

# Photomontage avec les graph-cuts

Mariia Drozdova, Runtian Zhang

Mars 2018

## 1 Introduction

Le but de ce projet est de faire un photomontage en utilisant la théorie des graph-cuts. Nous voudrions avoir une image finale qui semble naturelle alors que l'interaction humaine etait minime.

## 2 Exemples

### 2.1 Deux images

Premièrement, le but était de fusionner deux photos identiques pour pouvoir créer des images contenant beaucoup de petits objets : des fruits rouges, des légumes, une foule etc. Dans cet exemple on a utilisé une pelouse.

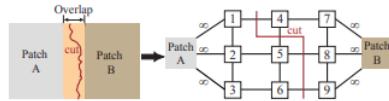


Figure 2: (Left) Schematic showing the overlapping region between two patches. (Right) Graph formulation of the seam finding problem, with the red line showing the minimum cost cut.

FIGURE 1 –

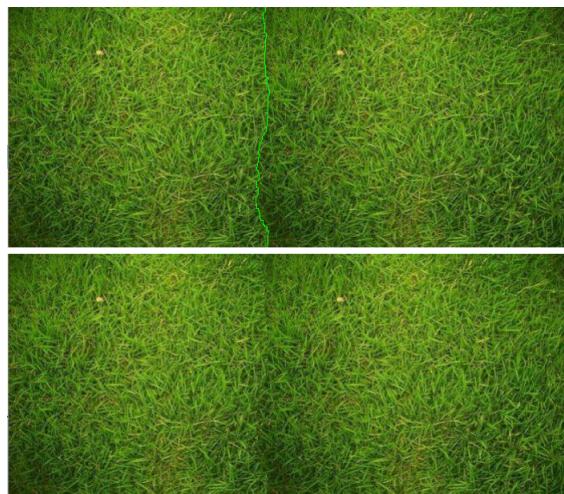


FIGURE 2 –

En utilisant *Entire patch matching*, nous avons aussi réussi à fusionner des photos avec une texture très régulière, voici l'exemple :

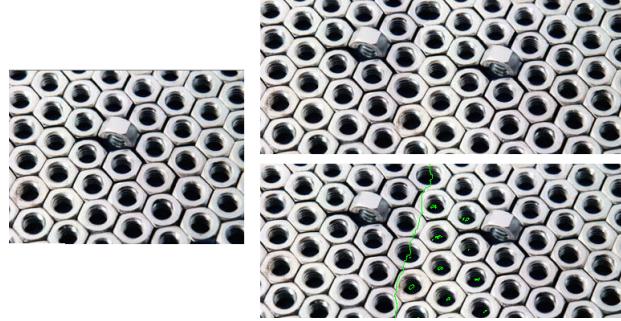


FIGURE 3 –

## 2.2 Photomontage

Dans cette partie, d'après plusieurs photos nous voudrions créer une image qui contient les parties de chaque photo afin de créer une photo idéale. Pour cela, l'algorithme d'alpha-expansion step a été utilisé dans deux fonctions alpha\_step et alpha\_step2 : optimal minimum et la version plus rapide basée sur l'article proposé ("Interactive Digital Photomontage" par Aseem Agarwala, Mira Dontcheva). Le but était d'avoir l'image la plus naturelle possible, et donc que la fonction optimale a bien été utilisée. Dans ce cas, les labels ont été choisis à l'aide du code direct.

$$E(f) = \sum_{\{p,q\} \in \mathcal{N}} V_{p,q}(f_p, f_q) + \sum_{p \in \mathcal{P}} D_p(f_p)$$

$$D_p(f_p) = \begin{cases} \infty & \text{label of } p \text{ is right} \\ 0 & \text{label of } p \text{ is not right} \end{cases}$$

$$V_{p,q}(f_p, f_q) = \|I_1(p) - I_2(p)\| + \|I_1(q) - I_2(q)\|$$

Le principe est de minimiser l'énergie qui est calculée avec cette formule d'après le graph construit :

