Projet Programmation dynamique
Mise en page automatique d'une bande dessin*

26 mars 2020

1 Description du programme

Le programme qu'on vous demande d'implémenter prendra en entrée une série ordonnée de cases de bande dessinée (c'est-à-dire des images), une largeur de page et une épaisseur de bords. Il devra fournir en sortie une nouvelle image de cette largeur sur laquelle les cases auront été rangées de gauche à droite et de bas en haut (en respectant l'épaisseur des bords). Ces cases auront également été étirées ou réduites grâce à une technique particulière, appelée "Seam carving" [AS07], de manière à occuper complètement (aux bords près) la largeur de l'image générée.

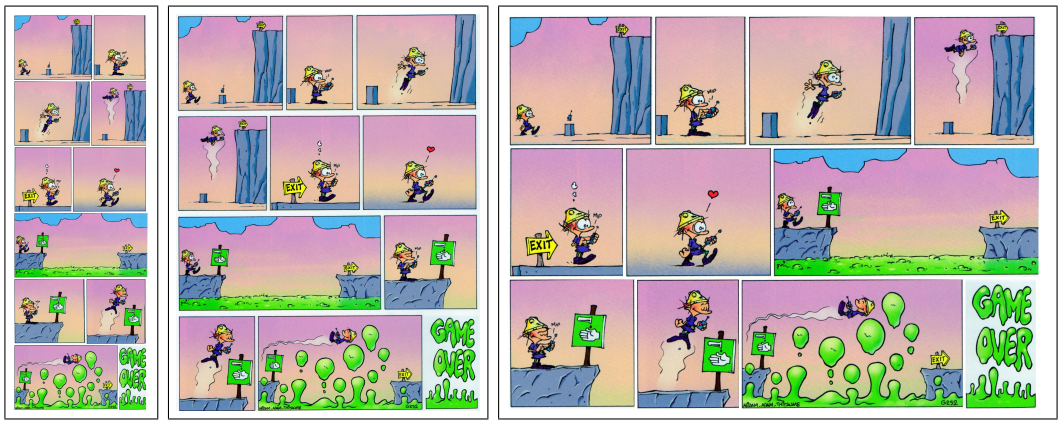

FIGURE 1 – Trois exemples de cases disposées en rectangles de tailles différentes.

*Ce projet a été crée par Pierre Geurts et Jean-Michel Begon

L'implémentation de cette méthode se fera en deux étapes :
— À partir de la largeur de page fournie, de la largeur des bords et des largeurs des différentes cases, on déterminera le nombre de rangées de la page finale et la répartition des différentes cases sur ces rangées. Cette répartition sera déterminée à l'aide d'un algorithme généralement utilisé pour faire de la justification de texte.
— À partir de la répartition obtenue des cases sur les différentes rangées de la planche, on déterminera pour chaque rangée si les cases doivent être réduites ou étendues de manière à remplir complètement la largeur de la page. Ces transformations seront ensuite appliquées aux cases en utilisant l'algorithme de Seam Carving et les cases ainsi transformées seront placées (de gauche à droite et de haut en bas) sur l'image finale.
L'algorithme de répartition des cases et le seam carving font tous deux appel à la programmation dynamique. Ces deux algorithmes sont décrits ci-dessous.

1.1 Algorithme de répartition des cases

L'algorithme pour déterminer le placement des cases sur l'image finale de largeur donnée W est similaire à un algorithme de justification de texte. À partir d'un tableau $l[1..n]$ contenant les largeurs, en nombre de pixels, des n cases dans leur ordre d'apparition sur la planche, il s'agit de déterminer un placement des cases rangée par rangée. Ce placement sera déterminé avec pour objectif de minimiser les transformations de largeur des cases, c'est-à-dire en minimisant pour chaque rangée le nombre de pixels à remplir (par un élargissement) ou à gagner (par une réduction) pour satisfaire la contrainte de largeur de page.
De manière plus formelle, afin de définir le coût d'une répartition, on définira les deux fonctions (ou tableaux) suivantes :
— $\texttt{extras}(i, j)$ (pour $1 \leq i \leq j \leq n$) qui calcule la différence entre la largeur de la page W et la somme des largeurs des cases i à j. $\texttt{extras}(i, j)$ représentera donc l'espace vide ou l'excédent par rapport W lorsqu'on place les cases i à j sur une même rangée.
— $\texttt{cost}(i, j)$ (pour $1 \leq i \leq j \leq n$) attribuant un coût au placement des cases i à j sur la même rangée. Plusieurs définition de coût sont possible. Dans le cadre du projet, on vous propose d'utiliser comme valeur de coût $c(i, j) = |\texttt{extras}(i, j)|$, qui pénalise de manière symétrique les déficits et les dépassements de largeur.
Le coût d'une répartition est alors obtenu en sommant les valeurs $c(i, j)$ pour toutes les rangées. Par exemple, le coût pour le placement suivant de 8 cases :

1	2	3
4	5	
6	7	8

sera donné par $c(1, 3) + c(4, 5) + c(6, 8)$. Pour déterminer la répartition de coût minimal, on peut utiliser la programmation dynamique. La fonction de coût à formuler de manière récursive sera $c(n)$, le coût d'une répartition optimale des cases 1 à n.

