Casamento de formas com *Shape Context*

Aluno: Diego Mira David
Supervisor: Ronaldo Fumio Hashimoto

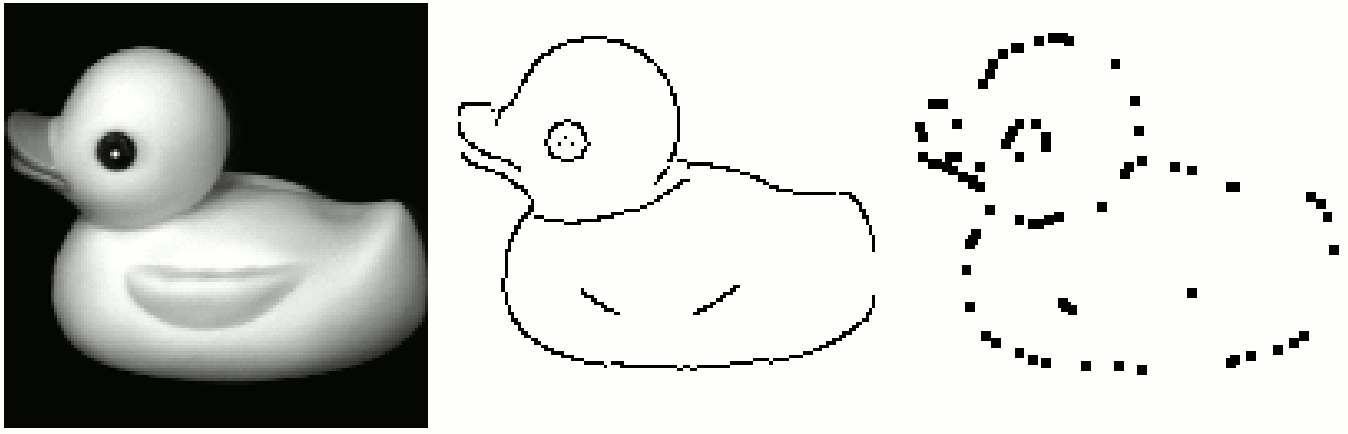
Trabalho baseado em artigos publicados por
Serge Belongie, Greg Mori, Jitendra Malik e Jan Puzicha

Objetivos do trabalho

- Estudo dos algoritmos para casamento de formas que utilizam o descritor shape context, proposto por **S. Belongie** e **J. Malik** inicialmente em "**Matching with Shape Contexts**".
- Desenvolvimento de um *software* para aplicar os conceitos estudados.

Shape Context

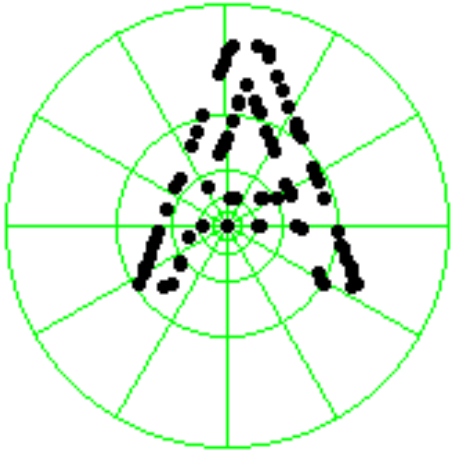
- Amostra de pontos do contorno de uma imagem:



- Para cada ponto p_i da amostra é calculado um **histograma** das coordenadas relativas dos demais pontos do contorno.

Shape Context

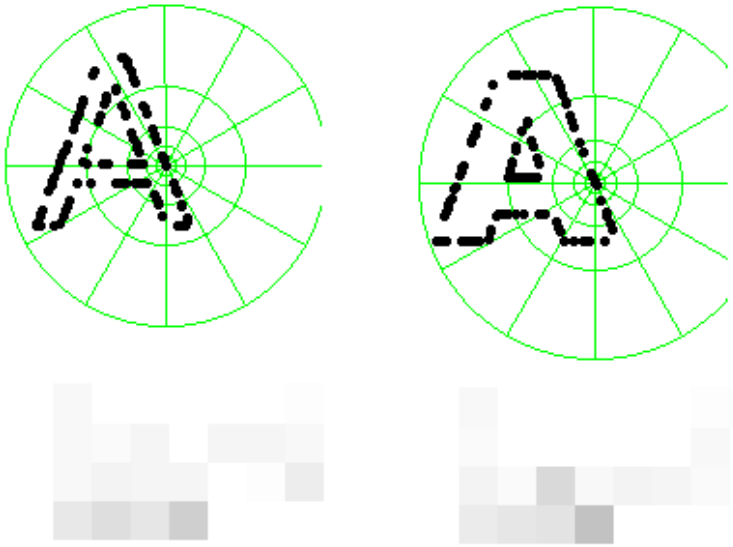
- Diagrama log-polar, particionado em 60 *bins*, torna o descritor mais sensível às posições mais próximas de p_i.



