

L3 Physique

UE PAX6PHAK « Analyse et Traitement de Signal »

Travaux Pratiques

Traitement d'images

Objectifs

- *Découvrir en pratique les procédés élémentaires de traitement des images numériques.*
- *Compléter les connaissances en traitement numérique du signal sur l'exemple de signaux bidimensionnels.*

Erik Kerstel

PLAN DU DOCUMENT¹

1 TRAVAIL DEMANDE	2
2 INTRODUCTION	3
2.1 LabView en quelques mots ...	3
2.2 Lecture d'un fichier jpeg et affichage de l'image	4
3 COULEURS, QUANTIFICATION, ECHANTILLONNAGE	5
3.1 Codage : « couleur » ou « noir et blanc »	5
3.2 Quantification	5
3.3 Sous-échantillonnage (à ne faire qu'à la fin si le temps le permet)	6
4 INTERVENTION SUR LE SPECTRE D'UNE IMAGE	6
4.1 Spectre d'une image	6
4.2 Négatif	6
4.3 Luminosité (note : à faire soit 4.3, soit 4.4)	6
4.4 Contraste (note : à faire soit 4.3, soit 4.4)	7
5 FILTRAGE D'UNE IMAGE	7
5.1 Filtrage par transformation de Fourier	7
5.2 Filtrage matriciel (filtre RIF)	7
5.3 Passe-bas (bruit, reconstruction, antialiasing ...)	7
5.4 Passe-haut (détection de contours, renforcement ...)	8
6 INTERCORRELATION (FACULTATIF)	8
6.1 Collage d'images par intercorrélation.	8

1 TRAVAIL DEMANDE

Ces deux séances de TP consacrées aux images ne seront pas de trop pour que vous puissiez vous familiariser avec le logiciel (LabView) et les concepts de traitement d'images. L'objectif est que vous réalisiez à l'aide de LabView des programmes remplissant la demande et que vous analysiez les résultats obtenus en s'appuyant sur des exemples (images « modèles » fournies ou autres).

Un compte rendu sous forme électronique vous sera demandé et il est vivement conseillé de le rédiger au fur et à mesure des séances. Ce compte rendu sera demandé à la fin de la quatrième séance.

Pour une sécurité maximale il vous est recommandé de ne pas laisser vos fichiers sur le « bureau » mais de les placer sur votre emplacement personnel (serveur « Sarrado »). Par ailleurs il peut être bon d'emmener vos fichiers sur une clef USB à la fin de la séance.

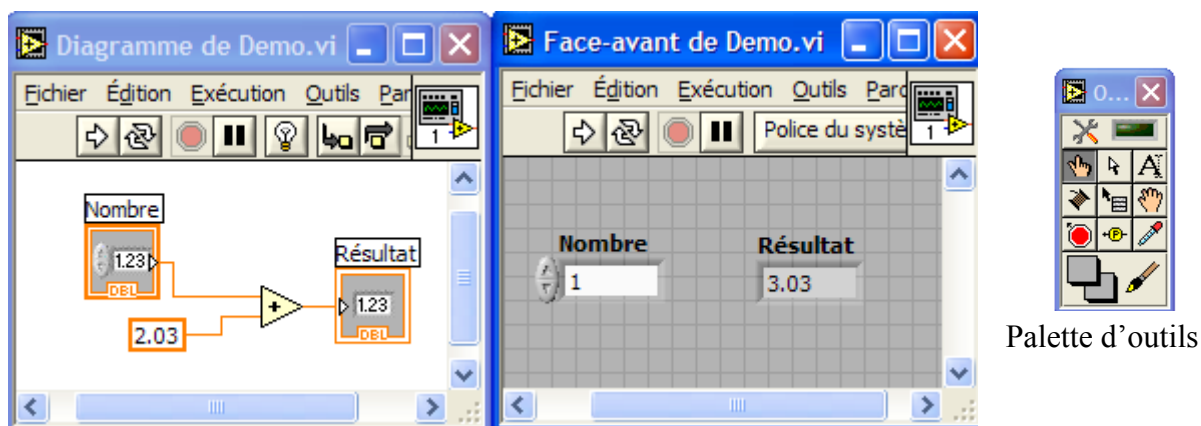
¹ Ce TP est basé sur le TP développé initialement par Hadrien Mayaffre.


