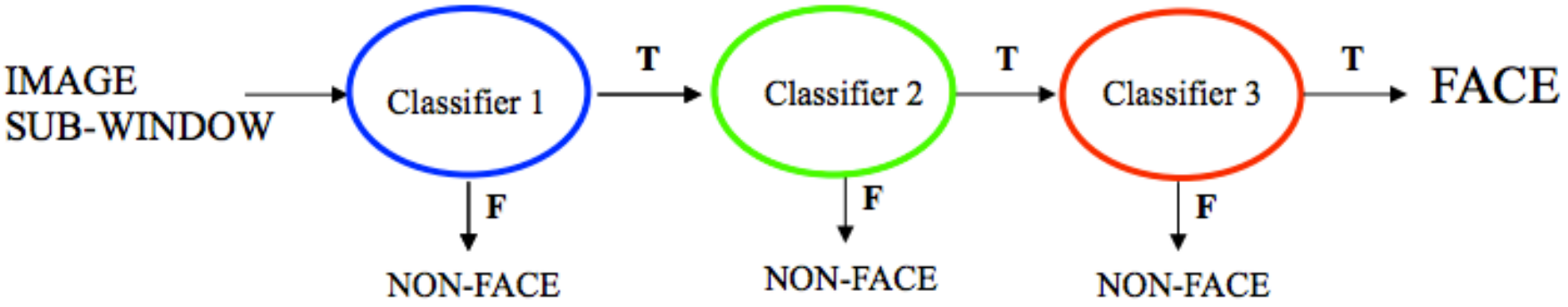
# A maldição dos falso positivos

- Um classificador com as 2 melhores features pode ter 100% de detecção, mas com 50% de falso positivos (0.5 a cada imagem)

- Um classificador com as 200 melhores features pode obter 95% de detecção com 1 falso positivo a cada 14084

# Cascade of Boosted

- Sequência de classificadores boosted com aumento constante de complexidade

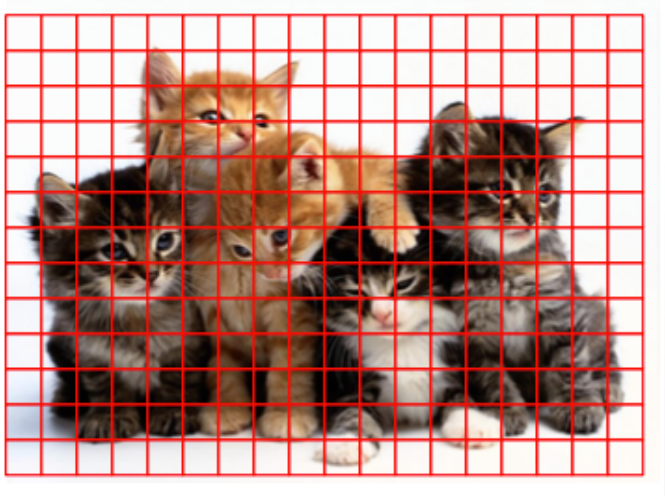



- Negativos são eliminados imediatamente!

# Alguma abordagem com o custo menor?

- Problema das abordagens anteriores está no tempo de treinamento
  - Alto

- Como fazer com que o tempo seja menor?

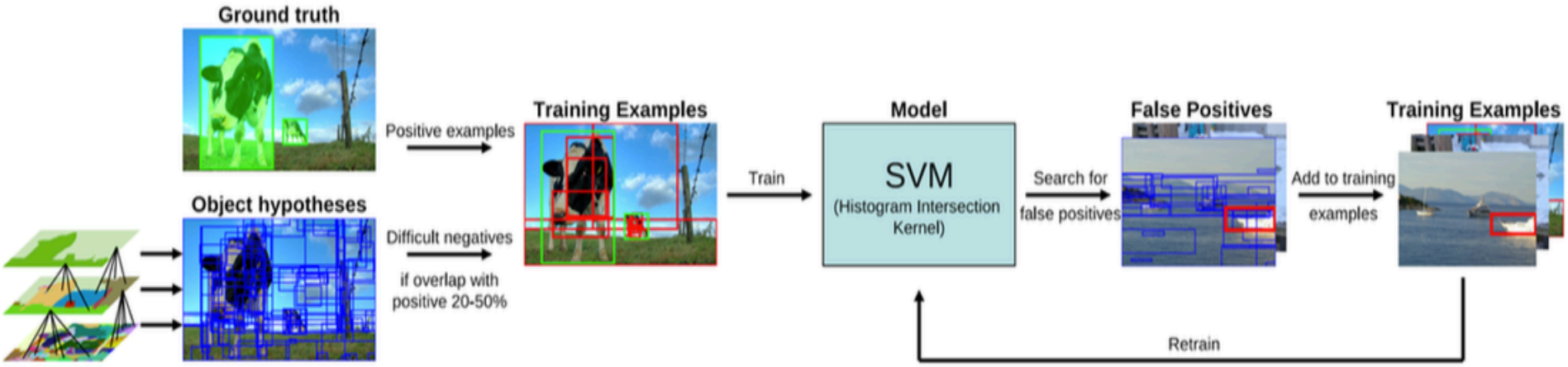


# Ideia

- Se segmentarmos corretamente a imagem e depois rodar o reconhecedor do objeto
- Custo menor!