$$h_i(k) = \#\{q \neq p_i : (q - p_i) \in bin(k)\}$$

- O histograma, que é o *shape context* de p_i, incorpora informação global da forma ao armazenar a distribuição dos pontos nos bins.

Shape Context



- O custo para casar um ponto p_i com um ponto q_i, que é uma medida da diferença entre os dois *shape contexts*, é dado por:

$$C(p_i, q_j) = \frac{1}{2} \sum_{k=1}^{K} \frac{[h_i(k) - h_j(k)]^2}{h_i(k) + h_j(k)}$$

Generalized Shape Contexts

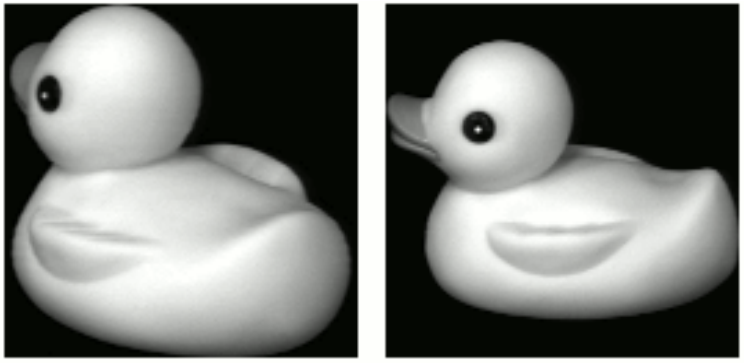
- Uma extensão ainda mais rica do descritor, proposta em "**Efficient Shape Matching Using Shape Context**", considera vetores tangentes t_i de comprimento unitário para indicar a direção do contorno em cada ponto p_i.

$$\hat{h}_i(k) = \sum_{q_i \in Q} t_i \quad onde \quad Q = \{q \neq p_i : (q - p_i) \in bin(k)\}$$

- Cada *bin* carrega um vetor com a direção dominante do contorno dentro do *bin*. A distância é dada por:

$$C(p_i, q_j) = \sum_{k=1}^{K} \frac{||\hat{h}_i(k) - \hat{h}_j(k)||^2}{||\hat{h}_i(k) + \hat{h}_j(k)||}$$

