ministère éducation nationale

Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE

éduscol

Ressources pour le lycée général et technologique

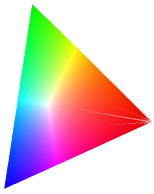
Ressources pour le cycle terminal général et technologique

Informatique et Sciences du Numérique

Algorithmes pour le traitement d'images - 1

Juin 2012

Présentation / Algorithmes pour le traitement d'images - 1



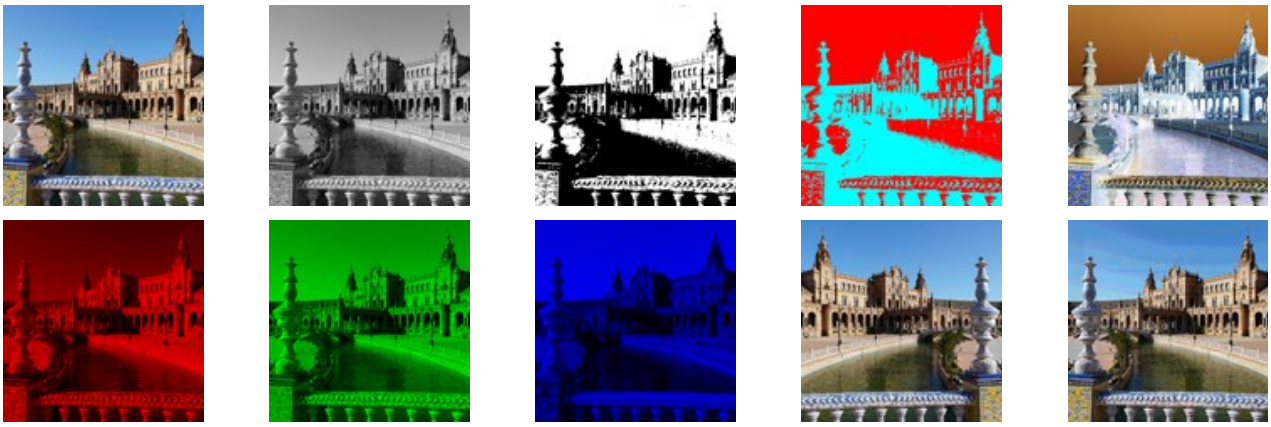
1 / Thème abordé

1.1 Problématique, situation d'accroche

Il est possible, en utilisant l'image pour support, d'étudier et de mettre en œuvre les structures algorithmiques de base ainsi que les opérations booléennes usuelles, et d'obtenir des résultats rendant cette étude à la fois ludique et motivante.

Ce travail peut être proposé dans le cadre d'activités ou de mini-projets.

Il s'agit, entre autres, de comprendre comment on peut passer d'une image couleur à une image en niveaux de gris ou bien en noir et blanc, comment on peut en transformer les couleurs, et même comment on peut y dissimuler une information. Se pose alors la question de pouvoir vérifier qu'une image n'a pas subi de transformations imperceptibles à l'œil nu.



À l'aide d'algorithmes très simples, il est possible à partir de l'image en haut à gauche, d'obtenir toutes les autres. Toutes ces transformations peuvent bien-sûr être effectuées à l'aide d'un logiciel tel que GIMP, mais il s'agit ici de comprendre ce que fait ce type de logiciel sur chaque pixel de l'image initiale.

Des langages de programmation tels que Python, avec notamment la bibliothèque PIL, permettent d'accéder à chaque pixel d'une image matricielle (ou bitmap), et d'en modifier ses caractéristiques (composantes rouge, verte, et bleue) ; les algorithmes évoqués peuvent alors être réalisés en quelques lignes de programme.

D'autres questions surgissent ensuite : que voit-on lorsqu'on observe la quatrième image à travers un filtre cyan ? Ou bien à travers un filtre rouge ? Pourquoi ?

Le dernier exemple a ceci de particulier, qu'une autre image y est dissimulée : c'est un exemple de stéganographie. L'idée est très ancienne, mais elle a pu évoluer grâce à la numérisation : dans le tableau d'Hans Holbein le Jeune « Les Ambassadeurs »[1], l'objet caché est facilement repérable en regardant attentivement le tableau, et en orientant son regard convenablement, ou bien en en déformant une image[2], alors qu'avec la stéganographie, c'est chaque pixel qui cache une partie de l'image dissimulée dans celle qui est visible à l'œil nu.

1.2 Frontières de l'étude et prolongements possibles

L'étude se limite à des algorithmes simples, et aux programmes courts correspondants, étant entendu que l'élève peut être amené à n'en réaliser qu'une partie, voire à simplement comprendre ce qu'un algorithme donné est censé produire. Néanmoins, le niveau envisagé doit permettre d'en réaliser la totalité.