- O problema é como segmentar mantendo todas as propriedades

# E o reconhecimento ficaria




**Selective Search for Object Recognition**

J.R.R. Uijlings, K.E.A. van de Sande, T. Gevers, and A.W.M. Smeulders

# O reconhecimento

Usando uma base de treinamento e validação

1. Obtenha as propostas
2. Treine um SVM com as melhores
3. Encontre falso positivos na resposta
4. Insira falso positivos no treinamento
5. Repita

# E para detectar com maior precisão:

- **Use Selective Search**
  - Codifica as regiões da imagens em sucessivas regiões melhoradas
  - Como uma hierarquia (abordagem bottom-up)

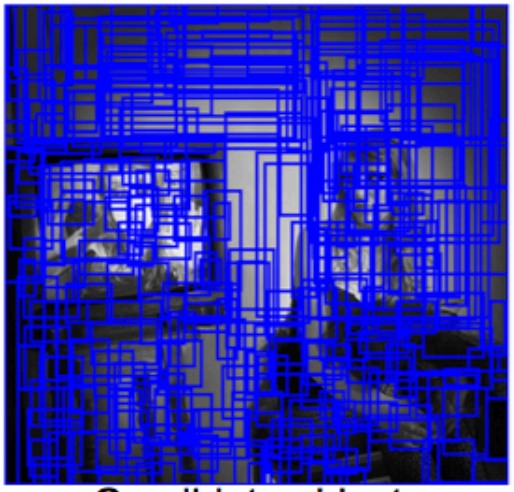
- Objetivos
  - velocidade
  - independencia de escala
  - várias informações de agrupamento

# Selective Search

1. Gere os candidatos iniciais
   - Use o algoritmo de P. F. Felzenszwalb and D. P. Huttenlocher. “Efficient Graph-Based Image Segmentation.” IJCV, 59:167–181, 2004.]

Input Image

Segmentation

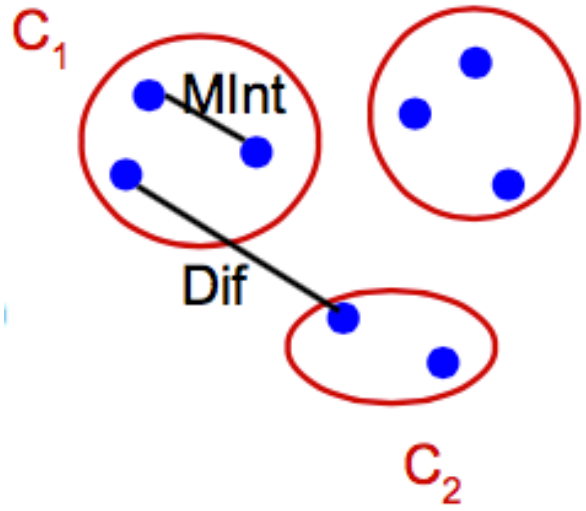


Candidate objects

# Felzenszwalb and Huttenlocher: Efficient Graph-Based Image Segmentation

- Determina onde fica a fronteira

$$D(C_1, C_2) = \begin{cases} \text{true} & \text{if } Dif(C_1, C_2) > MInt(C_1, C_2) \\ \text{false} & \text{otherwise} \end{cases}$$



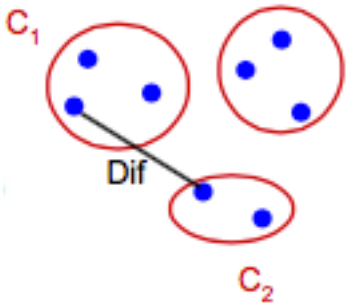

**C1** e **C2** são componentes. Regiões

**Mint** é a diferença interna da componente

# Predicado de segmentação

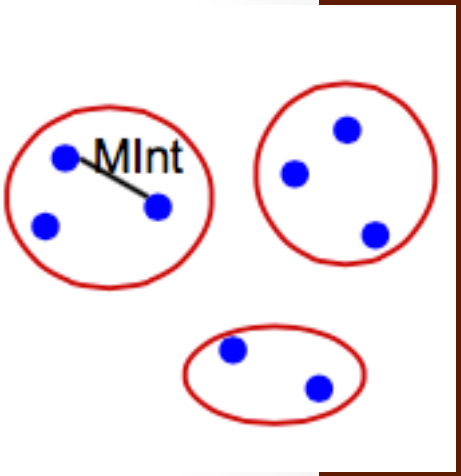
- A **diferença** entre duas componentes (**Diff**) e dada pelo **menor valor de aresta** que conecta o vertice i na componente C1 com o vértice j da componente C2

$$Dif(C_1, C_2) = \min_{v_i \in C_1, v_j \in C_2, (v_i, v_j) \in E} w(v_i, v_j).$$



# Predicado de segmentação

- A **diferença interna** de uma componente (**Int**) é dada pelo **maior valor de aresta** numa componentes, obtida após uma Árvove Geradora Mínima (**MST**)




$$Int(C) = \max_{e \in MST(C,E)} w(e).$$

$$MInt(C_1, C_2) = \min(Int(C_1) + \tau(C_1),\ Int(C_2) + \tau(C_2)).$$

$$\tau(C) = k/|C|$$

# Predicado de segmentação

- T(C) ajusta o grau de corte entre componentes
  - O limite do quão nós internos a uma componente podem ser diferentes

$$\tau(C) = k/|C|$$

- **k é uma constante que representa tamanhos de componentes**
- |C| representa o tamanho da componente