2 INTRODUCTION

2.1 LABVIEW EN QUELQUES MOTS ...

LabView est un logiciel développé initialement pour aider à l'acquisition de données en facilitant la programmation de la communication avec des appareils externes (carte d'acquisition, multimètre, générateurs etc...). Au cours de son développement le logiciel s'est beaucoup enrichi en outils de traitement des données et c'est cet aspect que nous allons utiliser ici.

La spécificité de LabView est qu'il est basé sur une programmation diagrammatique. Un programme Labview est appelé « instrument virtuel » ou « **vi** ». Un « **vi** » est constitué de deux fenêtres, l'une appelée « **diagramme du vi** » dans laquelle se trouve le diagramme représentant le programme, l'autre appelée « **face avant du vi** » dans laquelle se trouvent tous les paramètres accessibles à l'utilisateur du programme (commandes ou indicateurs). On passe d'une fenêtre à l'autre par la commande **CTRL E**.



Dans l'exemple ci-dessus « Nombre » est appelé « **commande** » car l'utilisateur peut agir dessus avant de lancer l'exécution (une commande peut prendre toutes sortes d'aspects : cadres de saisie, boutons, curseurs ...). « Résultat » est un « **indicateur** » qui permet d'afficher un résultat sur la face avant (un indicateur peut prendre toute sortes d'aspects : cadres, graphe, images ...). « 2.03 » est une « **constante** » qui n'est pas accessible à l'utilisateur (seulement au programmeur). Le symbole  est un « **opérateur** » ou « **sous vi** ». Vous pourrez créer vos propres opérateurs (sous-vi) à partir de vos diagrammes.

Tout objet est pourvu de **connecteurs**. Ces connecteurs sont directionnels c'est-à-dire qu'ils permettent soit d'accéder à une information que fournit l'objet (connecteurs en lecture) soit de fournir une information à l'objet (connecteurs en écriture). Par exemple l'opérateur « + » ci-dessus a deux connecteurs en écriture (entrées) et un connecteur en lecture (sortie).

Les connecteurs obéissent à des règles élémentaires et logiques. Par exemple un connecteur en écriture ne peut être connecté simultanément à deux connecteurs en lecture (conflit). Les types des données correspondant à deux connecteurs connectés doivent être identiques.

Certains connecteurs doivent impérativement être connectés pour que le vi soit exécutable. D'autres par contre peuvent être laissés libres.

L'agencement du diagramme et de la face avant se fait avec l'aide de la « **palette d'outils** » dans laquelle on trouve le nécessaire pour connecter, sélectionner, modifier du texte, modifier une commande etc...

Les commandes et indicateurs pourront être placés sur la face avant à partir de la « **palette de commandes** » (clic droit sur face avant ou *afficher palette de commandes*). Les constantes et



opérateurs pourront être placés sur le diagramme à partir de la « **palette de fonctions** » (clic droit sur diagramme ou *afficher palette de fonctions*).

L'exécution d'un vi se fera par l'action de la flèche blanche dans le bandeau de la fenêtre. Si cette flèche n'est pas blanche c'est que le vi n'est pas exécutable soit parce que le diagramme comporte une erreur (flèche brisée) soit parce que le vi est déjà en cours d'exécution (flèche noire).

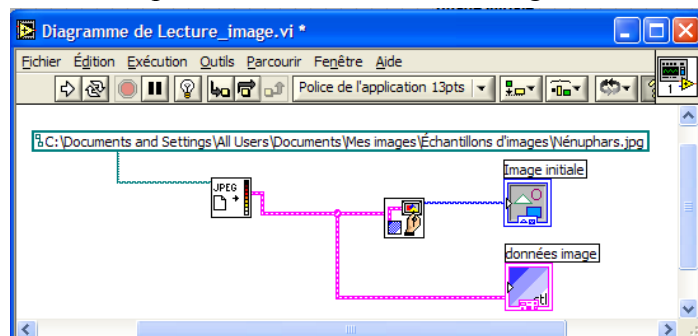
Lorsque vous passez la souris sur la face avant ou le diagramme vous pouvez disposer d'une **aide contextuelle** (CTRL H pour l'activer) qui s'affiche en fonction de l'objet pointé : n'hésitez pas à en abuser. Il serait également utile d'avoir à disposition les fichiers « LV_Quick_Guide.pdf » et « LV_Getting_Started.pdf ».

