



Fundamentos de Processamento de Imagens Trabalho Final

Crop and Drop

**Diogo Raphael Cravo
João Luiz Grave Gross**

Porto Alegre, 19 de dezembro de 2011.

Sumário

1. Propostas Inicial e Final
2. Material de Apoio
3. ☐ Objetivo
4. Algoritmo
 - 4.1 Exemplo
5. Problemas de implementação
6. Soluções de implementação
7. Estrutura do Projeto
8. Interface
9. Resultados
10. Pontos Positivos
11. Pontos Negativos
12. Conclusão

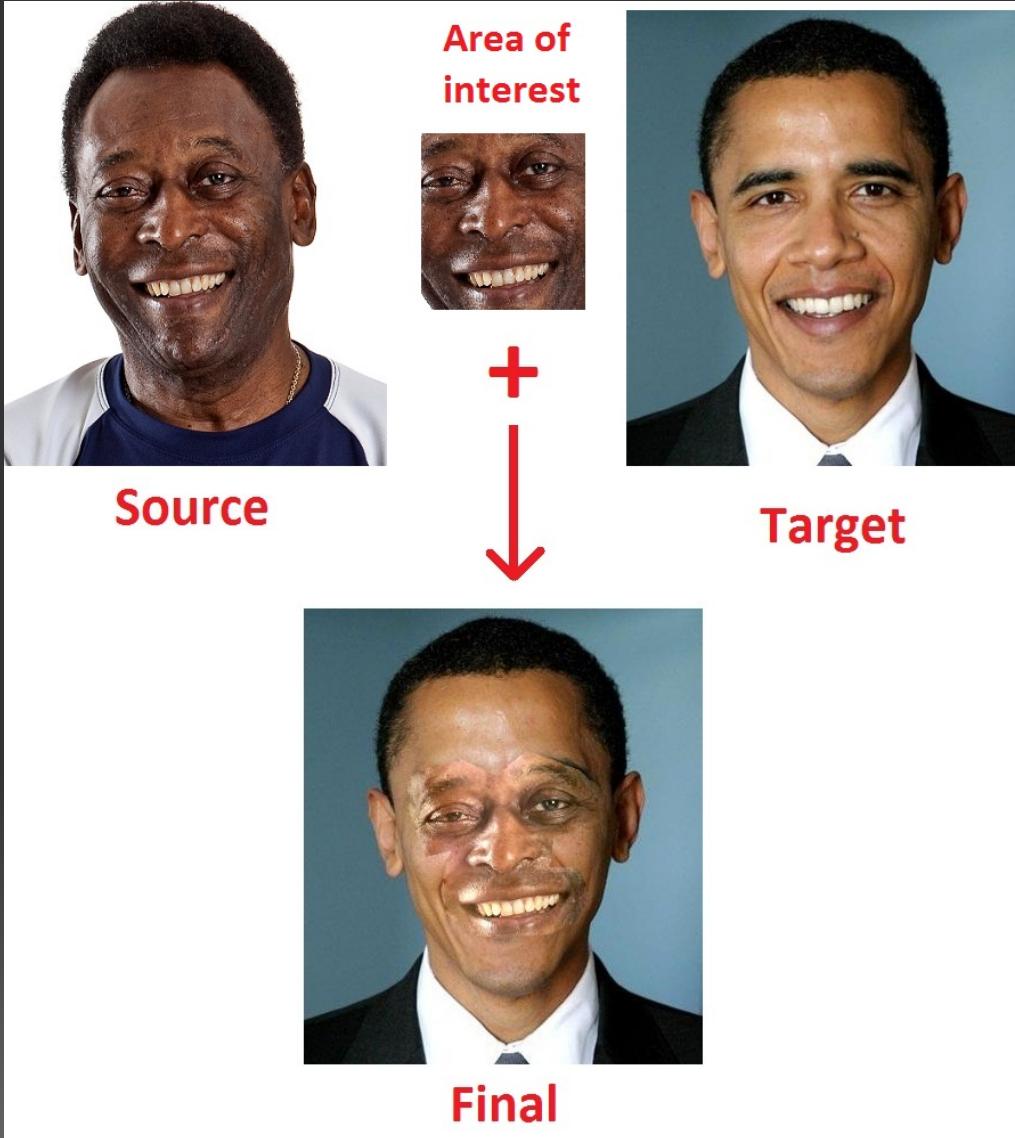
Propostas Inicial e Final

- Drag and Drop Pasting por Jia, et al. - 2009
 - Proposta interessante
 - Otimização da borda de seleção do trabalho de Perez, et al. - 2003
 - Muito complexo, faltou matemática...
- Poisson Image Editing por Perez, et al. - 2003
 - Era o que queríamos fazer. Selecionar pedaço de uma imagem, colar em outra e uní-las
 - Baseada da computação de sistemas lineares com equações e Poisson
 - Complexo, mas nem tanto...