# Dado o predicado, passo a passo ...

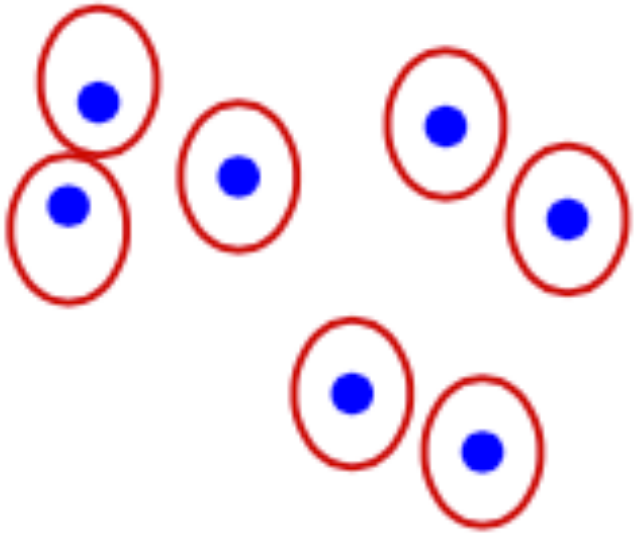
- Primeiro, inicialize:
  - construa o grafo, e ordene as arestas em ordem crescente de peso
  - Peso calculado pela diferença entre de intensidade

$$w(v_i, v_j) = |I(p_i) - I(p_j)|$$

    - para imagens coloridas, uma para cada canal

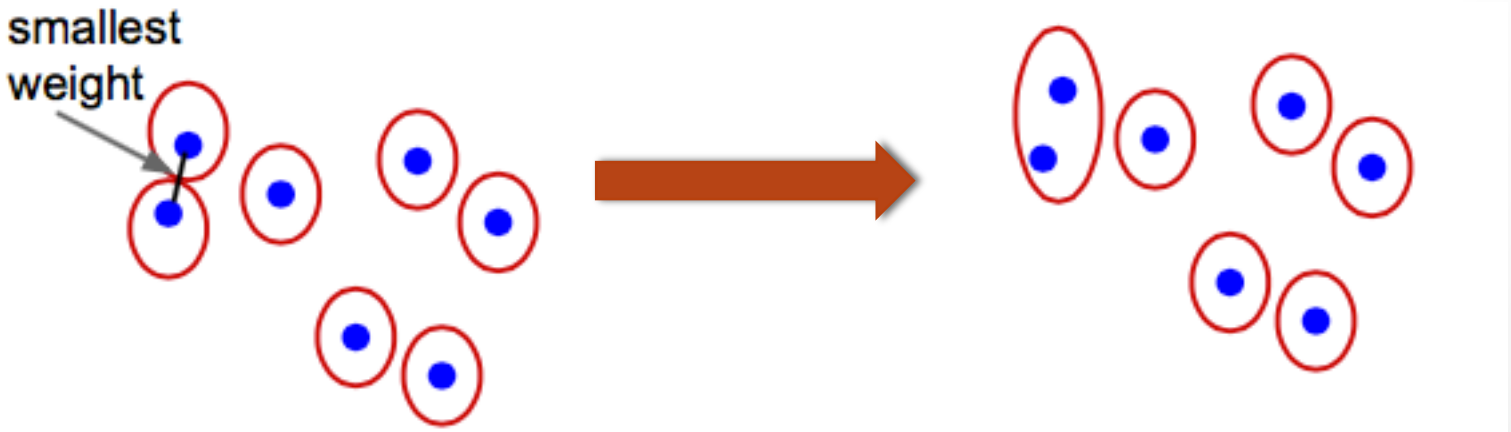
# Parte 1: inicilizando

- **Segmentação inicial:** Cada vértice tem sua própria componente



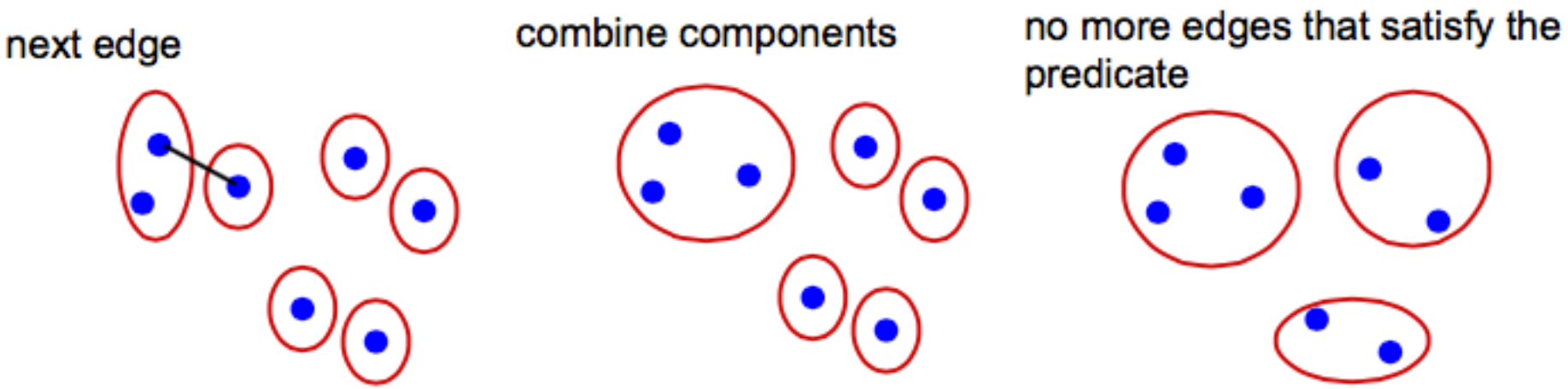
# Parte 2:

- Construa $S^q$ baseado no estado anterior $S^{q-1}$.
- Dado dois vértices, **vi e vj**, unidos pela **aresta de menor peso**, cheque o **predicado**:
  - atende? **combine** as componentes
  - Não atende? não faça nada



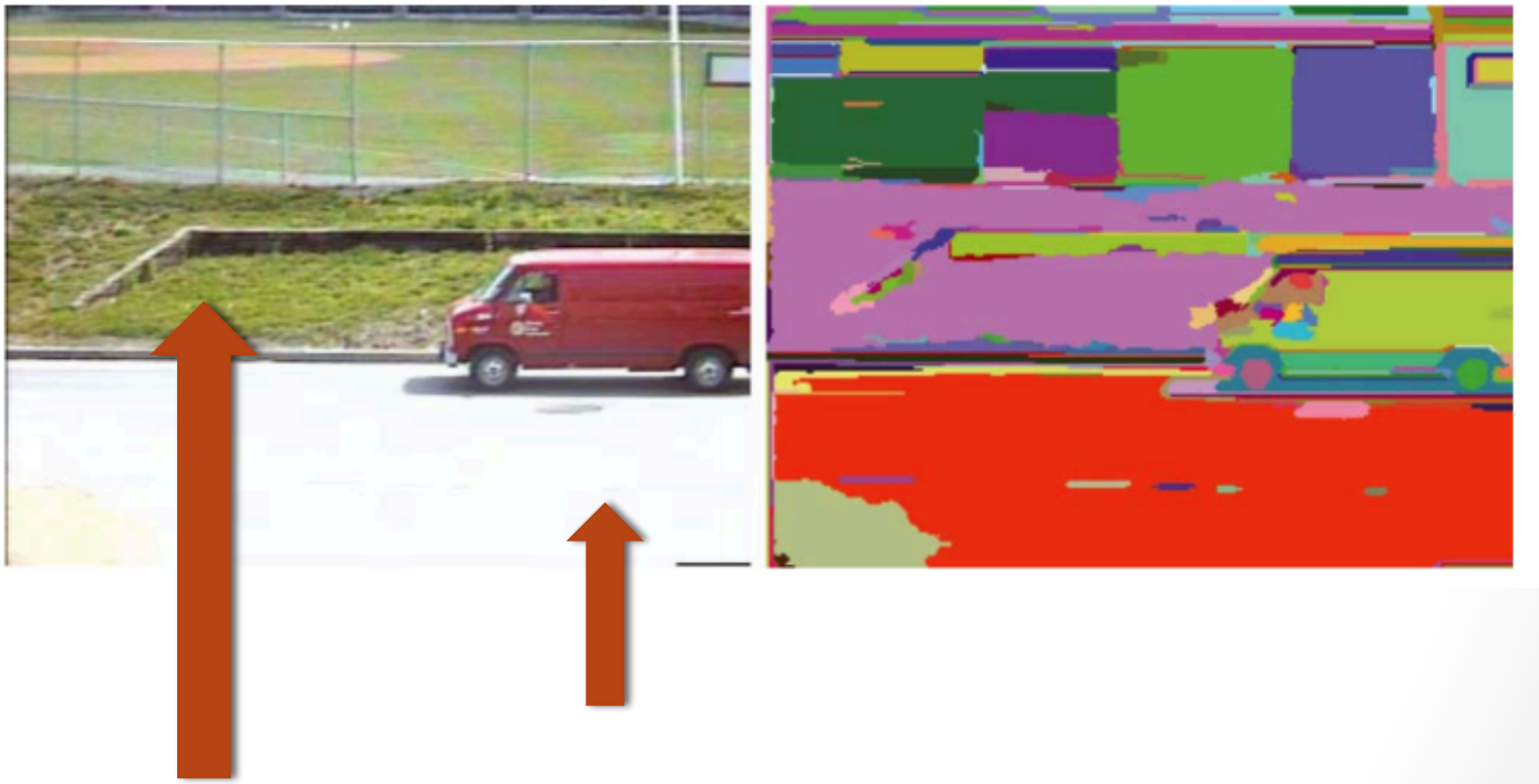
# Repita

- O passo 2 para todos os vértices

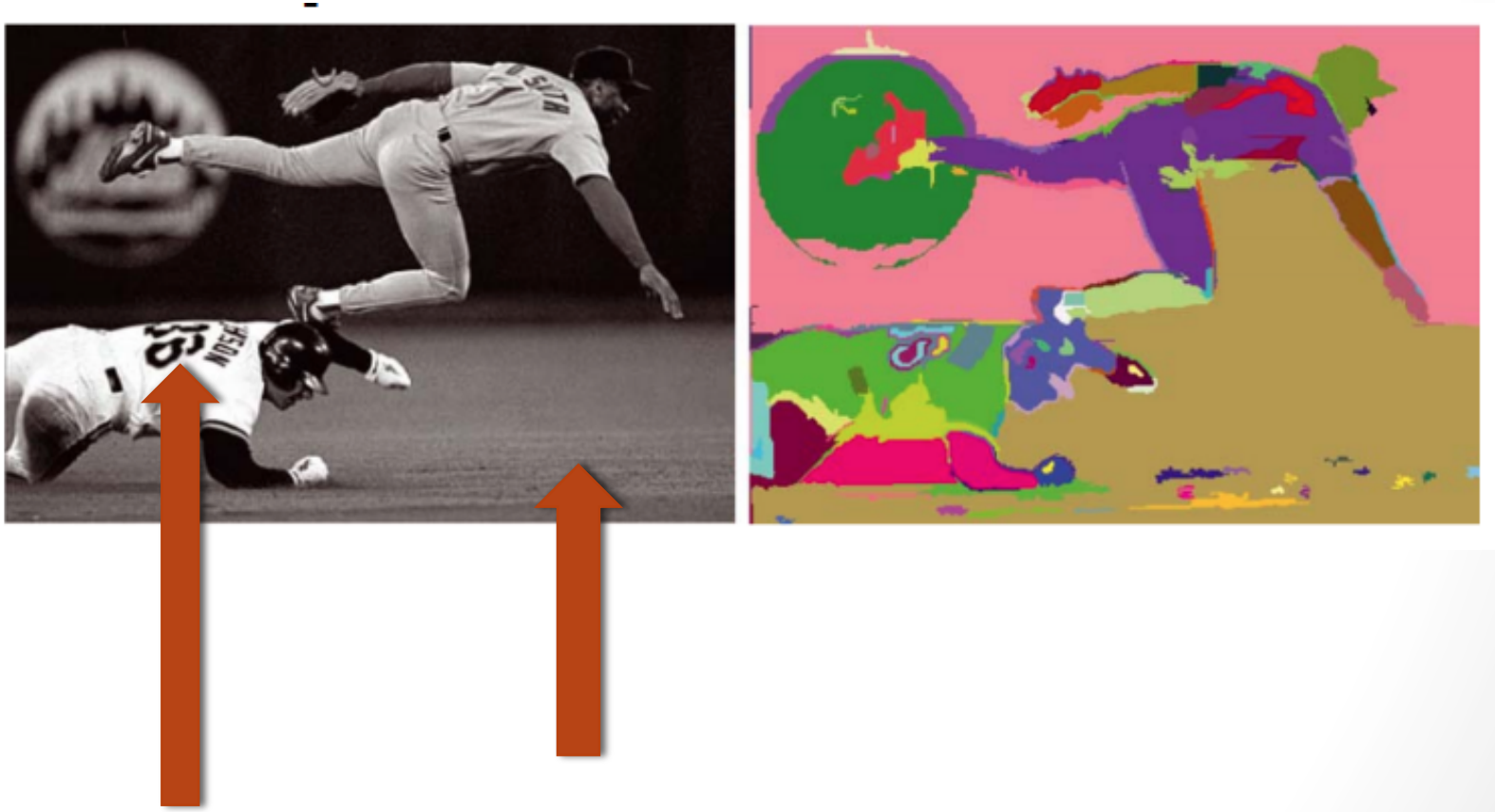



- No final, retorne o novo grafo com as componentes conectadas