2 / Objectifs pédagogiques

2.1 Disciplines impliquées

De nombreuses disciplines sont concernées par l'image. Des situations d'apprentissage en interdisciplinarité sont

1 Ce tableau se trouve à la National Gallery de Londres, voir aussi : http://fr.wikipedia.org/wiki/Les_Ambassadeurs

2 Cela s'appelle une *anamorphose*.

alors tout à fait envisageables. Des connexions sont possibles en particulier avec le programme 2012 de sciences physiques en terminale S (partie intitulée « transmettre et stocker de l'information ») tout comme avec celui de 1ère S (synthèse additive, synthèse soustractive, effet d'un filtre coloré sur une lumière incidente).

2.2 Prérequis

Aucun prérequis n'est véritablement indispensable, sauf pour la stéganographie, qui repose sur la représentation binaire de l'information et sur les opérations booléennes.

2.3 Éléments du programme

Contenus

- Représentation de l'information : numérisation.
- Algorithmes simples, programmation.

Compétences et capacités

Décrire et expliquer une situation, un système ou un programme :

- Comprendre un algorithme et expliquer ce qu'il fait.
- Modifier un algorithme existant pour obtenir un résultat différent.

Concevoir et réaliser une solution informatique en réponse à un problème :

- Concevoir, programmer un algorithme.
- Modifier format, taille, contraste, ou luminance d'images numériques.
- Filtrer et détecter des informations spécifiques.

Collaborer efficacement au sein d'une équipe dans le cadre d'un projet :

- Concevoir des programmes en autonomie.
- Gérer les étapes de l'avancement du projet en dialogue et en interaction avec le professeur.

3 / Modalités de mise en œuvre

3.1 Durée prévue pour la partie se déroulant en classe

6 à 8 heures suivies d'un mini-projet.

3.2 Type de l'animation

Activités en classe entière (passage de la couleur aux niveaux de gris, filtres ...) puis mini-projets (stéganographie, histogramme d'une image et applications, ...).

3.3 Projet

Les activités initiales doivent donner suffisamment d'éléments aux élèves pour leur permettre ensuite de mener à bien un mini-projet pouvant les inspirer dans l'optique d'un projet plus vaste ou plus approfondi.

3.4 Recherches documentaires

Une recherche peut être envisagée comme point de départ des mini-projets, qu'il s'agisse de la stéganographie, des histogrammes d'images et de leurs applications, ou encore des images en pseudo-couleurs.

3.5 Production des élèves

Algorithmes et programmes à concevoir ou à compléter dans le cadre d'activités en classe, puis mini-projets à rendre et à présenter oralement.

4 / Outils

Un langage de programmation simple, permettant de travailler aisément sur des images numériques ; Python 2.7 associé à la bibliothèque de traitement d'images PIL (Python Imaging Library) en est un bon exemple. Pour le reste, le logiciel de traitement d'images GIMP rendra les plus grands services.

5 / Auteur

François Passebon, professeur de Sciences Physiques, académie de Nantes

Algorithmes pour le traitement d'images - 1

1 / Activités autour de l'image

Plusieurs des activités décrites ci-dessous sont détaillées en annexe 1.

Le point de départ est la structure d'une image matricielle (ou bitmap) : lorsqu'on ouvre un fichier au format PNG dans GIMP par exemple, et qu'on réalise un agrandissement suffisamment important, l'image se présente sous forme d'une mosaïque dont chaque élément est un carré de couleur uniforme : le pixel.

Avec l'outil pipette, on accède à des informations sur la couleur : on voit qu'elle est définie dans ce cas à l'aide de trois composantes (rouge, verte, et bleue, notées RVB par la suite), et que chacune d'elle peut prendre une valeur entière comprise entre 0 et 255.

Le lien avec la synthèse additive des couleurs est immédiat : on peut utiliser des outils de simulation pour réactiver le travail fait sur la couleur en physique en 1ère S; celui-ci par exemple :

http://dev.physicslab.org/asp/applets/additivecolors/default.asp

(agir sur les trois curseurs), ou bien celui-là, où 100 % correspond à la valeur 255 :

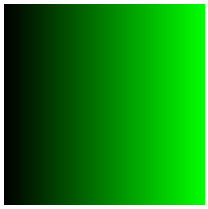
http://www.ostralo.net/3_animations/swf/couleurs_ecran.swf

Que se passe-t-il lorsqu'on transforme l'image couleur en une image en niveaux de gris ? Dans GIMP, avec l'outil pipette, on s'aperçoit que chaque pixel est différent, mais qu'un pixel donné a toujours ses composantes RVB identiques. Le noir correspond à RVB = 0,0,0 et le blanc à RVB = 255,255,255; les triplets intermédiaires sont les niveaux de gris.