Material de Apoio

- Artigo Poisson Image Editing - Pérez, et al. - 2003
- Projetos sobre Poisson Image Editing do Instituto de Ciência da Computação da Brown University - USA
 - materiais utilizados para compreender o algoritmo de Pérez
- Algoritmo/Implementação de um Linear System Solver

Objetivo



- Implementar composição de imagens
- Imagem final deve ser boa o suficiente para deixar o usuário na dúvida se aquela imagem é real ou não

Algoritmo

1. Selecionar área de interesse
2. Colar área de interesse sobre a imagem de destino
3. Iniciar composição
 - 3.1 Resolução de sistema de equações lineares do tipo
 $Ax = b$, onde:

Algoritmo

- Matriz A
 - matriz esparsa (com vários coeficientes zero)
 - para pixels p fora da máscara, as linhas em \mathbf{b} que correspondem aquelas linhas em \mathbf{A} terão a diagonal igual a 1, para indicar que o valor desse pixel já está definido
 - para pixels p dentro na máscara, a diagonal da linha em \mathbf{A} correspondente a linha em \mathbf{b} terá valor 4 e para os pixels q vizinhos a p é atribuído -1
 - se p está na posição (x,y) , seus vizinhos serão os pixels q $(x-1,y)$, $(x+1,y)$, $(x,y-1)$ e $(x,y+1)$

Algoritmo

- Vetor x
 - valores finais dos pixels, ou seja, a imagem final
 - quantidade de linhas igual a quantidade de pixels da imagem final
 - pixels fora da zona de interesse são conhecidos (estão prontos)
 - pixels dentro da zona de interesse são desconhecidos (é necessário resolver equações de poisson)

Algoritmo

- Vetor b
 - quantidade de linhas igual a quantidade de pixels da imagem final
 - para os pixels **p_i** conhecidos as correspondentes linhas **b_i** tem o valor copiado da imagem de destino
 - para os pixels **p_i** desconhecidos (dentro da área de interesse - imagem recortada) as correspondentes linhas **b_i** possuem as somas dos gradientes dos **n** vizinhos de **p_i** na imagem de origem, sendo **n** igual a 2, 3 ou 4

Algoritmo - Exemplo

RGB			
0x	red	green	blue
xx	xx	xx	xx

Image coordinates model

(0,0) | (0,1) | (0,2)

(1,0) | (1,1) | (1,2)

(2,0) | (2,1) | (2,2)

Gradient equation

$$4 * p(x,y) - p(x-1,y) - p(x+1,y) - p(x,y-1) - p(x,y+1)$$

Algoritmo - Exemplo

Source

0x00FF00 | 0x0000FF | 0xFF0000

0x00FF00 | 0x0000FF | 0xFF0000

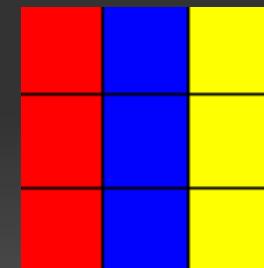
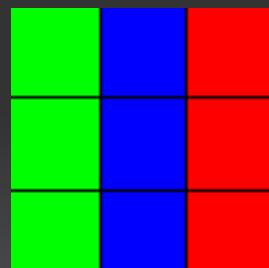
0x00FF00 | 0x0000FF | 0xFF0000

Source Gradient

0x00FE01 | 0x0002FD | 0xFEFF01

0x00FE01 | 0x0002FD | 0xFEFF01

0x00FE01 | 0x0002FD | 0xFEFF01



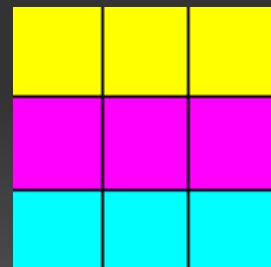
Algoritmo - Exemplo

Target

0xFFFF00 | 0xFFFF00 | 0xFFFF00

0xFF00FF | 0xFF00FF | 0xFF00FF

0x00FFFF | 0x00FFFF | 0x00FFFF



Algoritmo - Exemplo

```
A = [  
    1 0 0 0 0 0 0 0  
    0 1 0 0 0 0 0 0  
    0 0 1 0 0 0 0 0  
    0 0 0 1 0 0 0 0  
    0 -1 0 -1 4 -1 0 -1 0  
    0 0 0 0 0 1 0 0 0  
    0 0 0 0 0 0 1 0 0  
    0 0 0 0 0 0 0 1 0  
    0 0 0 0 0 0 0 0 1  
]
```

$x = [x_{00} \ x_{01} \ x_{02} \ x_{10} \ x_{11} \ x_{12} \ x_{20} \ x_{21} \ x_{22}]^T$

$b = [$

0xFFFF00	Final pixel
0xFFFF00	
0xFFFF00	
0xFF00FF	
$4 * 0x0002FD - 0xFEFF01 - 0x00FE01 - 2 * 0x0002FD$	Source Gradient
0xFF00FF	
0x00FFFF	
0x00FFFF	
0x00FFFF	
]	

Algoritmo - Exemplo

```
//Equação completa  
0*x00 - x01 + 0*x02 - x10 + 4*x11 - x12 + 0*x20 - x21  
+ 0*x22 = 4 * 0x0002FD - 0xFEFF01 - 0x00FE01 - 2 * 0x0002FD
```