Correspondência entre os pontos

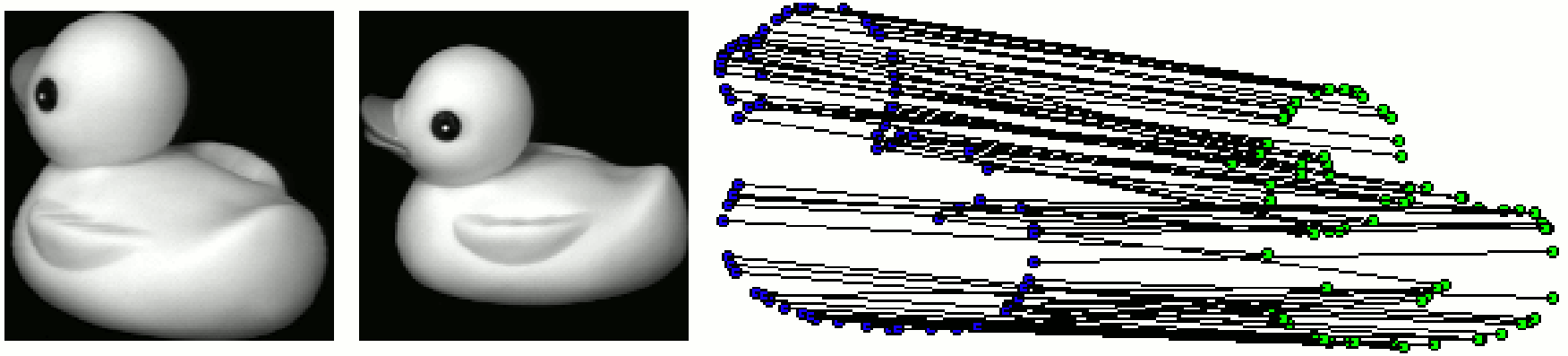


- Dada a matriz com os custos C_{ij} de todos os pares de pontos p_i da amostra de pontos de uma forma e q_i da outra, desejamos encontrar uma **permutação** π que minimize o custo total, dado por:

$$H(\pi) = \sum_i C(p_i, q_{\pi(i)})$$

Correspondência entre os pontos

- Foi implementado o algoritmo Kuhn-Munkres (também conhecido como Método Húngaro), um algoritmo de otimização que resolve o *assignment problem* em tempo $O(n^3)$



Representative Shape Contexts

- Objetivo: Obter rapidamente uma lista de candidatos que inclua a forma mais semelhante à consulta.
- Apenas alguns *shape contexts* são calculados para a imagem de consulta.

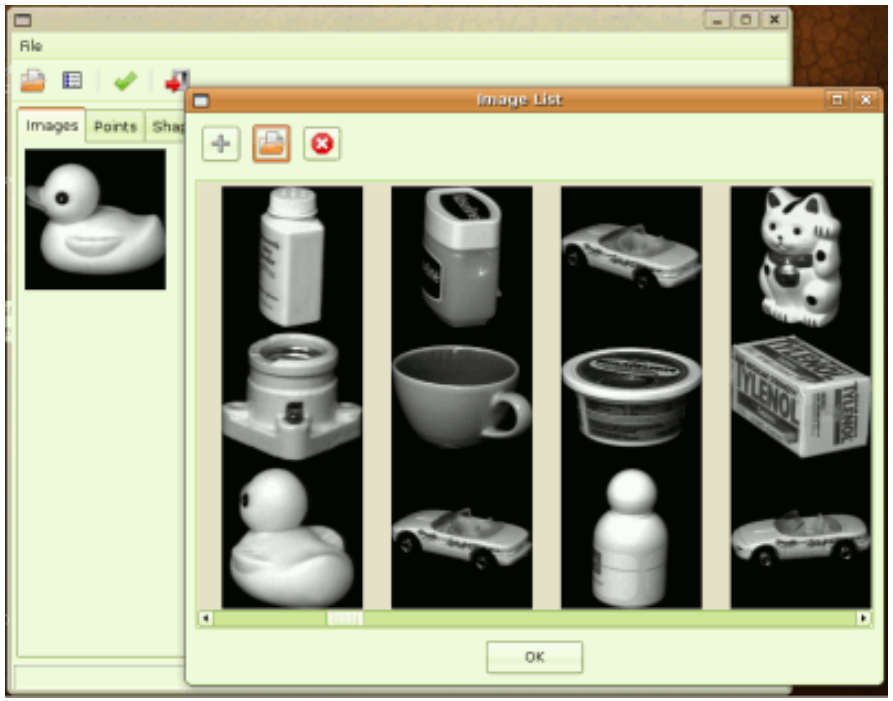
$$d_S(Q, S_i) = \frac{1}{k} \sum_{u \in G_i} \frac{d_{GSC}(SC_Q^u, SC_i^{m(u)})}{N_u}$$