À partir de là, pour envisager de modifier une image par programmation, il faut pouvoir balayer cette image pixel par pixel. Deux boucles imbriquées permettent d'y parvenir. Un exercice peut être proposé pour comprendre ce type de structure. Dans le cas d'une image 512 x 512 pixels par exemple, cela donne :

```
pour i = 0 à 511                # i représente la ligne à laquelle appartient le pixel
     pour j = 0 à 511           # j représente la colonne
          traitement du pixel (i,j)
     fin pour
fin pour
```

On peut être amené à permuter les deux boucles si nécessaire. À ce propos, un premier exercice peut consister à représenter les nuances de rouge de 0 (noir) à 255, et de faire de même pour le vert et le bleu, ainsi que pour les niveaux de gris. Quelle largeur doit-on alors donner à l'image pour réaliser cela le plus simplement possible ?

Pour l'obtention d'une image en niveaux de gris à partir de l'image en couleurs, le traitement envisagé peut s'inspirer de ce que propose le logiciel GIMP.

Le menu Couleurs > Désaturer … offre trois options :

« Clarté », où le niveau de gris de chaque pixel est la moyenne entre le minimum et le maximum des trois composantes RVB. Si par exemple (R,V,B) = (122,200,147), cette moyenne vaut (122+200)/2 = 161, et le résultat est (R,V,B) = (161,161,161).

« Luminosité », où le niveau de gris correspond à R = 0,21*R+0,71*V+0,07*B, et V = R, B = R.

« Moyenne », où R = (R+V+B)/3 et V = R , B = R.

On peut aussi tester R = 0,299*R+0,587*V+0,114*B , V = R, B = R, utilisé dans les signaux vidéos PAL.

Comment passer ensuite à une image en noir et blanc ? (Composantes RVB identiques, mais ne pouvant prendre que les valeurs 0,0,0 (noir) ou 255,255,255 (blanc))

Il faut se fixer un seuil, par exemple 128 : en-dessous de 128, c'est 0 par exemple, et au-dessus, c'est 255.

```
si R < seuil
     R=0
sinon
     R=255
fin si
V=R ; B=R
```

On peut obtenir facilement à partir de là l'image en cyan et rouge : il suffit de convertir les pixels noirs en cyan et les blancs en rouge. On réinvestit les connaissances sur les couleurs : Cyan = Vert + Bleu, donc RVB = 0,255,255 pour le cyan, et ce codage laisse alors bien apparaître que cette couleur est complémentaire du rouge !

Si l'on revient à l'image couleur de départ, comment la verrait-on à travers un filtre rouge ? Il suffit de mettre à zéro les composantes V et B de chaque pixel. Même principe pour un filtre vert ou bleu. C'est l'occasion de faire le lien avec les images satellitaires ou plus généralement les images en fausses couleurs (les astrophysiciens utilisent tout le spectre électromagnétique pour obtenir des images).

Comment obtenir une image en négatif ? Il suffit de faire : R = 255-R ; V = 255-V ; B = 255-B (ce qui revient à prendre le complément à un des valeurs binaires R, V, et B).

Comment opérer un retournement horizontal de l'image ? L'ordre des lignes est inchangé, mais le pixel de la colonne j passe en colonne 511-j pour une image de largeur 512 pixels (511 car j varie de 0 à 511).

On peut à cette occasion faire le lien avec l'étude des fonctions en Mathématiques : il s'agit de comparer les graphes de $x \to f(x)$ et de $x \to f(a-x)$.

Enfin, le retournement simultané en horizontal et en vertical ne devrait plus poser de problèmes ...

On peut aussi demander aux élèves de superposer deux images : une qui puisse être vue à travers un filtre rouge, et l'autre à travers un filtre cyan. À partir de là, si les images sont deux prises de vue de la même scène, mais légèrement décalées pour reproduire la vision gauche et la vision droite, on obtient un *anaglyphe*. Dans les deux cas, il faut prévoir des lunettes « 3D » (filtre rouge sur un œil et cyan sur l'autre), qu'on trouve très facilement. Si l'on dispose d'autres filtres de couleurs complémentaires (vert et magenta, ou bleu et jaune), il peut être intéressant de créer différents anaglyphes à partir des deux mêmes prises de vues et de s'interroger sur la différence dans la perception visuelle du résultat.