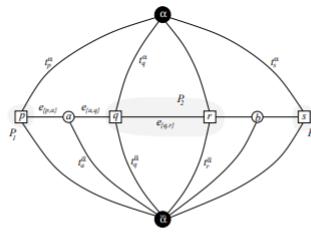


Figure : An example of  $\mathcal{G}_\alpha$  for a 1D image. The set of pixels in the image is  $\mathcal{P} = \{p, q, r, s\}$  and the current partition is  $\mathbf{P} = \{\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_\alpha\}$  where  $\mathcal{P}_1 = \{p\}$ ,  $\mathcal{P}_2 = \{q, r\}$ , and  $\mathcal{P}_\alpha = \{s\}$ . Two auxiliary nodes  $a = a_{\{p,q\}}$ ,  $b = a_{\{r,s\}}$  are introduced between neighboring pixels separated in the current partition. Auxiliary nodes are added at the boundary of sets  $\mathcal{P}_t$ .

FIGURE 4 – Graph construit

Pour cela la fonction "maxflow" est utilisée.

edge	weight	for
$t_p^\alpha$	$\infty$	$p \in \mathcal{P}_\alpha$
$t_p^\alpha$	$D_p(f_p)$	$p \notin \mathcal{P}_\alpha$
$t_p^\alpha$	$D_p(\alpha)$	$p \in \mathcal{P}$
$e_{\{p,\alpha\}}$	$V(f_p, \alpha)$	
$e_{\{q,p\}}$	$V(\alpha, f_q)$	$\{p, q\} \in \mathcal{N}, f_p \neq f_q$
$t_a^\alpha$	$V(f_p, f_q)$	
$e_{\{p,q\}}$	$V(f_p, \alpha)$	$\{p, q\} \in \mathcal{N}, f_p = f_q$

FIGURE 5 – Les formules utilisées



FIGURE 6 – Trois images initiales : (a) l'image tachée par la ligne noire, une personne sur deux sourit ; (b) une personne sur deux sourit ; (c) l'image tachée par la ligne rouge, une personne sur deux sourit.

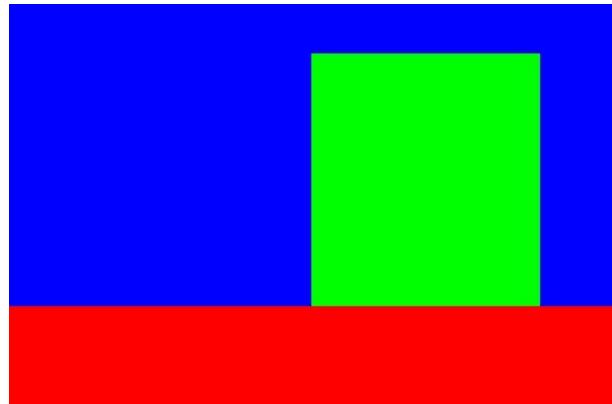
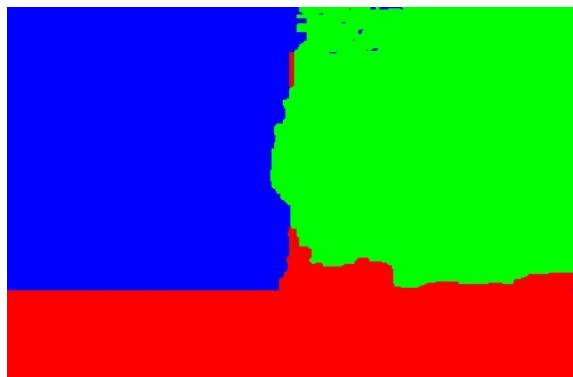


FIGURE 7 – Label souhaité par l'utilisateur



(a)



(b)

FIGURE 8 – Résultats : (a) Label finale ; (b) Image ;

## 2.3 Photomontage interactif

Pour pouvoir plus facilement choisir les parties des photos désirables, le choix des parties des photos de façon interactive a été réalisé (le principe de "Paint"). L'utilisateur sélectionne successivement dans chaque image la partie qu'il souhaite garder. Si pour le pixel  $(p, q)$  aucun pixel correspondant n'était pas choisi, l'algorithme choisit lui-même. A partir de ces labels, alpha-expansion entre en marche. Ci-dessous un exemple est présenté.