1.2 Algorithme de recadrage intelligent d'images ("Seam Carving")

Une fois, l'arrangement optimal des cases obtenu, celles-ci seront redimensionnées de manière à remplir complètement la largeur de la page. Le redimensionnement d'image consiste à faire passer une image d'une résolution initiale $n \times m$ à une résolution cible $n' \times m'$ différente. Ici, on se focalisera sur les modifications de la largeur uniquement de l'image ($n = n'$). Les techniques les plus classiques de redimensionnement sont la mise à l'échelle ("rescaling") et le recadrage ("cropping"). L'inconvénient de la première approche est qu'elle déforme l'image si le ratio n'/m' est différent du ratio n/m original (ce qui est la cas pour notre application puisqu'on maintiendra la hauteur des cases fixée). L'inconvénient du recadrage classique est qu'il résulte en une perte d'information. L'idée du recadrage intelligent est comme pour le recadrage classique de supprimer des pixels de l'image originale mais en sélectionnant ces pixels non pas sur les bords de l'image mais comme étant les pixels les moins importants de l'image. La taille de l'image est réduite d'un pixel de largeur à la fois. A chaque étape, les pixels supprimés sont sélectionnés le long d'un chemin (appelé couture, ou "seam" en anglais) allant du haut au bas de l'image, ce chemin étant choisi de manière à minimiser l'importance totale des pixels qu'il traverse.
Les différentes étapes de l'algorithme sont décrites en détail ci-dessous.

1.2.1 Énergie

L'importance d'un pixel (i, j) est quantifée par le biais d'une mesure d'énergie, notée $E(i, j)$. Plus l'énergie d'un pixel est élevée, plus ce pixel est important et doit donc être maintenu dans l'image. Plusieurs mesures d'énergie sont possibles. Dans le contexte de ce projet, on utilisera la mesure d'énergie suivante (correspondant au gradient de l'image) :

$$E(i, j) = E(i, j, \mathrm{R}) + E(i, j, \mathrm{G}) + E(i, j, \mathrm{B}),$$

où

$$E(i, j, c) = \frac{|I(i-1, j, c) - I(i+1, j, c)|}{2} + \frac{|I(i, j-1, c) - I(i, j+1, c)|}{2},$$

avec $I(i, j, c)$ la valeur associée au pixel (i, j) dans le canal de couleur $c \in \{\mathrm{R}, \mathrm{G}, \mathrm{B}\}$ de l'image. Pour les pixels au bord de l'image, le ou les valeurs de pixels inexistantes seront remplacées dans la formule par la valeur du pixel lui-même $I(i, j, c)$.

1.2.2 Couture d'énergie minimale

Une couture (verticale) s_v dans une image de taille $n \times m$ est définie comme une suite ordonnée de pixels $(1, j_1), (2, j_2), ..., (n, j_n)$ telle que $j_k \in \{1, ..., m\}$ pour tout $1 \leq k \leq n$ et telle que $|j_k - j_{k-1}| \leq 1$ pour tout $2 \leq k \leq n$. La suite de pixels correspondant à une couture verticale définit donc un chemin connecté de pixels allant du haut vers le bas de l'image. Dans l'optique de réduire la largeur d'une image d'un pixel, on cherchera à déterminer la couture $s_v^\star$ d'énergie minimale, l'énergie d'une coupure étant définie comme la somme de l'énergie des pixels qui la composent :