L'annexe 1 détaille aussi la gestion des images transparentes. L'utilisation de la transparence peut être utile dans l'étude de la rotation d'une image, si l'on souhaite éviter un fond coloré uniforme dans l'image destination.

D'autres activités peuvent être proposées : application d'un masque à une image, correction d'une image à l'aide d'une courbe tonale, ... (voir annexe 1).

Les activités proposées ici n'ont pas vocation à être envisagées en totalité pour tous les élèves quoi qu'il arrive. Certaines notions peuvent nécessiter pour certains d'entre eux des exercices préparatoires ou annexes pour comprendre leur principe et leur intérêt. Lors de l'implémentation des deux boucles "pour" imbriquées, le contenu peut dans un premier temps se limiter à afficher l'état du compteur d'une boucle ou d'une autre, avec des valeurs maximales volontairement réduites. S'il est possible à chaque fois d'arriver rapidement à un résultat intéressant, chaque élève doit avoir compris les structures algorithmiques à l'œuvre et leur implémentation, le but étant de les réinvestir ensuite dans le cadre de mini-projets.
Les aspects sociétaux et juridiques peuvent compléter l'étude : utilisation de filigranes pour attester des droits sur une image; peut-on remettre en ligne une image que l'on a modifiée ? Etc.
*La question des **droits** et **licences** est suffisamment importante pour envisager une activité à ce propos. Les élèves créent par exemple des images (retouches, filtrages, anaglyphes, transformations diverses, à partir de photos libres de droits et autant que possible de leurs propres photos). Ils déposent ces images sur le site de partage Flickr (http://www.flickr.com) après avoir créé chacun un compte. Ils décident individuellement de la licence qu'ils vont appliquer à leurs propres créations, allant de la paternité simple (CC BY) à la paternité sans utilisation commerciale ni modification (CC BY-NC-ND) et débattent ensuite du bien-fondé de ces choix. Lors de la création de leur compte, ils sont amenés à s'interroger sur certaines options et leur signification : masquage ou non des données EXIF associées par défaut aux images obtenues à l'aide d'un appareil photo numérique, par exemple.*

Un aspect important concernant l'implémentation des algorithmes : la division en langage Python

Certains aspects spécifiques aux langages de programmation doivent être détaillés pour permettre ensuite une implémentation correcte des différents algorithmes. C'est le cas de l'opération de division en langage Python.
L'interface Python shell est suffisante pour aborder cet aspect à travers de petits exercices très simples.
3/7 donne pour résultat 0 : lorsqu'on divise deux entiers, l'opérateur / correspond donc à la division entière (quotient).
3.0/7 ou 3/7.0, ou encore 3.0/7.0 donnent par contre 0.42857142857142855. L'opérateur / réalise cette fois la division attendue car le dividende et/ou le diviseur ne sont plus considérés comme des entiers.
Pour obtenir la division entière, on peut utiliser explicitement l'opérateur de division entière, le double slash //. Par exemple, 3.0//7.0 donne 0.0 cette fois. Pour avoir un résultat de type entier, il faut ensuite prendre la partie entière à l'aide de int(...), qui donne 0 (au lieu de 0.0).
Les composantes RVB des pixels ayant des valeurs entières comprises entre 0 et 255, si leur traitement fournit des valeurs non entières, il faut penser à convertir le résultat en entier à l'aide de int(...).
int(3.8) donne 3 pour résultat, c'est bien la partie entière, tandis que round(3.8) donne 4.0 c'est à dire l'arrondi. En langage Python, les types entier, réel, ... étant implicites, il faut s'attendre à des erreurs avant que cela soit bien acquis.
Dernière remarque : si l'on veut dans un programme Python 2.7 qu'une division de deux entiers donne "le bon résultat", par exemple que 3/7 donne 0.42857142857142855 , on peut insérer en première ligne de ce programme :
```
from __future__ import division
```
avec deux caractères "underscore" à gauche du f et deux à droite du e de future.
Sachant que int(...) peut être utilisé pour obtenir un résultat entier en cours de programme si nécessaire.

2 / Mini-projets

Trois exemples de mini-projets sont décrits ici, assez sommairement. L'annexe 2 donne des détails supplémentaires.

Une étude succincte de la stéganographie tout d'abord. On peut obtenir un résultat intéressant même avec un algorithme simple tel que celui évoqué en annexe 2, et demander aux élèves de le programmer. Les nouvelles notions nécessaires (rudiments sur le binaire notamment) peuvent être précisées dans un petit cahier des charges, à moins qu'elles aient été étudiées auparavant. Pour mener à bien ce mini-projet, il y a finalement peu de lignes à insérer dans les deux boucles évoquées précédemment, mais il faut les trouver …

Le décodage d'une image stéganographiée avec la méthode proposée en annexe peut aussi être envisagé, il n'est guère plus compliqué.