FIGURE 9 – Les images initiales



FIGURE 10 – Labels interactivement choisis par l'utilisateur



FIGURE 11 – Labels donnés par "alpha-expansion"



FIGURE 12 – Result

Les frontières entre deux labels différents sont très remarquable. Pour éviter cet effet le poisson-blending a été réalisé par résolvant  $Ax = b$  à la base de la solution donnée par Eric Yuan. ( <http://eric-yuan.me/poisson-blending/> )



FIGURE 13 – Result avec blending

## 2.4 Les autres exemples

### 2.4.1 Family



FIGURE 14 – Les photos originaux

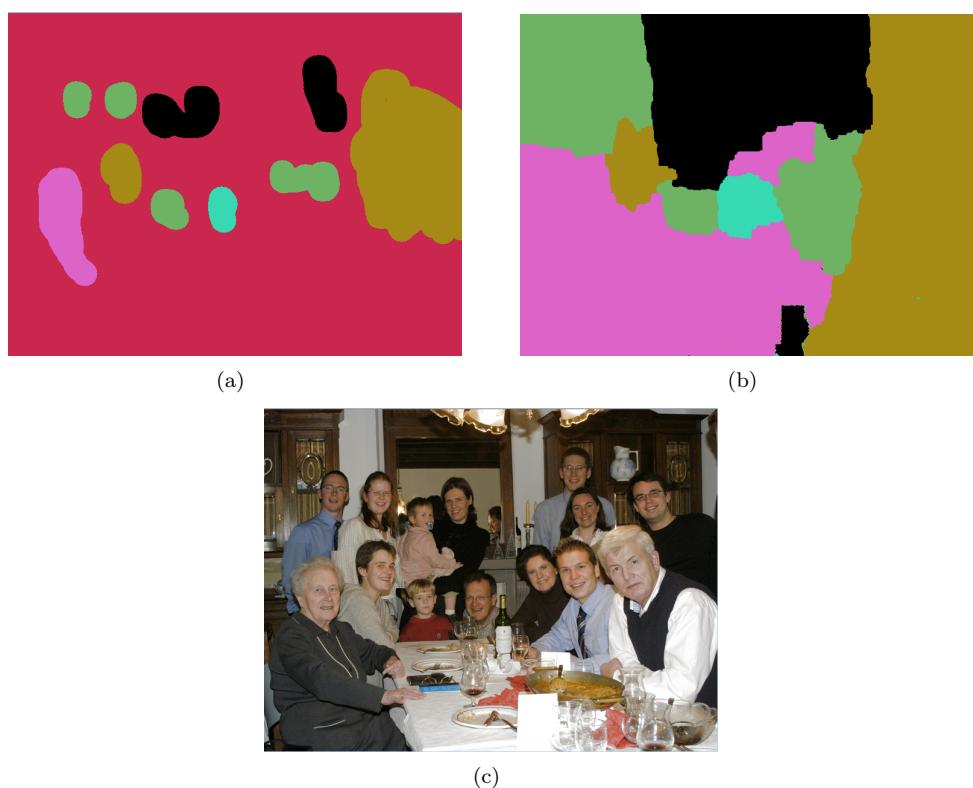


FIGURE 15 – (a) Labels diserés ; (b) Labels donnés par "alpha-expansion" ; (c) Result.