2.2 LECTURE D'UN FICHIER JPEG ET AFFICHAGE DE L'IMAGE

En guise de première prise en main nous allons écrire un programme permettant de lire un fichier de format jpeg et d'en extraire les informations et de le visualiser sous la forme d'une image dans LabView. Ce programme constituera un préalable au traitement d'image.

Réalisez le programme décrit ci-contre. Pour ce faire vous trouverez les deux « vi's » ( « lire un fichier jpeg » et  « tracer une table de pixels aplatie ») dans la librairie dédiée aux images de la palette de fonctions.

A noter que la fonction utilisée ici pour la visualisation de l'Image initiale est disponible seulement sur la face avant du vi principale (« Lecture_image.vi ») sur la palette « classic / classic graphs / image ».



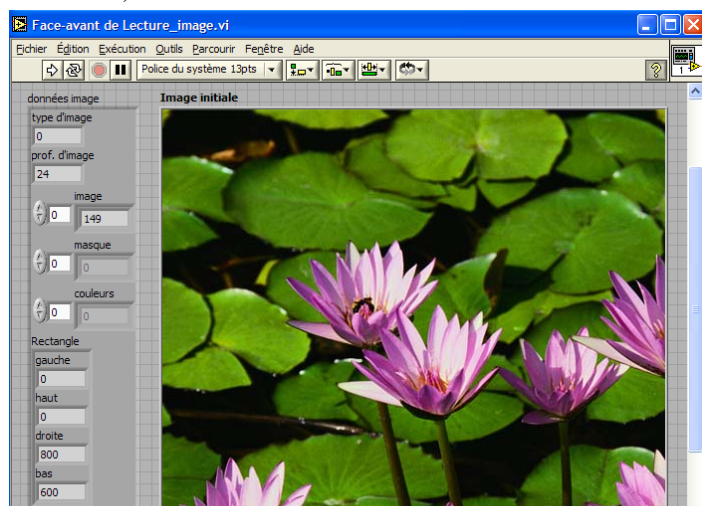
Un moyen très simple pour créer les constantes, commandes et indicateurs ad hoc consiste à se placer avec l'outil de connexion (bobine de fil) sur le connecteur et d'un clic droit sélectionner *créer constante (indicateur ou commande)*.

Une fois le diagramme fait, la face avant doit se présenter comme ci-dessous après exécution. L'indicateur de « données image » vous permet de visualiser toutes les informations relatives à l'image.

Ces données sont rassemblées sous la forme d'un « **cluster** » (assemblage de différents types de données qui permet de les manipuler ensemble).

Ce cluster contient :

- « type d'image » : entier inutilisé
- « profondeur » : entier précisant sur combien de bits sont codées les couleurs
- « image » : tableau 1D d'entiers définissant l'image
- « masque » : inutilisé ici



- « couleurs » tableau 1D définissant la table de couleur (système RougeVertBleu si tableau vide)
- « rectangle » cluster contenant la taille de l'image.

Il est possible et nécessaire d'accéder à ces différentes données par dé-assemblage du cluster. Vous trouverez dans la palette de fonction une librairie de vi dédiée à la manipulation des clusters.

Pour terminer ce petit vi préparatoire il peut être intéressant de voir comment faire pour que LabView ouvre une boîte de dialogue permettant de sélectionner un fichier image sur le disque (palette de fonction *E/S sur fichiers*). On étudiera de même la possibilité de sauvegarder une image en jpeg ou bmp.

3 COULEURS, QUANTIFICATION, ECHANTILLONNAGE

3.1 CODAGE : « COULEUR » OU « NOIR ET BLANC »

Lorsque vous ouvrez une image couleur « standard », la profondeur d'image est de 24. Cela signifie que les couleurs sont codées sur 24 bits. L'image est en fait sauvée dans un tableau unidimensionnel dans lequel un pixel va être défini par la succession de 3 octets ($3 \times 8 \text{ bits} = 24$). Chaque octet représente la luminosité respective des trois couleurs primaires RVB (Rouge Vert Bleu). Par concaténation de ces 3 octets on obtient le code couleur RVB. Pour chaque pixel on a donc les correspondances 0-0-0 (noir 0) 255-255-255 (blanc 16777200), 128-128-128 (gris moyen 8421500), 0-255-0 (vert lumineux 65280), 0-0-255 (bleu lumineux 255) ... Lorsque la profondeur n'est pas de 24 il est nécessaire de connaître la table de correspondance entre les valeurs de codage et le code couleur RVB : c'est le rôle de la table de couleur. Par exemple pour une profondeur de 4 cette table contient 16 valeurs correspondant aux 16 couleurs utilisées.