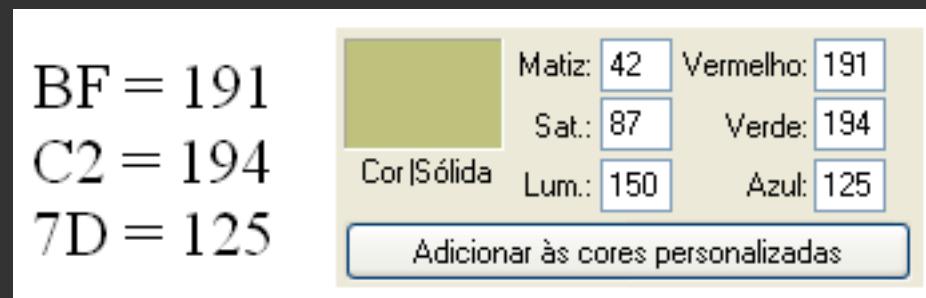
```
//Equação simplificada  
- x01 - x10 + 4*x11 - x12 - x21 = 0x0008F7
```

```
//Substituição de variáveis conhecidas  
- 0xFFFF00 - 0xFF00FF + 4x11 - 0xFF00FF - 0x00FFFF = 0x0008F7
```

Algoritmo - Exemplo

```
//Obtendo o pixels desconhecido
```

$$x_{11} = (0x0008F7 + 0xFFFF00 + 0xFF00FF + 0xFF00FF + 0x00FFFF) / 4 \\ = 0xBFC27D$$

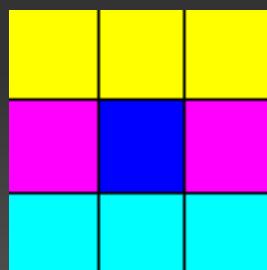


$$BF = 191$$

$$C2 = 194$$

$$7D = 125$$

Resolução $Ax = b$



Pixel Posicionado

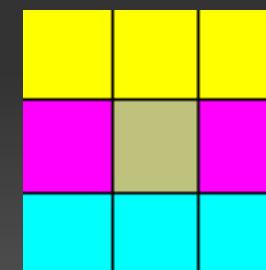
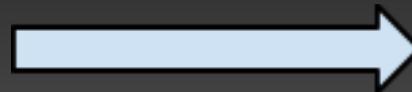