FIGURE 16 – (a) Les autres labels dispersés ; (b) Labels donnés par "alpha-expansion" ; (c) Result ;  
(c) Result avec poisson-blending.

### 2.4.2 Test

Dans ce test nous avons trois images monochromatiques. Il est fait pour vérifier que le programme marche correctement.

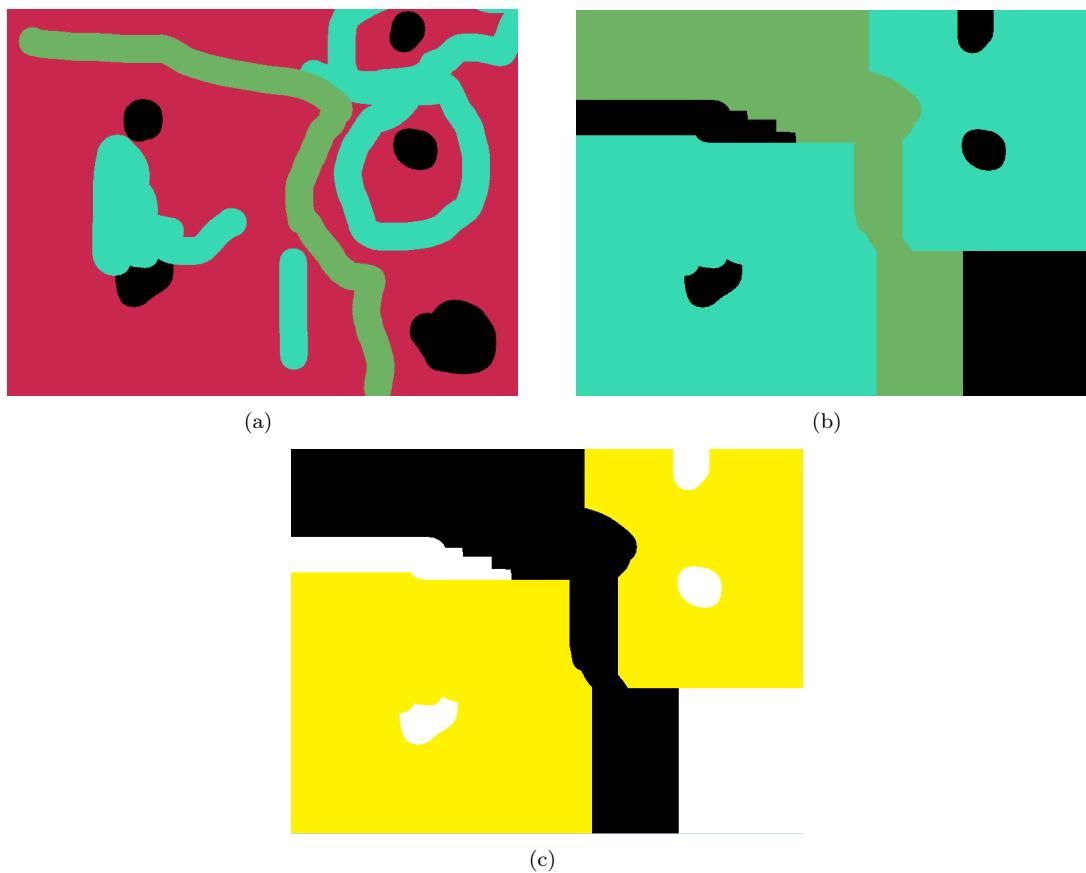


FIGURE 17 – (a)Labels donnés ; (c) Labels donnés par "alpha-expansion".

En plus nous pouvons predire que l'image obtenue à la fin doit dépendre de l'ordre des alphas. Cela peut être vérifier en utilisant ce programme. Il y a une démonstration avec le fichier "test".