$$m(u) = arg\ min_j d_{GSC}(SC_Q^u, SC_i^j)$$

$$N_u = \frac{1}{|S|} \sum_{s_i \in S} d_{GSC}(SC_Q^u, SC_i^{m(u)})$$

O software desenvolvido

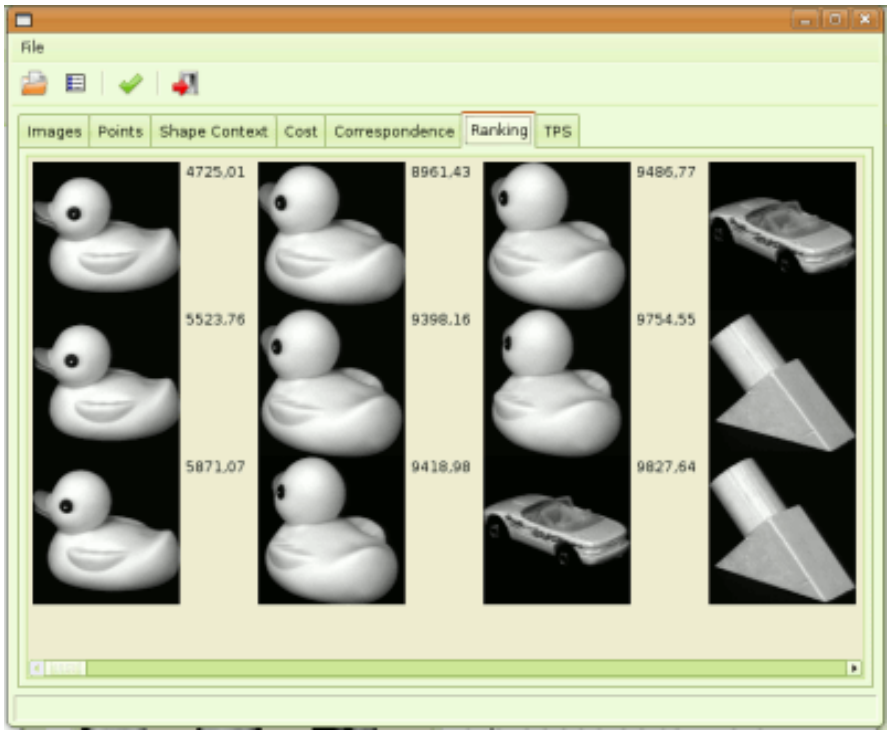
- Linguagem: C++
- Biblioteca gráfica: wxWidgets
- Coleção de bibliotecas: Boost



Funcionamento do programa com imagens do dataset *Columbia Object Image Library (COIL-20)*

O software desenvolvido

- Ranking das imagens do conjunto mais semelhantes à imagem de consulta.
- Fornece informações para acompanhar o funcionamento do método utilizado.