$$s_v^\star = \arg\min_{s_v} E(s_v) = \arg\min \sum_{i=1}^{n} E(i, j_i)$$

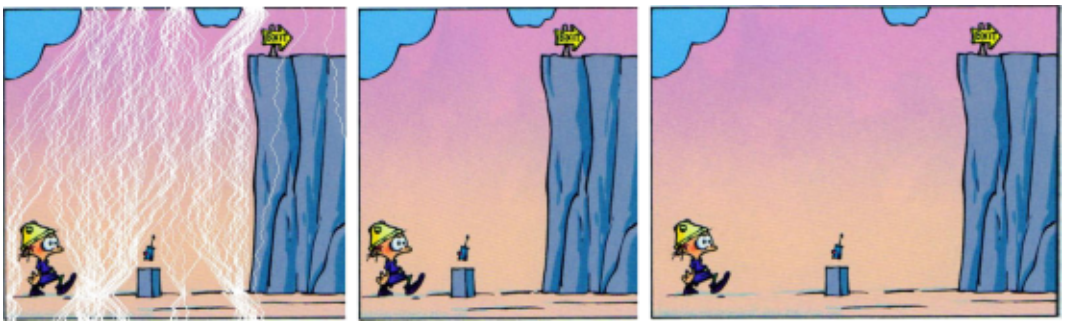

FIGURE 2 – À gauche, l'image de départ où les 100 coutures de coût minimal sont représentées en blanc. Au centre, la version réduite de l'image obtenue en supprimant ces 100 coutures. À droite, la version étendue obtenue en dupliquant ces coutures.

Pour déterminer la couture d'énergie minimale, vous devrez également utiliser la programmation dynamique. La fonction de coût à formuler de manière récursive sera dans ce cas $C(i, j)$, l'énergie de la couture d'énergie minimale s'arrêtant au pixel (i, j).

1.2.3 Réduction de la taille d'une image

Pour réduire la largeur de l'image de m à $m' < m$, on appliquera $m' - m$ fois les opérations suivantes en partant de l'image originale :

1. On recherche la couture verticale d'énergie minimale $s_v^\star$ dans l'image courante.
2. On crée une nouvelle image en enlevant de l'image courante les pixels de $s_v^\star$.

Etant donné qu'une couture verticale contient un pixel par ligne, chaque application de l'étape 2 diminue bien la largeur de l'image d'un pixel tout en maintenant sa hauteur inchangée. Notez qu'à chaque recalcul de la couture d'énergie minimale (étape 1), les valeurs d'énergie doivent être recalculées pour prendre en compte l'image courante (qui est donc partiellement réduite par rapport à l'image initiale). La figure 1 représente une image avec les 100 premières coutures obtenues par cette prcédure (à gauche) et la version réduite résultante (au centre).

1.2.4 Élargissement d'une image

La même technique peut être utilisée pour augmenter artificiellement la largeur d'une image. La procédure fonctionne de la manière suivante (pour étendre l'image de k pixels en largeur) :
— On détermine exactement comme dans le cas de la réduction les k coutures d'énergie minimale. Le résultat de cette opération devra être un marquage des pixels de l'image

originale qui aurait du être supprimés si on avait effectué une réduction plut^ot qu'un élargissement.
— On parcourt l'image de gauche à droite et de haut en bas. Chaque pixel non marqué à l'étape précédente est recopié tel quel dans l'image finale. Chaque pixel marqué est recopié également mais on ajoute un nouveau pixel à sa droite dans l'image finale dont la valeur (pour chaque couleur) est obtenue en moyennant la valeur du pixel marqué et du pixel à sa droite dans l'image originale. Si le pixel marqué se trouve sur le bord droit de l'image, la valeur ajoutée est simplement la valeur du pixel marqué. Cette procédure ne donne pas de bon résultats si le nombre de pixels à ajouter est trop important par rapport à la taille initiale de l'image. Il est préférable dans ce cas de répéter l'opération d'élargissement plusieurs fois avec un nombre moindre de pixels. Soit k' un paramètre fixé arbitrairement à 20% de la taille de l'image, on procédera comme suit pour étendre la taille de l'image de k pixels lorsque $k > k'$:
— On élargit l'image de k' pixels en utilisant la procédure ci-dessus
— On répète cette opération tant que l'image obtenue n'a pas atteint la taille cible

1.3 Génération de la bande dessinée

Les différentes étapes pour générer la bande dessinée sur base d'une liste de cases et d'une largeur de page seront les suivantes :
— Récupération des largeurs des cases et détermination de leurs positions optimales en fonc- tion de la largeur cible.
— Pour chaque rangée de la bande dessinée :
— Calcul de la réduction ou de l'élargissement à appliquer à chaque case. On répartira de manière équitable la réduction ou l'élargissement nécessaire sur chacune des cases de la rangée (aux arrondis près).
— Application de la transformation (réduction ou expansion) à chaque case de la rangée et placement des cases sur l'image finale.
Pour donner un aspect visuel plus agréable, on ajoutera une bordure blanche de taille fixée entre les cases et autour de l'image. La largeur de ce bord devra être pris en compte notamment lors du calcul du placement des cases. Les largeurs de cases soumises à l'algorithme de répartition seront simplement augmentées de la largeur du bord et la largeur de page sera diminuée de la largeur du bord (pour tenir compte du bord à gauche).

