

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
FUNDAMENTOS DE PROCESSAMENTO DE IMAGENS
Prof. Manuel Menezes de Oliveira Neto

RELATÓRIO DO PROJETO FINAL
Camouflage Images (SIGGRAPH 2010)

Rafael Valer - 220489
Matheus Schilling Michel - 205684

Porto Alegre, 01 de Dezembro de 2014

1. Inspiração

Dentre os artigos apresentados pelo professor, o projeto escolhido para implementação foi sobre camuflagem de imagens. O artigo se chama “Camouflage Images”, publicado em 2010, na conferência anual de computação gráfica SIGGRAPH, tendo os autores: Hung-Kuo Chu, Wei-Hsin Hsu, Niloy J. Mitra, Daniel Cohen-Or, Tien-Tsin Wong e Tong-Yee Lee.

2. O artigo

O trabalho desenvolvido no artigo consiste basicamente em compor imagens com figuras escondidas, dificultando a percepção destas figuras pelo olho humano. A finalidade do algoritmo por eles criado é basicamente adiar a percepção das figuras ocultas, e para isso usam da teoria de que a percepção humana funciona em duas etapas: feature search e conjunction search.

Feature Search: etapa paralela, onde ocorre a procura por toda imagem por características como cores, arestas, texturas que permitam uma rápida caracterização de figuras.

Conjunction Search: já esta é uma etapa mais serial e demorada, e é responsável pelo reconhecimento de figuras através da integração de vários detalhes.

Algoritmos eficientes de camuflagem fazem com que a feature search seja praticamente impossível, forçando o reconhecimento de figuras somente através de conjunction search. As imagens resultantes geralmente são ricas em detalhes e cores, atraindo assim a atenção de quem as olha.

O algoritmo exemplificado no artigo realmente dificulta a feature search, visto que particiona elementos que poderiam ser agrupados em uma única característica, forçando assim a busca por conjuntos de características, em diferentes dimensões para a formação de alguma figura.

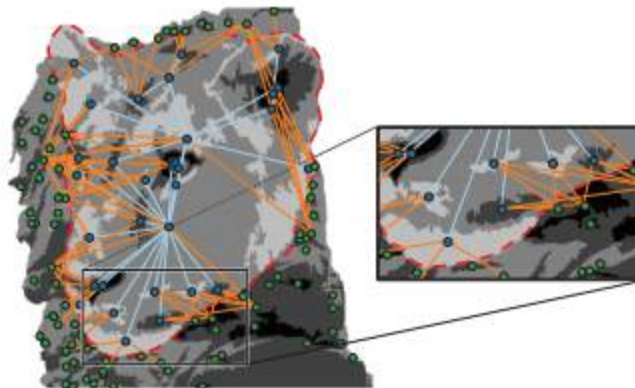
O Algoritmo:

Devido a importância dada ao fator de luminância para o reconhecimento de imagens, o algoritmo usa disso para esconder as figuras camufladas. A distribuição de luminância das imagens (foreground e background) são obtidas através de, primeiramente uma transformação para grayscale das imagens, e posteriormente uma quantização,

resultando em segmentos com valores variados de luminância, depois disso ocorria a segmentação entre as duas figuras, onde os segmentos da imagem de fundo seria cobertas pelos segmentos da imagem a ser camuflada quando as duas se sobrepunham.

Após isso é construído um grafo a partir da segmentação realizada entre as duas figuras, este grafo é gerado perante algumas regras, onde cada nodo representa um segmento da etapa anterior. Além disso, dois nodos estariam conectados somente se os dois pertencessem a um segmento da imagem a ser camuflada e compartilhassem uma aresta, ou se fossem de figuras diferentes, mas o segmento da imagem de fundo estivesse entre os "k" nodos mais próximos do segmento da imagem a ser camuflada, sendo k um valor arbitrário.

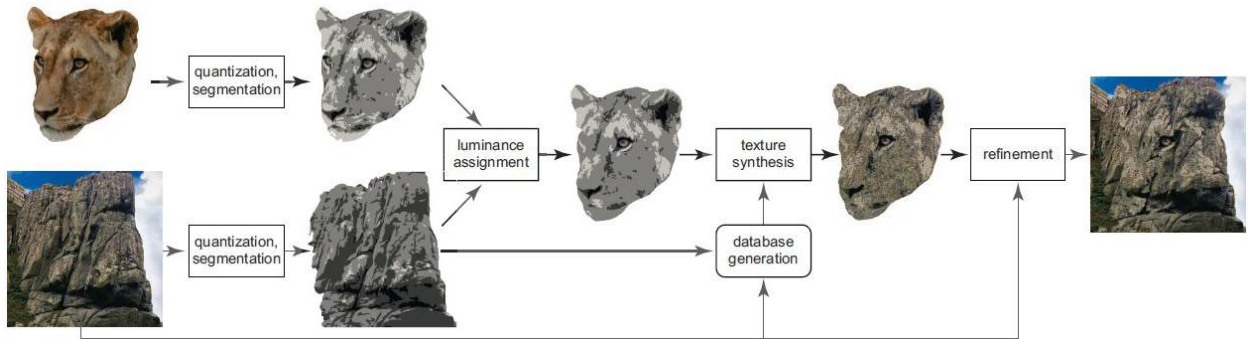
Exemplo de um grafo de luminância retirado do artigo:



Os segmentos do grafo têm seus valores de luminância modificados através de alguns critérios de conexão e um valor que representa a distância entre os segmentos, um nível de camuflagem também opera sobre as luminâncias, variando assim seus valores. Este processo faz com que o tempo de reconhecimento sobre as figuras de uma imagem aumente pois retira alguns detalhes da figura original.

Ao final do processo ocorre a síntese da textura, onde após a aplicação de luminância sobre a imagem de fundo, é guardado dela, em uma espécie de banco de dados, os pares de textura e segmentos de luminância, preenchendo primeiramente as arestas do segmentos, e então seus interiores.

A imagem a seguir exemplifica os passos do algoritmo:



3. Implementação:

Após a leitura e entendimento do artigo, iniciou-se o processo de desenvolvimento. Devido a familiaridade com c++ e o uso de Qt durante os pequenos trabalhos feitos durante o semestre, decidimos usá-los para a realização da tarefa. A janela principal possui duas áreas para carregar imagens, uma para o background e outra para as imagens que serão camufladas, após definir o nível de camuflagem e apertar no botão para início do processo, uma nova janela é criada com o resultado da camuflagem.

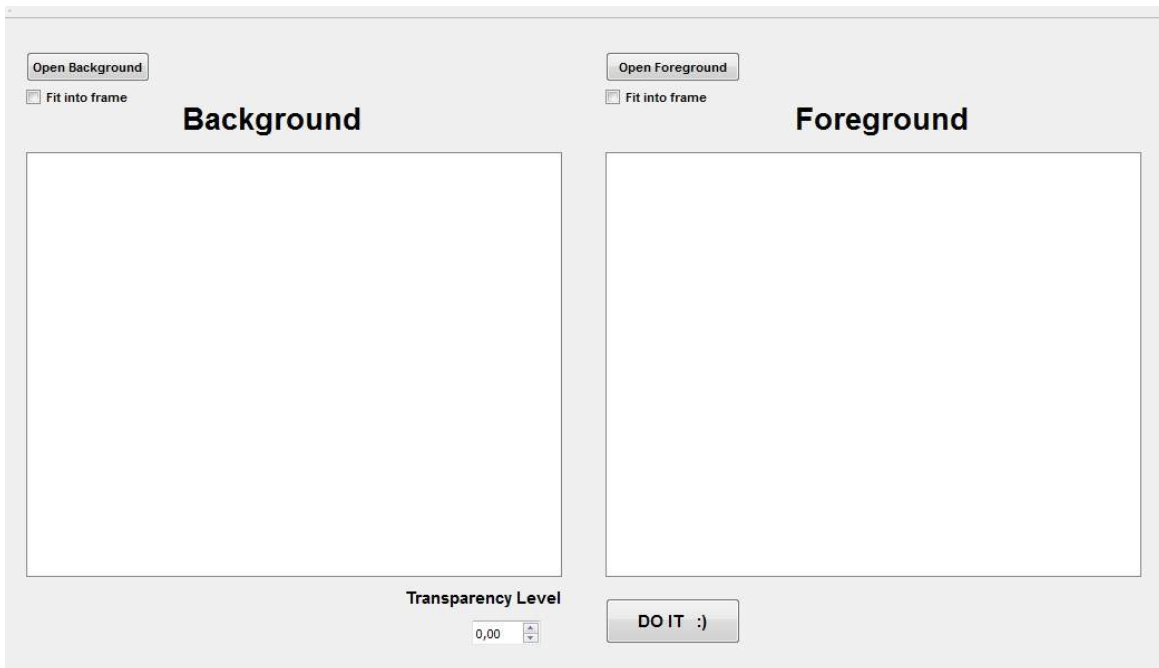


Imagem da tela principal do programa

Inicialmente tentamos implementar a solução proposta pelo artigo, conseguimos realizar a transferência para níveis de luminância e realizamos a quantização e segmentação proposta pelos autores, no entanto, não conseguimos realizar a construção do grafo para a continuação do algoritmo. Sendo assim, utilizando de alguns conceitos do artigo, utilizamos de uma técnica para construir nosso próprio algoritmo.