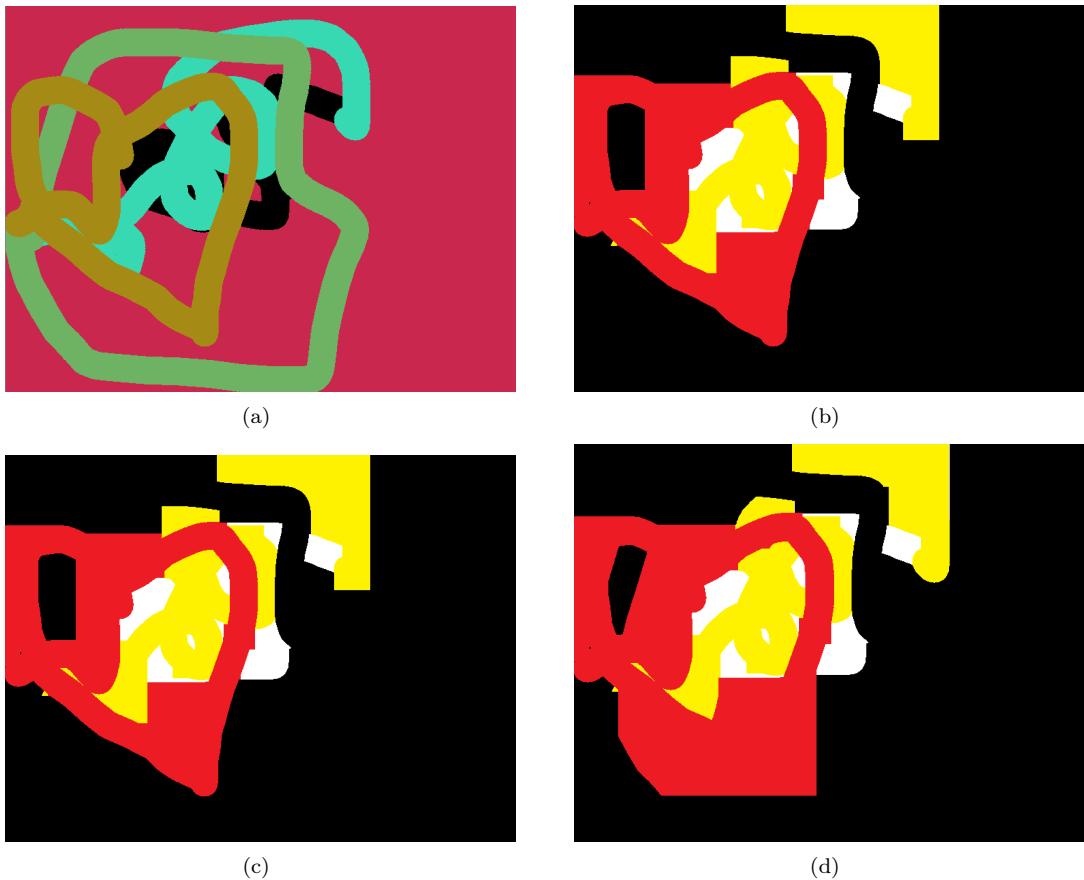


FIGURE 18 – Correspondence : 0 - blanc, 1 - jaune, 2 - noire, 3 - rouge.(a)Labels donnés ; (c) Labels obtenu avec l'ordre 0123 ; (c) Labels obtenu avec l'ordre 0132 ; (c) Labels obtenu avec l'ordre 3012.

### 3 Structure du programme

**GraphCuts.cpp** : contient Tests(image longue non structurée/srtucturée/photomontage : family/test/cathedral/ceremonie photos)

**alphaexpansion** : tous les fonction d'alpha-expansion ainsi les fonctionnes pour calculer energie de l'image et pour appliquer poisson-blending

**blending** : les fonctionnes de poisson-blending

**distance** : les fonctionnes pour calculer les poids des côtes

**openfiles** : une fonction primitive pour ouvrir les fichiers

**unitedimage** : les fonctions pour former les images longues non structurées/srtucturées