La création de l'histogramme d'une image et ses différentes exploitations possibles peuvent donner lieu également à plusieurs mini-projets différents. Des pistes assez précises sont données en annexe 2.

La création d'images en pseudo-couleurs peut constituer aussi un projet intéressant; voir annexe 2.

Ces deux dernières propositions amèneront sans doute les élèves à travailler avec des fichiers. Les lignes de code permettant de créer un fichier et d'écrire dans celui-ci devront être fournies aux élèves, ou bien un lien vers la documentation Python correspondante.

Quant aux productions des élèves, celles-ci peuvent tout à fait s'intégrer dans l'enseignement de l'histoire des arts au lycée (art du visuel). Une exposition peut pourquoi-pas être envisagée, où les élèves expliqueraient à d'autres, non scientifiques en particulier, les principes physiques utilisés. L'ENT peut quant à lui être mis à profit pour le stockage et l'échange de documents (cahier des charges des mini-projets, production des élèves et suivi de leur remise dans les délais).

3 / Références

3.1 Sitographie

- Ressources Python : http://pythonfacile.free.fr/python/ressources.html
- Opérateurs de base en Python (en anglais) : www.tutorialspoint.com/cgi-bin/printversion.cgi?tutorial=python&file=python_basic_operators.htm
- Python Imaging Library: http://www.pythonware.com/products/pil
- Tutoriel PIL : http://www.pythonware.com/library/pil/handbook/image.htm
- Fundamentals of Image Processing – document très complet, une centaine de pages, se trouve en cherchant le titre sur : http://studenten.tudelft.nl/en
  L'adresse actuelle est : http://www.tnw.tudelft.nl/fileadmin/Faculteit/TNW/Over_de_faculteit/Afdelingen/Imaging_Science_and_Technology/Research/Research_Groups/Quantitative_Imaging/Education/doc/FIP2_3.pdf
- Autre site sur les formats d'image : http://www.martinreddy.net/gfx
- Traitement d'images : http://www.efg2.com/Lab/Library/ImageProcessing
- Contrats Creative Commons : http://creativecommons.org/licenses
- Logiciel de traitement d'images GIMP : http://gimp.org
- Documentation française de GIMP : http://docs.gimp.org/2.6/fr/

3.2 Bibliographie

DUPRÉ X. *Programmation avec le langage PYTHON*. Éditions Ellipses, 2011.

LAMBERT K., OSBORNE M. *Fundamentals of Python : first programs.* Cengage Learning, 2011.

Les exemples ayant trait à l'image présentés dans cet ouvrage n'utilisent pas PIL, mais une bibliothèque créée par l'auteur. L'ouvrage est néanmoins très pédagogique quant à la présentation d'ensemble de Python.

FOLEY & VAN DAM. *Computer Graphics : Principles and Practice.* Addison-Wesley, nouvelle édition, 1995.

De BERG et.al. *Computational Geometry : Algorithms & Applications.* Springer, 2008.

A. MARION. *Introduction aux techniques de traitement d'images.* Eyrolles, Paris, 1987. Nouvelle édition 1998.

M. COSTER & J.L. CHERMANT. *Précis d'Analyse d'Images.* Ed. du CNRS, Paris.

M. LAUG. *Traitement Optique du Signal et des Images.* Ed. Cépaduès, Toulouse.

A.K. JAIN. *Fundamentals of Digital Image Processing.* Prentice Hall, 1989.

J.M. CHASSERY & A. MONTANVERT. *Géométrie Discrète en Analyse d'Images.* Hermes, Paris, 1991.

R. HORAUD & O. MONGA. *Vision par Ordinateur : Outils Fondamentaux.* Hermes, Paris, 1993.

M. KUNT, G. GRANLUND & M. KOCHER. *Traitement numérique des Images.* P.P.U.R, Lausanne, 1993.

J.P. COCQUEREZ & S. PHILIPP. *Analyse d'Images : Filtrage et Segmentation.* Masson Ed., Paris, 1995.

A. BOVIK. *Handbook of Image and Video Processing.* Academic Press, San Diego, 2000.

DELAHAYE J.-P. *Information noyée, information cachée.* Pour la Science n° 229, novembre 1996.

LECARNE O. & DELVARE K. *GIMP, manuel de référence pour l'édition d'images,* Pearson, Paris, 2010.

3.3 Crédits photographiques

Les photographies incluses dans ce document sont propriété de l'auteur.