# Alguns resultados



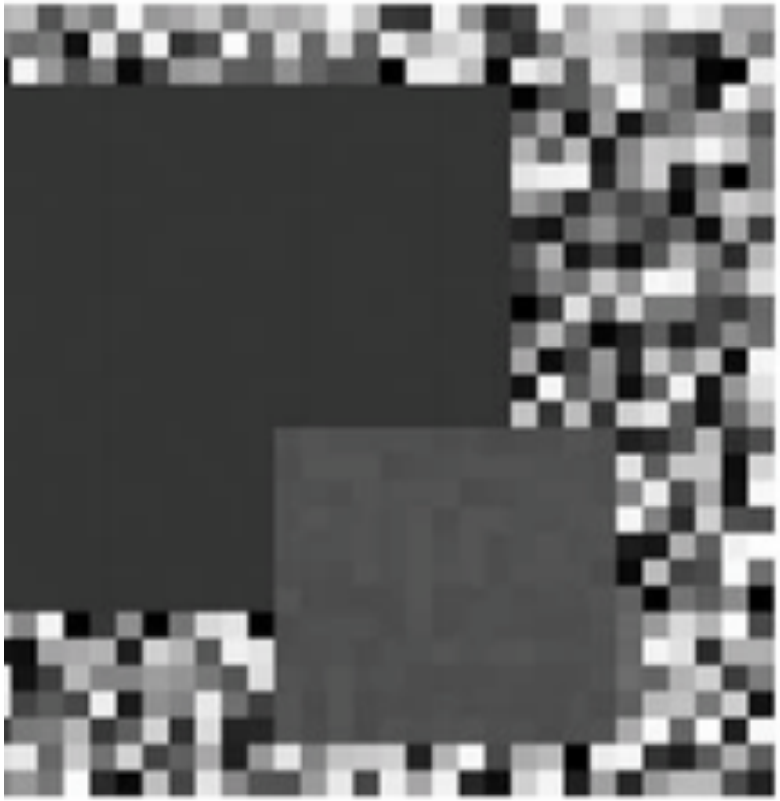
# Alguns resultados



# Um pequeno ajuste no peso

- Troque a diferença de intensidades por distância euclidiana (**L2 norm**) entre os pixels
  - No caso da colorida faz mais sentido
  - (x, y, r, g, b) -> features
  - **Neighbor Graph Weights**
    - limitado aos 10 vizinho mais próximos