3.1. Algoritmo Desenvolvido

Assim como no artigo, para dificultar a feature search, o primeiro passo no algoritmo desenvolvido foi atribuir grayscale às imagens que serão camufladas, após isso, foi passado um filtro utilizando valores de um kernel gaussiano a essas imagens, suavizando os detalhes das imagens, tudo isso para dificultar a identificação das figuras escondidas.

Depois disso, primeiramente pensamos em passar um laplaciano às imagens a serem camufladas, e então detectar as arestas das mesmas, criando um mapa que continha as posições das arestas das imagens a serem camufladas. Com isso iríamos fazer uma composição com as duas imagens nos pixels determinados por arestas e nos outros pixels manteríamos o pixel original a imagem de fundo. No entanto esta aplicação nos retornou alguns maus resultados, fazendo com que somente o contorno da imagem aparecesse.

Não ficando satisfeitos, nosso próximo passo foi descartar esta ideia, nossa próxima implementação consistia em detectar espacialmente onde estavam as imagens a ser camufladas e descartar o que não seria parte da imagem resultante. Após isso, faríamos uma composição alfa para cada pixel que possuía uma imagem a ser camuflada, para isso, cada canal do pixel resultante seria igual a:

$$P_n = (\alpha * p_i) + (1 - \alpha) * p_f, \text{ onde:}$$


- P_n = pixel resultante
- α = coeficiente de transparência da imagem a ser camuflada (valor entre 0 e 1)
- p_i = pixel da imagem a ser camuflada
- p_f = pixel da figura de background

O resultado é uma imagem com composição somente nos pixels onde a imagem irá ser camuflada, respeitando o coeficiente de transparência (quanto maior o coeficiente, maior será a visibilidade da figura camuflada). Já os outros pixels serão iguais a imagem de background.

Segue os resultados do nosso algoritmo:

☒ Fit into frame

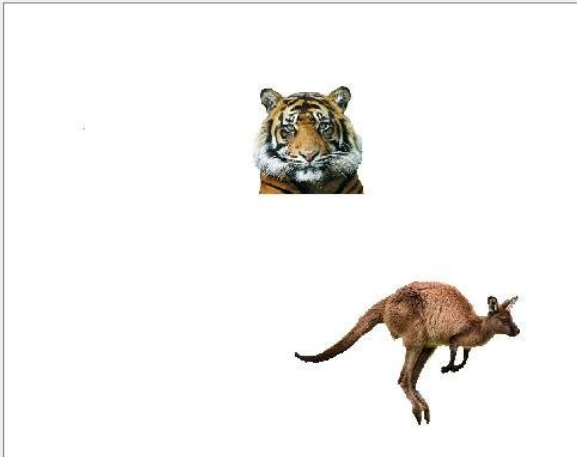
Background



Transparency Level
0,40

☒ Fit into frame

Foreground




Camouflage Result



Open Background

☒ Fit into frame

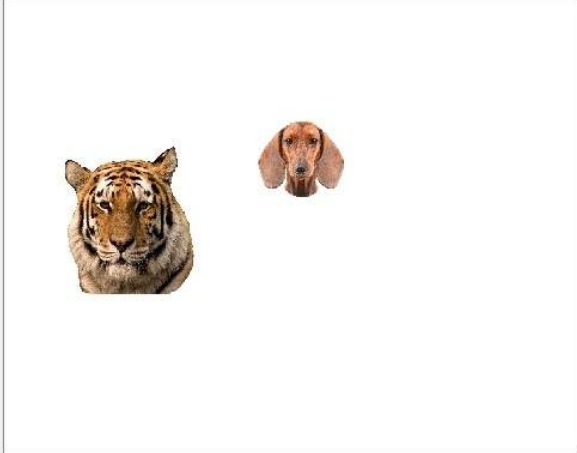
Background



Open Foreground

☒ Fit into frame

Foreground



Transparency Level

0,25

DO IT :)



4. Conclusão

Embora não pudemos ter realizado o algoritmo como apresentado no artigo devido a problemas técnicos, a experiência obtida pelo desenvolvimento de nosso próprio algoritmo de camuflagem de imagens foi muito proveitosa, pois nos exigiu muita reflexão sobre qual método escolher e como aplicá-lo para a realização da tarefa. Além disso, exigiu muito sobre um problema que não é muito comum aos nossos olhos, fazendo com que buscássemos alternativas e conhecimentos que não sabíamos até então.