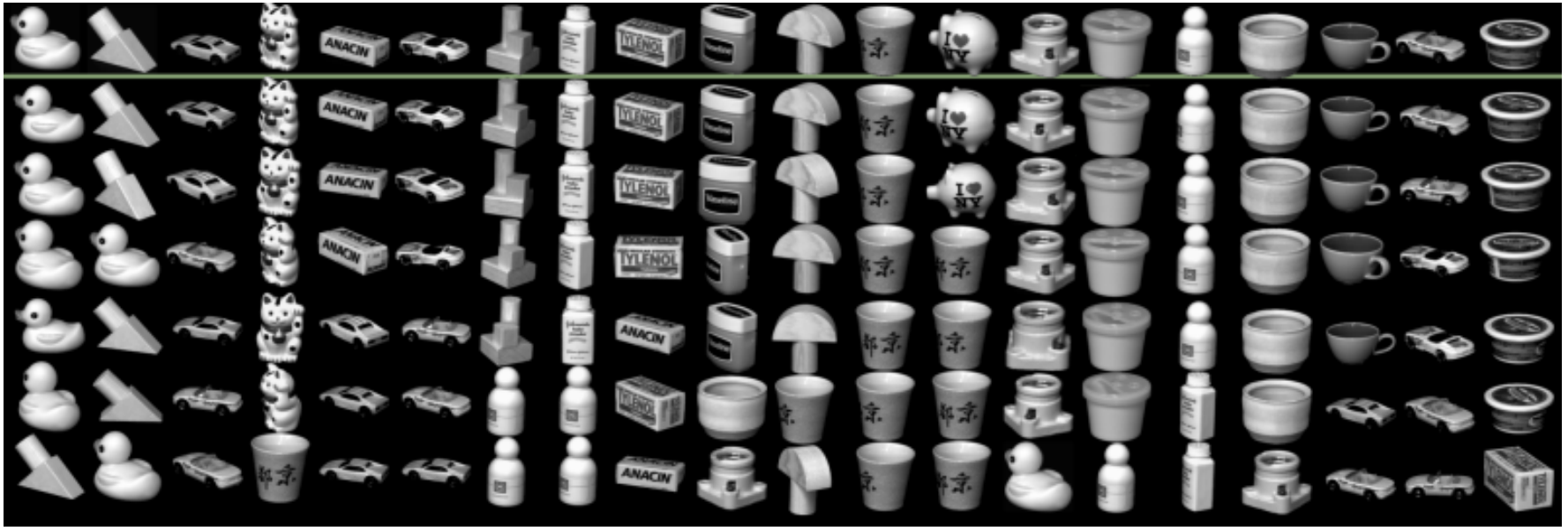


Funcionamento do programa com imagens do dataset *Columbia Object Image Library (COIL-20)*

Experimento

- Imagens do *dataset* COIL-20
- Foram utilizadas 100 imagens no experimento. São considerados 20 objetos, 5 imagens para cada (-15,-30,0,15,30)

Resultado:



Gestos do mouse

- É possível utilizar a mesma abordagem no reconhecimento de gestos do mouse (*mouse gestures*).
- Interação por meio de formas desenhadas com o movimento do mouse.
- Movimentos feitos descrevem formas simples e sem ruídos. Poucas formas precisam ser comparadas.

Captchas

- **CAPTCHA** é acrônimo de “Completely Automated Public Turing test to tell Computers and Humans Apart”.
- Visa distinguir humanos de computadores para prevenir contra acessos automatizados a um sistema.
- Problemas de usabilidade e acessibilidade.
- Projetos têm demonstrado que captchas visuais baseados em caracteres podem ser quebrados por computadores!

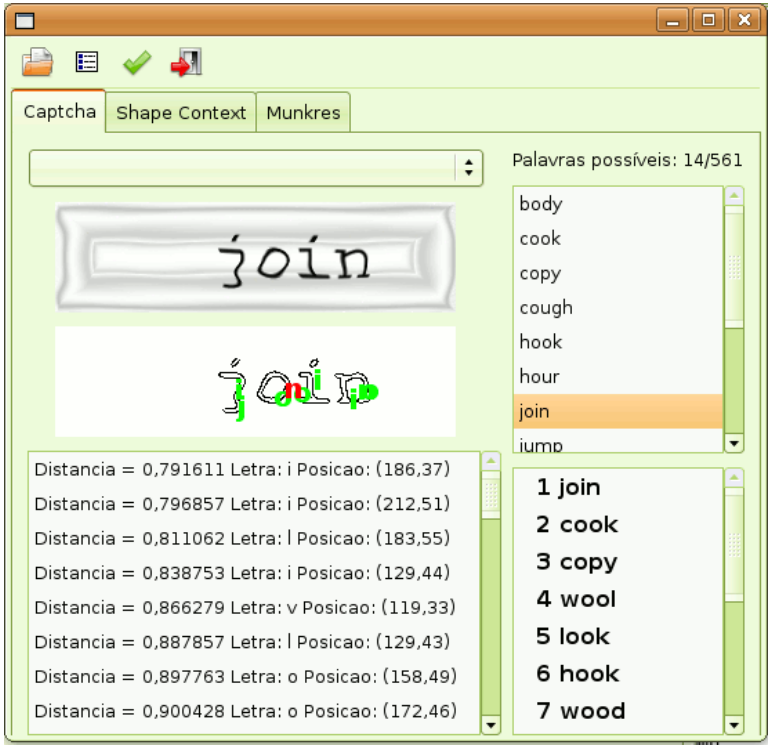
Captchas

Neste trabalho foi estudado um algoritmo para quebrar o captcha EZ-Gimpy descrito em “**Recognizing Objects in Adversarial Clutter: Breaking a Visual CAPTCHA**”.

Algoritmo

1. Localizar possíveis letras em diversas localizações.
2. Construir um grafo com todas as letras consecutivas que podem ser usadas para formar uma palavra.
3. Dentre todos os caminhos possíveis no grafo, são selecionadas apenas palavras reais com auxílio de um dicionário. Estas palavras são então avaliadas em um processo mais caro computacionalmente.

Captchas



- No estágio atual de desenvolvimento, alguns resultados em imagens com contorno bem definido.

Agradecimentos

- Ronaldo Fumio Hashimoto
- Wonder A. L. Alves
- Alexandre Noma