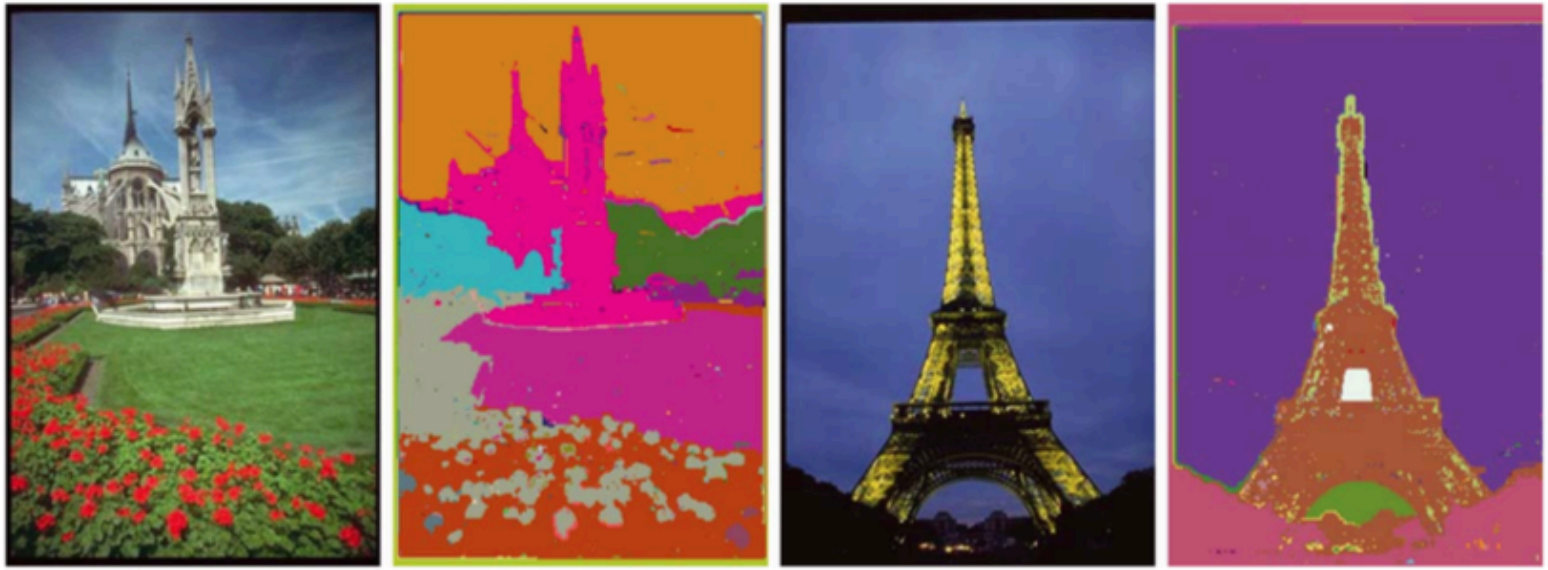
# Novos resultados



Image

Segmentation

# Novos Resultados

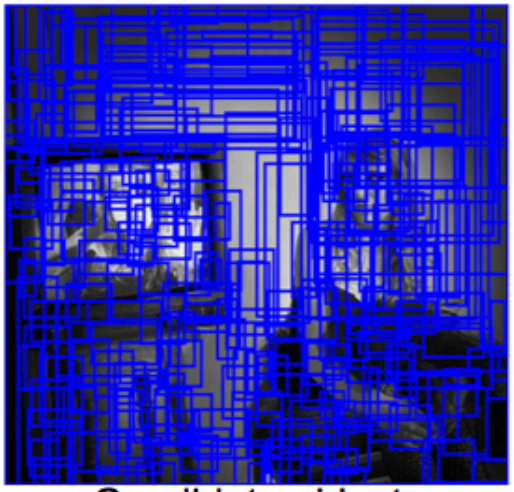


# Selective Search

1. Gere os candidatos iniciais
   - Use o algoritmo de P. F. Felzenszwalb and D. P. Huttenlocher. “Efficient Graph-Based Image Segmentation.” IJCV, 59:167–181, 2004.]