A l'aide d'un programme, explorez ces aspects en transformant une image couleur en noir et blanc (attention il y a plusieurs possibilités pour passer en noir et blanc) et en essayant de réduire sa taille de stockage (passage de 24 à 8 bits) (le format bmp sera peut être plus clair à interpréter).

3.2 QUANTIFICATION

Pour les images, la quantification peut se voir comme le nombre de niveaux de couleurs utilisés.

A partir d'une image de 256 (8 bits) niveaux de gris explorer l'évolution de cette image en réduisant la quantification à 4 bits (et 2 bits, puis 1 bits, si vous avez le temps) (si le temps le permettrait, on pourra aussi s'amuser avec une image en couleur sachant que les solutions de quantification peuvent être multiples avec des effets qui peuvent devenir psychédéliques !). Pour ces opérations on ne perdra pas de temps à comprendre comment Labview modifié le format des données pour des profondeurs de 4, 2 ou 1. On restera donc en profondeur 8 (ou 24) et on se contentera de réduire la quantification en limitant les possibilités de niveaux sur un nombre de 8 bits.

3.3 SOUS-ECHANTILLONNAGE (A ne faire qu'à la fin si le temps le permet)

Pour une image, l'échantillonnage correspond à la « pixellisation » de celle-ci. Cette opération a lieu soit lors de la prise de vue dans un appareil numérique (capteur) soit lors de la numérisation d'une photo ou diapo (scanner).

Les images dont vous disposez sont déjà numérisées et nous allons étudier les effets de l'échantillonnage en réduisant la fréquence spatiale de celui-ci. L'idée est donc d'agrandir artificiellement la taille d'un pixel.

Faire un programme permettant de faire ce « sous échantillonnage » (la taille des nouveaux pixels pourra être ajustée depuis la face avant). Pour cela on remplacera le code couleur d'un bloc de $n \times n$ pixels par la couleur du premier pixel du bloc (on peut aussi prendre la couleur moyenne du bloc) puis on passera au bloc suivant :

Par exemple ci-dessous on sous échantillonne un tableau par blocs de 3.

12	1	100	23	13	12	31	231	21	..
3	21	31	2	190	45	34	11	77	..
255	8	0	231	11	32	98	21	321	..
..

12	12	12	23	23	23	31	31	31	..
12	12	12	23	23	23	31	31	31	..
12	12	12	23	23	23	31	31	31	..
..

Il sera donc nécessaire pour faire cela de transformer l'image en un tableau bidimensionnel.

On testera ce procédé en particulier sur une image présentant des périodicités marquées (bandes noires par exemple : voir les images de test *bandes_sinus*, *bandes_rect*, *damier_sinus*, *damier_rect*).

Les défauts inhérents à ce processus seront repris plus tard (corrections par filtrage pour reconstruction et anti-aliasing).

4 INTERVENTION SUR LE SPECTRE D'UNE IMAGE

4.1 SPECTRE D'UNE IMAGE

On entend par spectre d'une image l'histogramme des niveaux de couleur c'est-à-dire la représentation du nombre de pixels ayant un niveau de couleur en fonction de celui-ci. Pour une image en couleur ce spectre peut être fait sur chaque couleur fondamentale (r-v-b) et/ou sur la luminosité globale (niveau de gris équivalent : par exemple $(r+v+b)/3$).

Ecrire un programme permettant de visualiser le(s) spectre(s) de l'image.

Il est possible de modifier le spectre d'une image en appliquant une transformation à la table des couleurs.

4.2 NEGATIF

En inversant la table de couleur, fabriquez le négatif d'une image.

4.3 LUMINOSITE (note : à faire soit 4.3, soit 4.4)

Il est possible d'assombrir ou d'éclaircir une image en utilisant une transformation qui « tasse » les niveaux vers le noir ou vers le blanc.

Faites l'essai avec une loi polynomiale du type $256 \left(\frac{x}{256} \right)^\alpha$ le signe de $(\alpha - 1)$ définissant s'il

on assombrir ou éclaircit.

On visualisera la forme de la transformation ainsi que l'effet sur l'image et le spectre. Que peut-il se passer au niveau d'une image en couleurs ?