Imagen Final

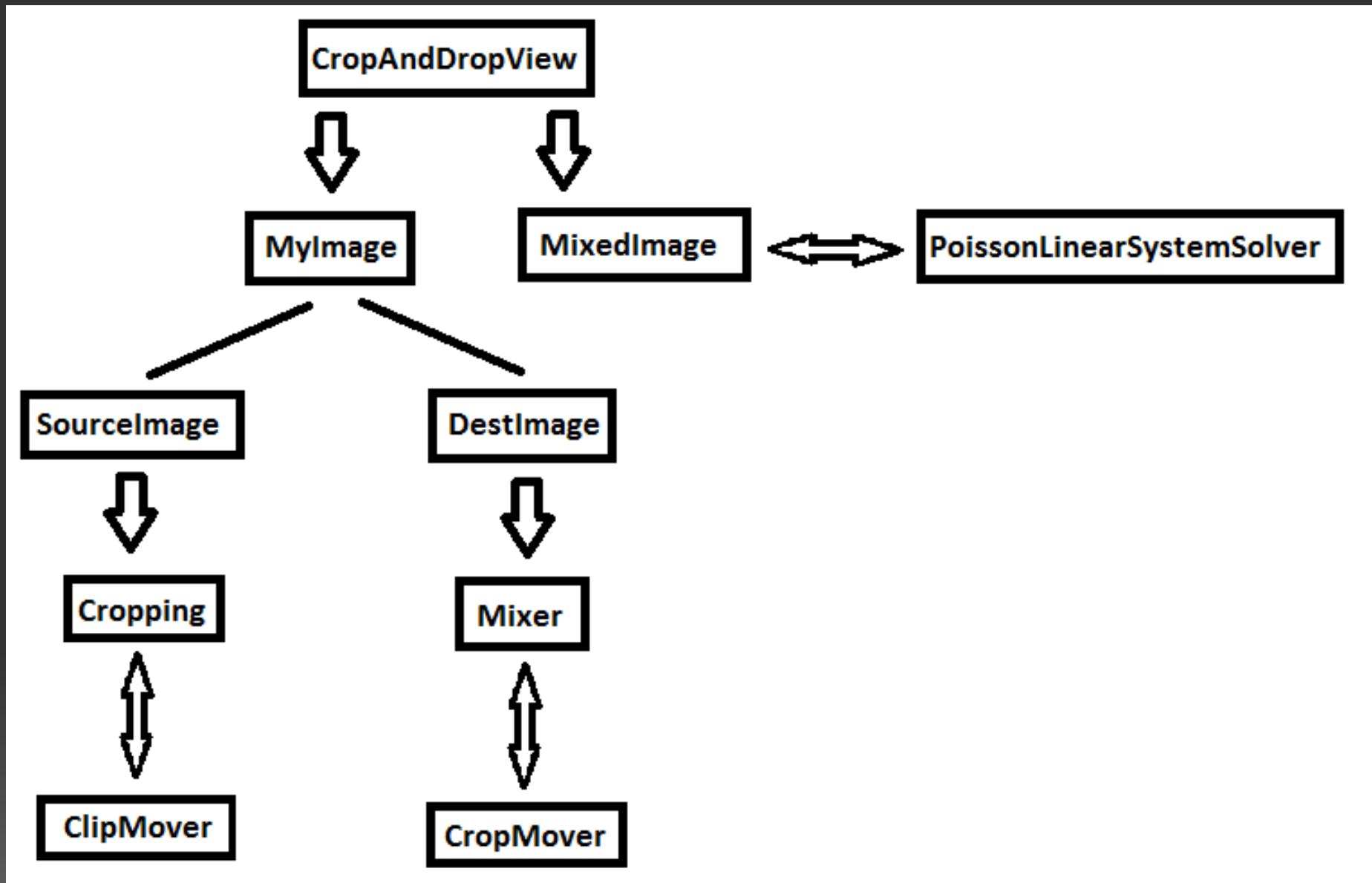
Problemas de implementação

- Implementar seleção da imagem de origem com área de seleção móvel (com o mouse)
- Matriz esparsa é muito grande
- Implementar solver de sistema linear de equações para uma matriz **GIGANTE**, como é a nossa (uma área recortada de 500x500 pixels necessitaria de 7,27GB)
- Procurar uma forma otimizada de representar uma matriz esparsa (matriz cheia de coeficientes zero)