Input Image

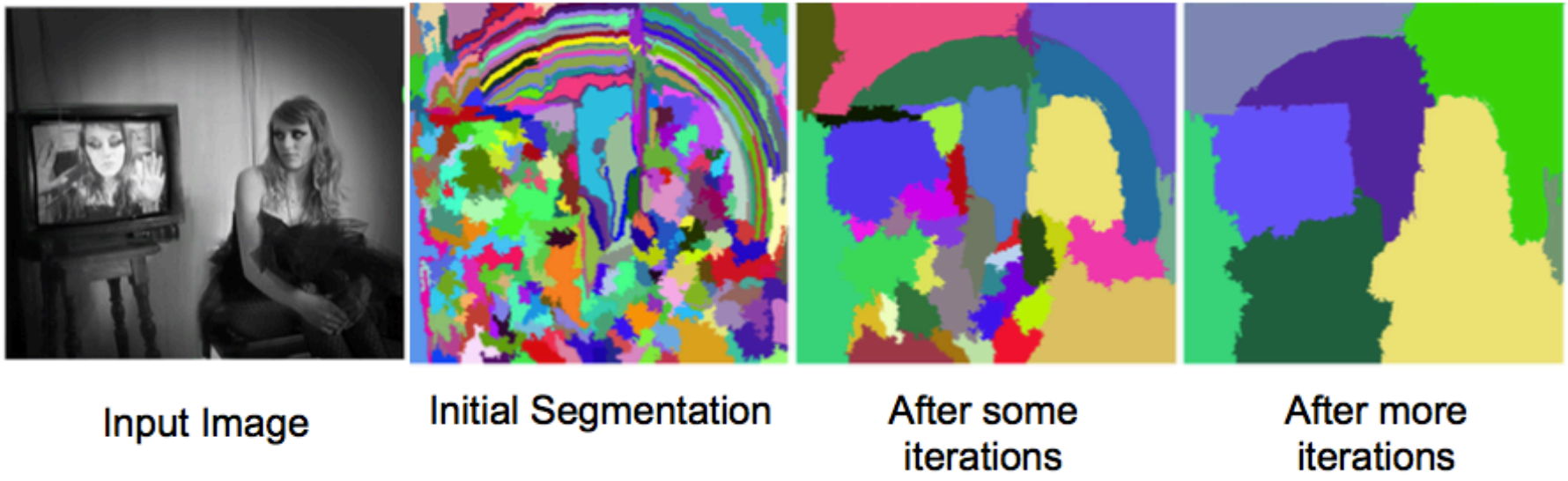
Segmentation



Candidate objects

# Selective Seach

2. Combine partes similares recursivamente
   - Selecione as duas regiões mais similares
   - combine-as em uma
   - repita até que exista apenas uma região



# Selective Search

3. Gere os candidatos!

Input Image

# E como se considera duas regiões parecidas?

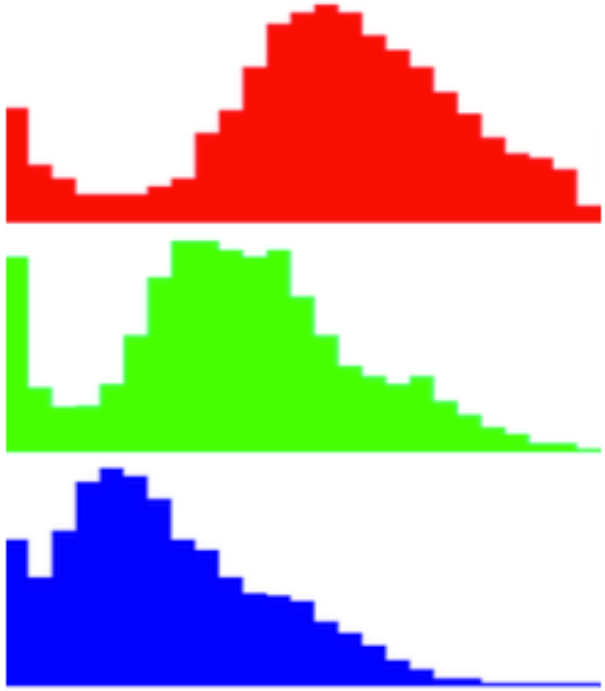
- No artigo são definidas algumas méticas
  - Mas pode-se customizar de acordo com a necessidade

- Similaridade
  - Cor
  - Textura
  - Tamanho
  - Forma

# Similaridade de Cor

- Medida de similaridade de cor calculada pela através da interseção dos histogramas de cada canal

$$s_{colour}(r_i, r_j) = \sum_{k=1}^{n} \min(c_i^k, c_j^k)$$



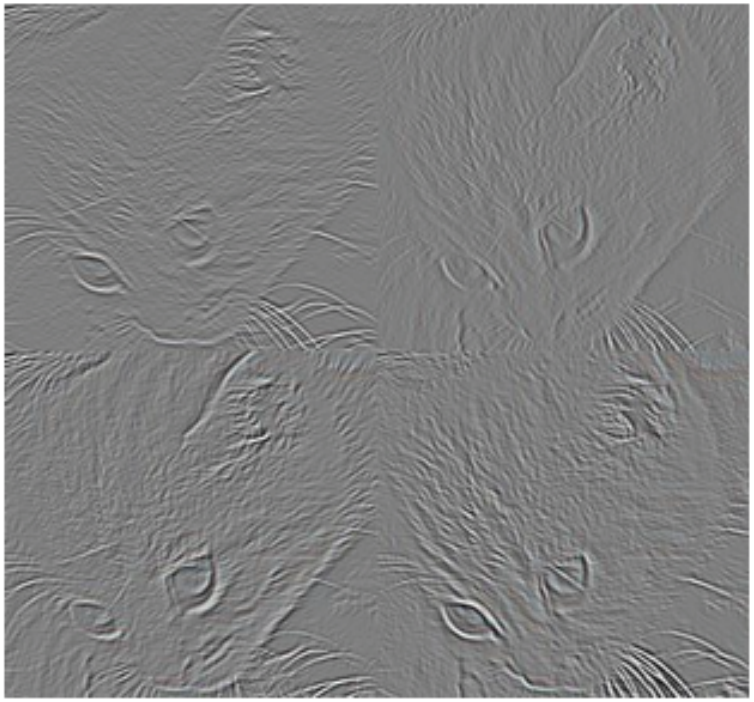
- No artigo foram usados

25 bins, portanto 75 features

- Escolha o melhor esquema de cor

# Similaridade de Textura

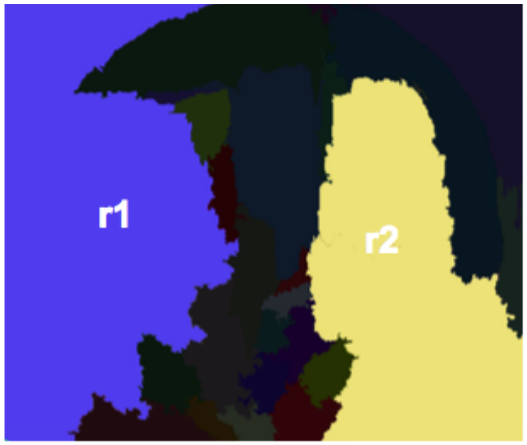
- Pode ser calculada usando HOG (original do artigo)
  - HOG de oito direções para cada canal  (24 HOG)
  - Cada HOG com 10 bins (240 features)