4.4 CONTRASTE (note : à faire soit 4.3, soit 4.4)

Il est possible de modifier le contraste d'une image en modifiant l'étalement du spectre. Un spectre plus regroupé sur sa valeur moyenne correspond à une image moins contrastée.

Faites l'essai avec une loi $128 + 128 \left(\frac{x-128}{128} \right)^\alpha$ pour $x > 128$ et $128 - 128 \left(\frac{128-x}{128} \right)^\alpha$ pour

$x < 128$. Cette loi fait une distorsion autour de la valeur 128. On pourra envisager une modification pour distordre le spectre autour de sa vraie valeur moyenne.

On visualisera la forme de la transformation ainsi que l'effet sur l'image et le spectre. Que peut-il se passer au niveau d'une image en couleurs ?

5 FILTRAGE D'UNE IMAGE

Le filtrage d'une image peut se faire soit par intervention directe sur celle-ci (convolution par le noyau du filtre) soit en passant dans l'espace de Fourier (intervention dans l'espace dual).

5.1 FILTRAGE PAR TRANSFORMATION DE FOURIER

L'idée est de faire la transformée de Fourier de l'image (attention c'est une fonction complexe) puis par transformation inverse de revenir à l'image.

Ecrire un programme permettant cette opération (en utilisant les fonctions de transformées de Fourier 2D de Labview). On visualisera l'« image » dans l'espace réciproque (on peut pour cela utiliser un graphe d'intensité visualisant directement le contenu de la matrice représentant la TF) puis on s'assurera que l'image reformée n'est pas altérée.

5.2 FILTRAGE MATRICIEL (FILTRE RIF)

L'idée est de définir la réponse impulsionnelle du filtre sous la forme d'une matrice $F_{a,b}$ ($0 \leq a < A$ et $0 \leq b < B$) (noyau du filtre) et de calculer le produit de convolution de l'image $I_{i,j}$ ($0 \leq i < N$ et $0 \leq j < M$) par ce noyau. L'image transformée $I'_{i,j}$ s'obtient par l'opération :

$$I'_{i,j} = \sum_{k=0}^{A-1} \sum_{l=0}^{B-1} F_{k,l} I_{i+k,j+l}$$

Ecrire un programme permettant de calculer ce produit de convolution pour une taille quelconque du noyau.

5.3 PASSE-BAS (BRUIT, RECONSTRUCTION, ANTIALIASING ...)

Expérimentez ces deux méthodes pour un filtre passe bas.

- Fourier : on pourra utiliser une fenêtre rectangulaire dans l'espace de Fourier pour couper les hautes fréquences.
- Matriciel : on peut utiliser une matrice uniforme (moyenne mobile) :

Exemple : $\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$

Appliquez ces filtres en vue de corriger des effets de bruits, d'aliasing ou pour reconstruire une image sous-échantillonnée.

On pourra enfin tester des filtres non linéaire (Médian, Nagao ...)

5.4 PASSE-HAUT (DETECTION DE CONTOURS, RENFORCEMENT ...)

Expérimentez ces deux méthodes pour un filtre passe haut.

- Fourier : on pourra utiliser une fenêtre rectangulaire dans l'espace de Fourier pour couper les basses fréquences.
- Matriciel : on peut utiliser une matrice dite de dérivation ou de gradient ou encore un opérateur dit Laplacien (dérivée seconde) :

$$\text{Ex : Gradient selon x : } \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} \quad \text{Laplacien en croix : } \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Appliquer ces filtrages pour la détection de contours ou le renforcement de contours qui peut se faire en soustrayant à l'image de départ son image de contours.

6 INTERCORRELATION (FACULTATIF)

6.1 COLLAGE D'IMAGES PAR INTERCORRELLATION.

Nous proposons ici d'illustrer l'utilité de la fonction d'intercorrélation pour coller deux images. Il s'agira dans notre exemple de trouver quelle est la translation à opérer pour superposer les deux images « nénuphars1 » et « nénuphars2 ». Pour cela on calculera la fonction d'intercorrélation I des deux images I^1 et I^2 :

$$I_{i,j} = \sum_{k=0}^{k=N} \sum_{l=0}^{l=M} I^1_{k,l} I^2_{i+k,j+l} \quad \text{pourra être visualisé sur un graphe d'intensité (on pourra aussi}$$

réfléchir au problème des conditions aux bords des images : convolution circulaire ou non)