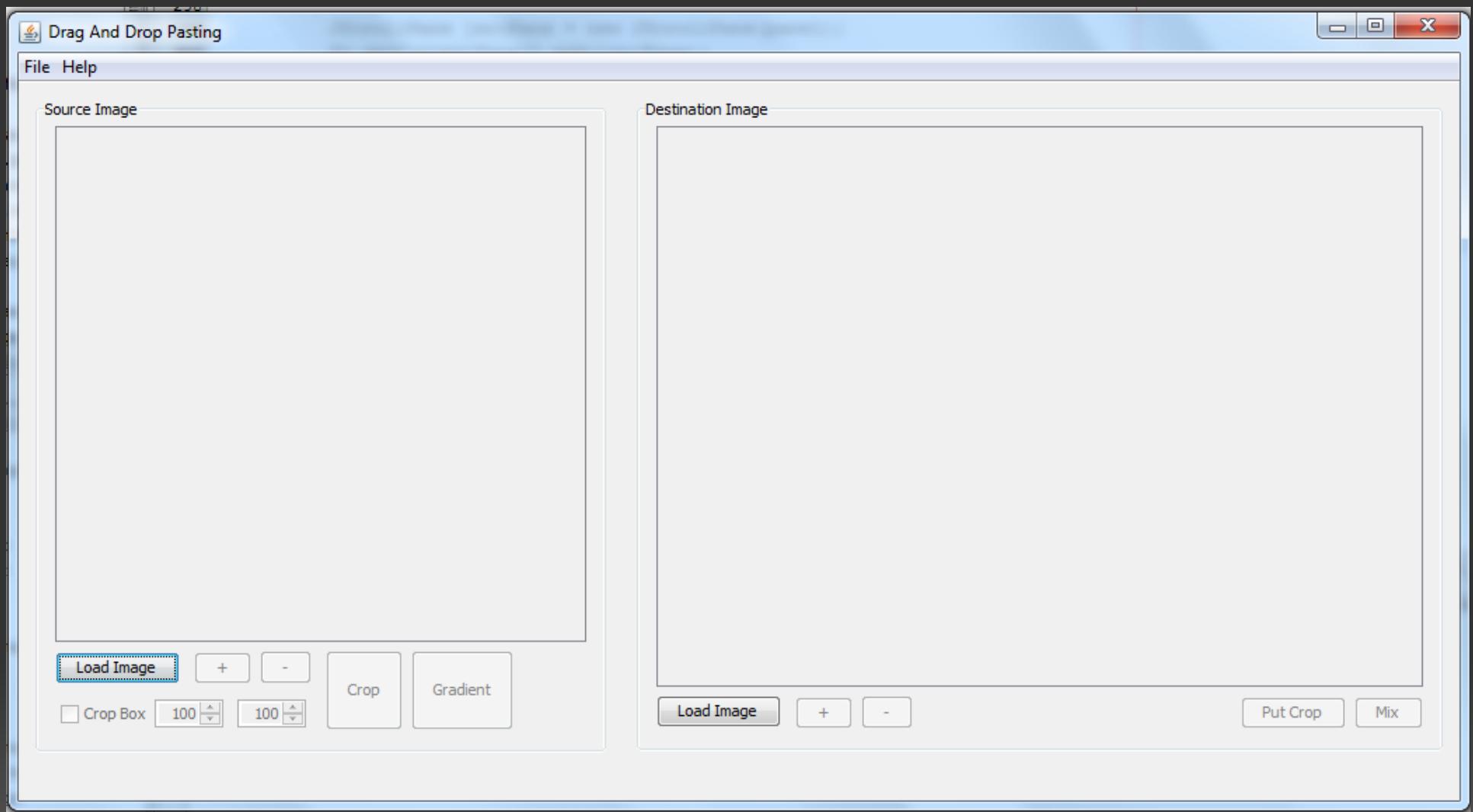
Soluções de implementação

- Não encontramos algoritmo eficiente para resolver matriz esparsa em Java, logo a resolução do sistema $Ax = b$ é feita em MatLab
- O programa em Java teve seu propósito alterado. Ele é usado apenas para selecionar as imagens, gerar o recorte, a máscara e prepará-las para nosso programa em MatLab