2 Directives d'implémentation

On vous demande d'implémenter l'algorithme de mise en page intelligent décrit ci-dessus. Vous devrez implémenter les fonctions suivantes.

`wrapImages`. Cette fonction calculera le placement optimal des cases. Elle prendra comme arguments un tableau d'images (les cases de la BD), un nombre de cases, une largeur de page et une taille de bordure. La sortie de cette fonction sera une table de

taille égale au nombre de cases et reprenant pour chacune la rangée à laquelle elle doit être placée.

`reduceImageWidth`. Cette fonction implémentera la réduction d'une image par la technique décrite dans la section 1.2.3. Elle prendra en argument une image (le format d'une image est décrit également dans le fichier `PNM.h`) et un nombre de pixels et elle renverra une nouvelle image obtenue en réduisant l'image initiale de k pixels.

`increaseImageWidth`. Cette fonction implémentera l'élargissement d'image suivant la technique décrite dans la section 1.2.4. Elle prendra en argument une image et un nombre k de pixels et renverra une nouvelle image élargie de k pixels.

`packComic`. Cette fonction sera la fonction principale de votre programme. Elle prendra en argument un tableau d'images (les cases de la BD), un nombre de cases, un tableau d'images (les cases de la BD), une largeur de page et une épaisseur de bord. Elle fournira en sortie une nouvelle image contenant la planche complète générée selon la procédure décrite dans la section 1.3.

Pour vous aider dans vos tests, nous vous fournissons également deux fichiers "main" :
— `mainSeamCarving.c` : qui prend en arguments sur la ligne de commande un nom de fichier image d'entrée, un nom de fichier d'image de sortie et un nombre de pixels (positif ou négatif). Ce programme génère dans un nouveau fichier l'image réduite (si le nombre de pixels est négatif) ou étendue (si le nombre de pixels est positif) à l'aide de vos fonctions `ImageWidthReduction` et `ImageWidthExpansion`.
— `mainComicPacking.c` : qui prend en arguments sur la ligne de commande une largeur d'image, une épaisseur de bord, le nom d'un fichier de sortie et une liste de fichiers correspondant au cases de la BD. Ce programme génère dans un nouveau fichier l'image de bande dessinée formée à partir des cases.
Ces fichiers font appel aux fonctions de chargement et d'écriture d'images fournies dans la libraire tierce `PNM.h/c`.

Afin de visualiser les images au format `.pnm`, vous pouvez utiliser :
— `pnmtopng` dans `Linux/macOS`. Exemple : `pnmtopng < 01.pnm > 01.png`.
— `ImageMagick` qui est un logiciel de ligne de commande qui marche dans tous les plateformes.

3 Rapport et évaluation

Pour chacun des deux algorithmes par programmation dynamique (répartition des cases et calcul de la couture d'énergie minimale) :

Partie 1 (`packComic`). (a) Montrez qu'une approche par recherche exhaustive serait de complexité exponentielle (respectivement par rapport au nombre de cases et par rap-

port à la hauteur de l'image).
(b) Proposez une approche gloutonne pour résoudre le problème (sans donner de pseudo-code), donnez sa complexité et montrez par un contre-exemple qu'elle ne fonctionnera pas dans le cas général. Discutez la pertinence pratique de cette solution.
(c) Donnez la formulation récursive complète de la fonction de coût (respectivement $c(n)$ et $C(i, j)$), en précisant bien les cas de base.
(d) Représentez le graphe des appels récursifs pour un problème de petite taille.
(e) Décrivez en pseudo-code un algorithme efficace de calcul des valeurs de cette fonction de coût (par l'approche ascendante ou descendante).
(f) Donnez la complexité en temps et en espace de cet algorithme.

Partie 2 (réduction et élargissement). (a) Implémentez une fonction `showEnergies` qui produit une image montrant les énergies de chaque pixel sur l'échelle en noir et blanc.
(b) Implémentez une fonction `showSeams` qui produit une image avec les coutures identifiées avec une couleur blanche (comme dans Figure 2).
Pour chacune des deux fonctions `ImageWidthReduction` et `ImageWidthExpansion` :
(c) Décrivez vos choix d'implémentation.
(d) Précisez leurs complexités en temps et en espace en fonction de la taille de l'image, $n \times m$, et du nombre k de pixels à supprimer ou à enlever.

Références

[AS07] S. Avidan and A. Shamir. Seam carving for content-aware image resizing. In ACM Trans. Graphics, volume 26, page 10, 2007.