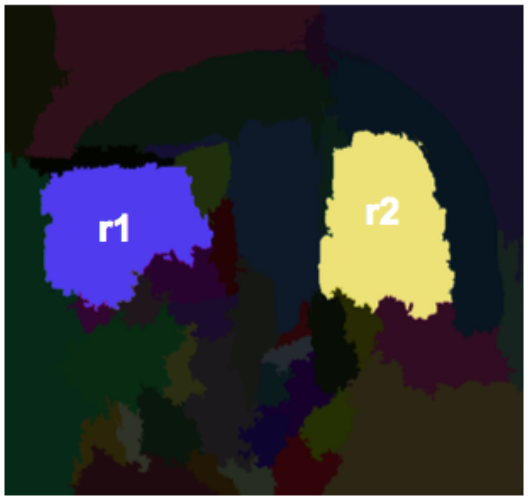


# Similaridade de Tamanho

- Consiste em agrupar regiões menores com regiões maiores
- Assim, as regiões vão crescendo e se tornando balanceadas

$$s_{size}(r_i, r_j) = 1 - \frac{\text{size}(r_i) + \text{size}(r_j)}{\text{size}(im)}$$

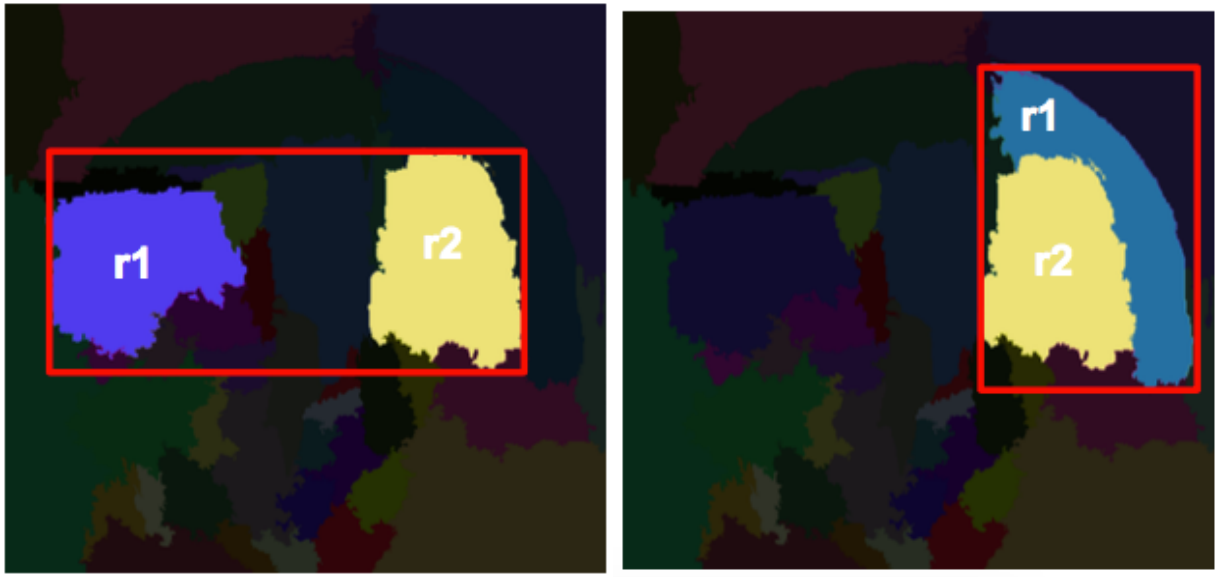


# Similaridade de Forma

- Medida que avaliar o quão bom duas regiões estariam se estivessem juntas
  - Basicamente mede se existem mudanças drásticas de

$$fill(r_i, r_j) = 1 - \frac{\text{size}(BB_{ij}) - \text{size}(r_i) - \text{size}(r_i)}{\text{size}(im)}$$



# Similaridade Final

- A contribuição de cada similaridade é adicionada numa equação linear

$$s(r_i, r_j) = a_1 s_{colour}(r_i, r_j) + a_2 s_{texture}(r_i, r_j) + a_3 s_{size}(r_i, r_j) + a_4 s_{fill}(r_i, r_j),$$

- Comum:
  - Usar esquemas de ponderação
  - Usar LDA para ajustar os pesos automaticamente

# Alguns Resultados

(a) Bike: 0.863

(b) Cow: 0.874

(c) Chair: 0.884


(d) Person: 0.882

# Conclusões

- Selective Search apresenta um modelo rápido para detecção de objetos
  - Também é eficiente
  - e customizável de acordo com o problema

- O janelamento ainda é adequado quando não se consegue uma segmentação minimamente adequada!