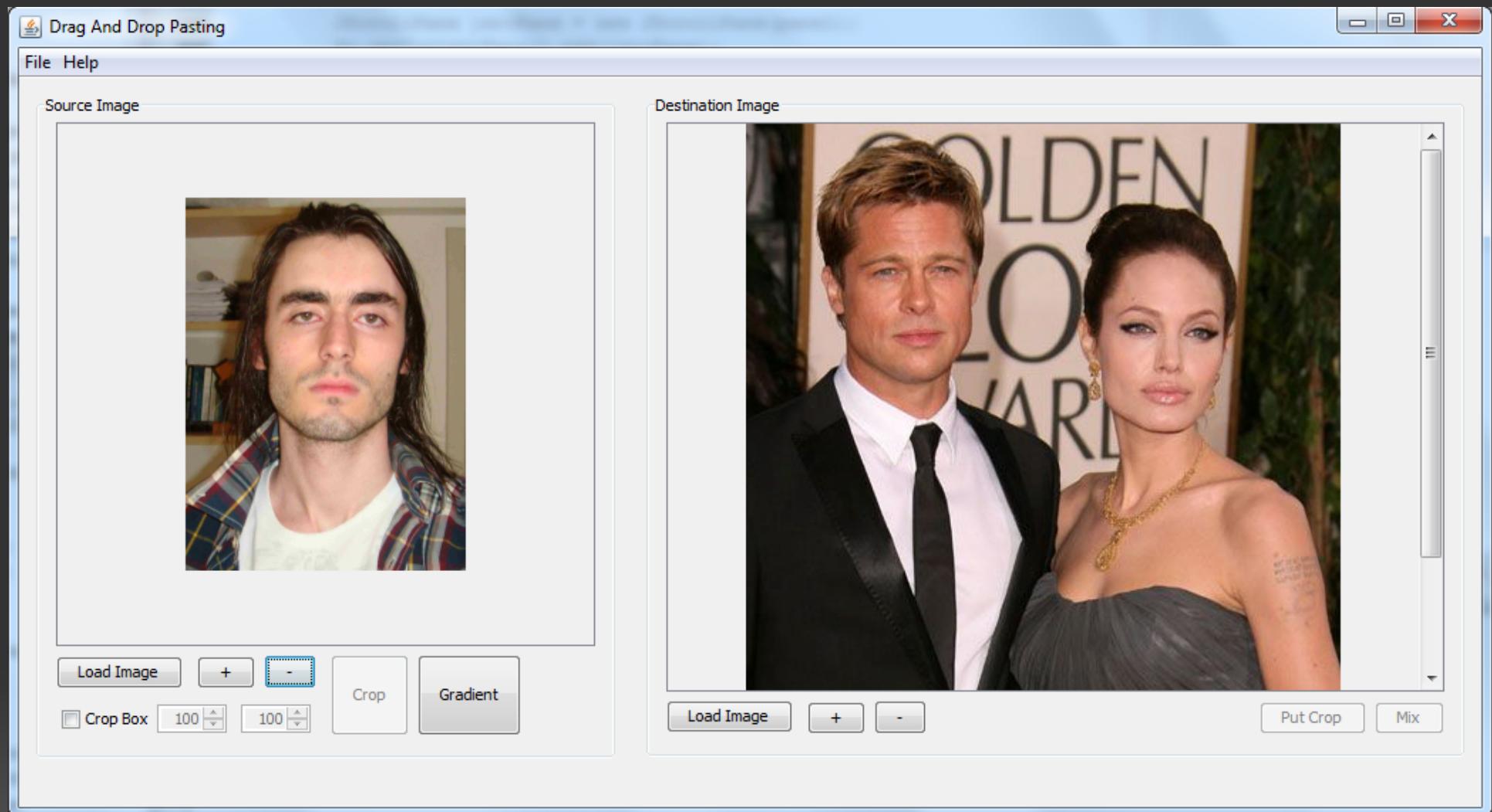
Estrutura do projeto



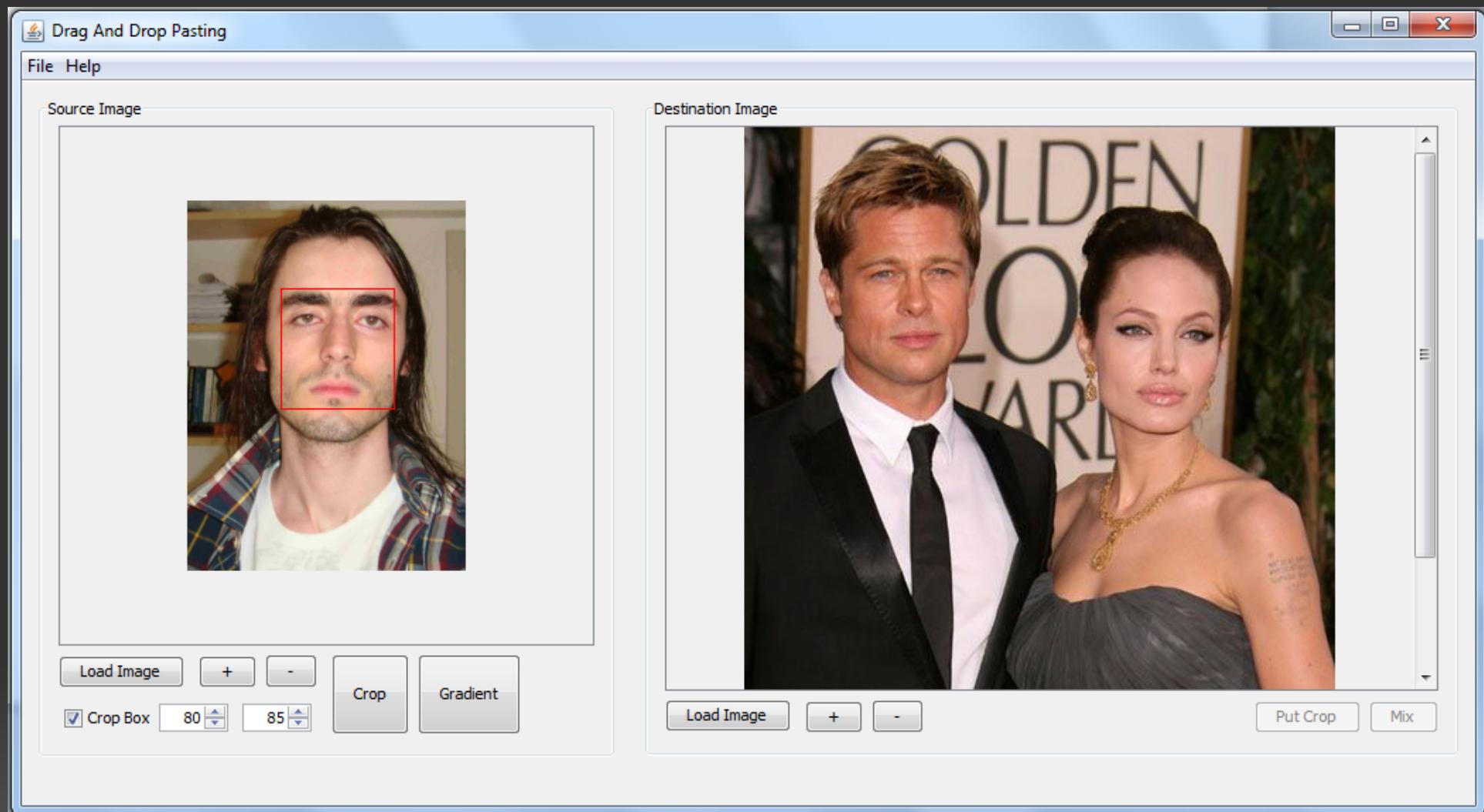
Interface



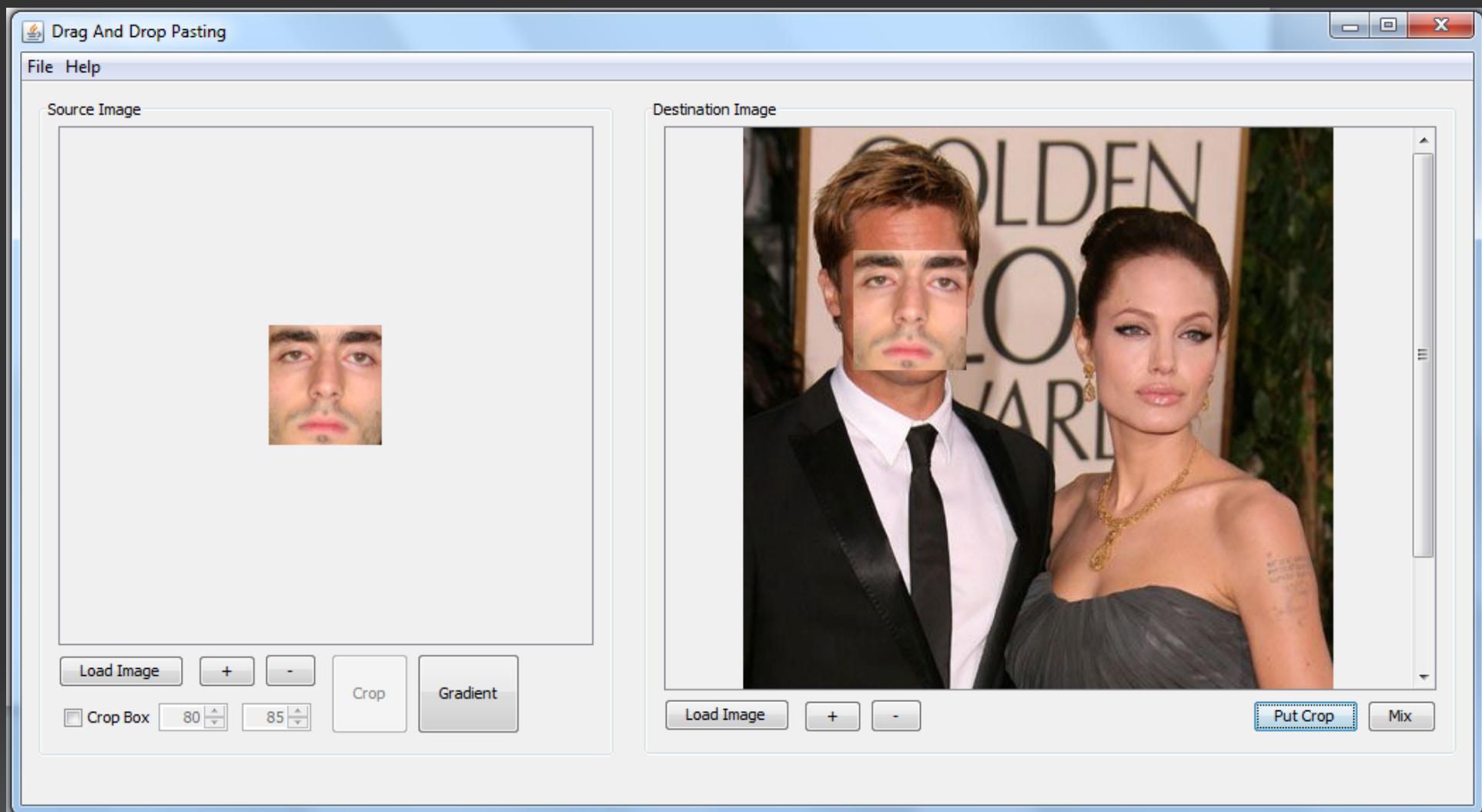
Interface



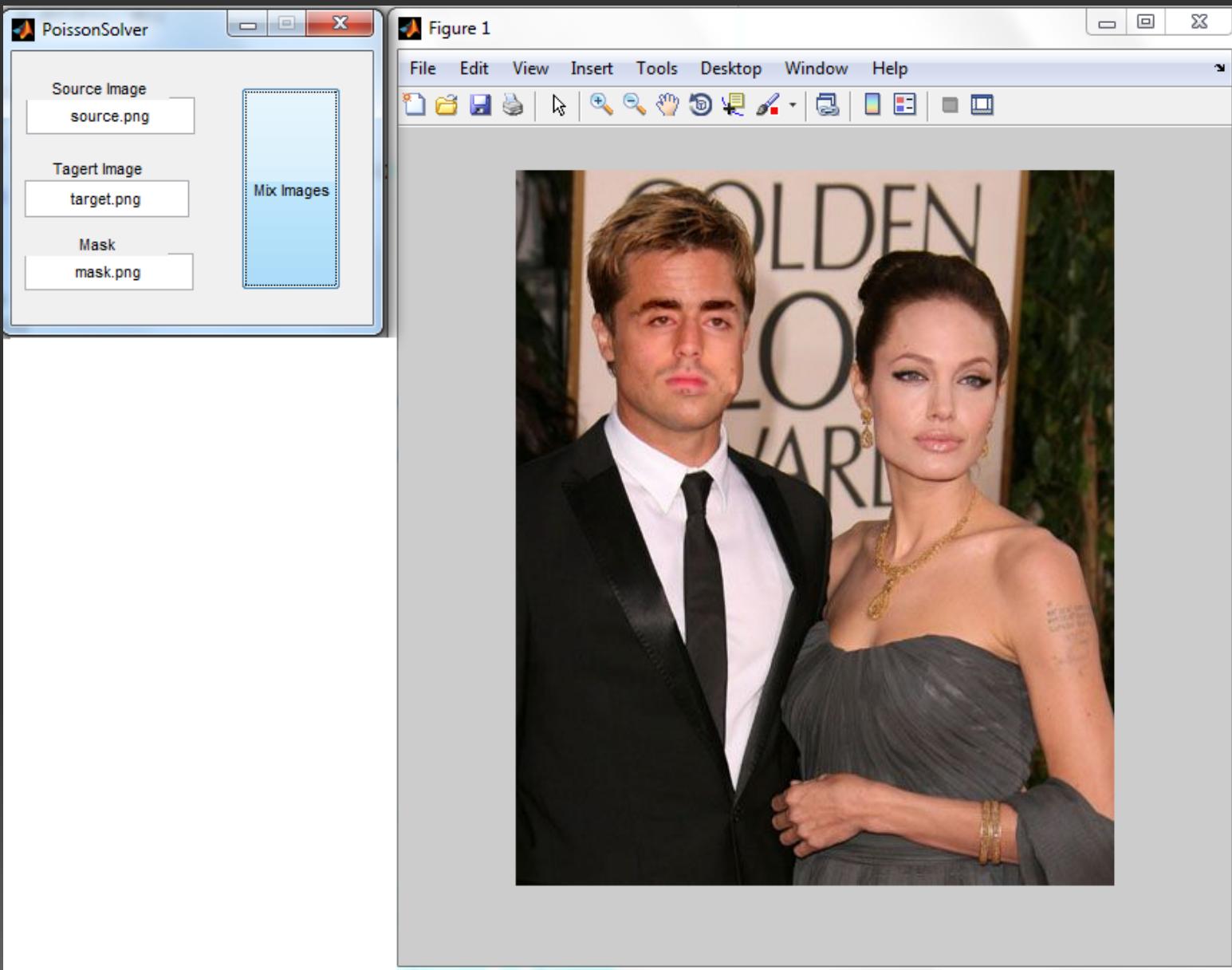
Interface



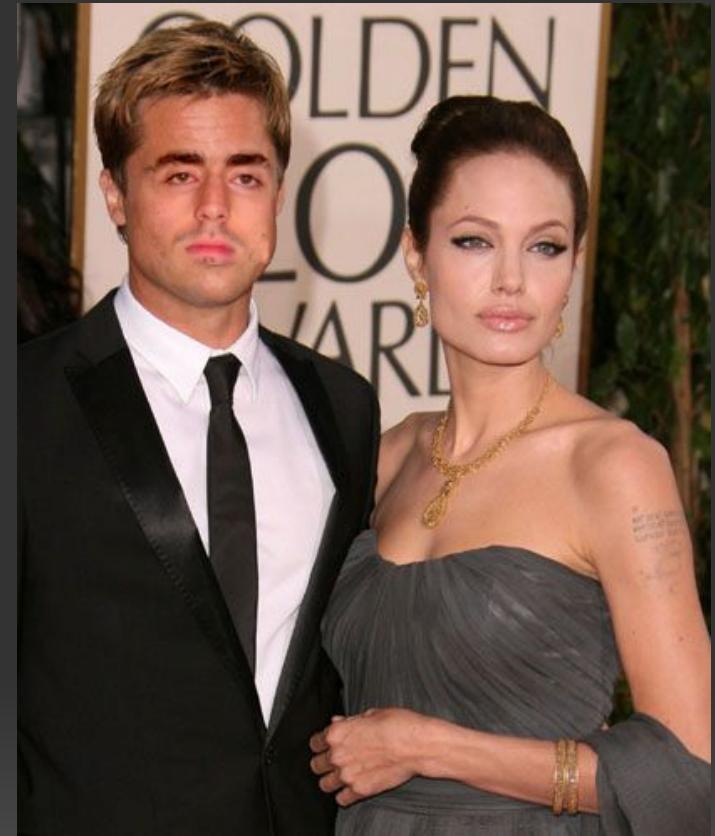
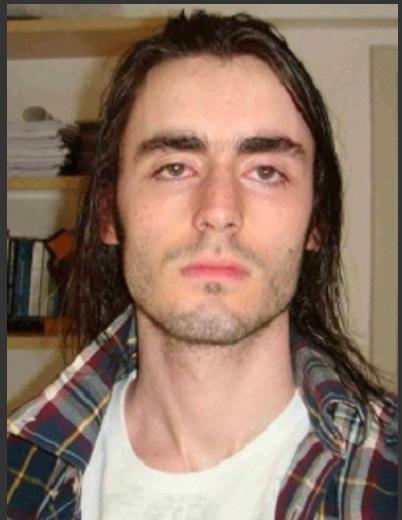
Interface



Interface



Resultados



Resultados



Resultados



Resultados



Resultados



Pontos Positivos

- + Melhor compreensão a respeito da representação RGB, gradientes e convolução
- + Tecnologias Java antes não manipuladas
- + Bons resultados
- + Possibilidade de trabalhar novamente com MatLab

Pontos Negativos

- Muita demora até entender o algoritmo
- Implementação demorou para ser concluída, logo não pôde ser refatorada visando melhorias
- Implementação em Java do core matemático da aplicação não funcionou
- Gostaríamos de ter feito versionamento de código
- Google docs

Conclusão

Acreditamos que este projeto agregou muito para à disciplina como complemento de nosso aprendizado, pois pudemos pôr em prática uma aplicação de algumas tecnologias vistas em aula, bem como conhecer outras e/ou aprimorar as já conhecidas.

Obrigado!

Diogo Raphael Cravo (diogo.rafael.cravo@gmail.com)
João Luiz Grave Gross (joaoluizgg@gmail.com)