

Programmation pour Ingénieur

Projet

ME 4^e semestre

Rev. 2024.10

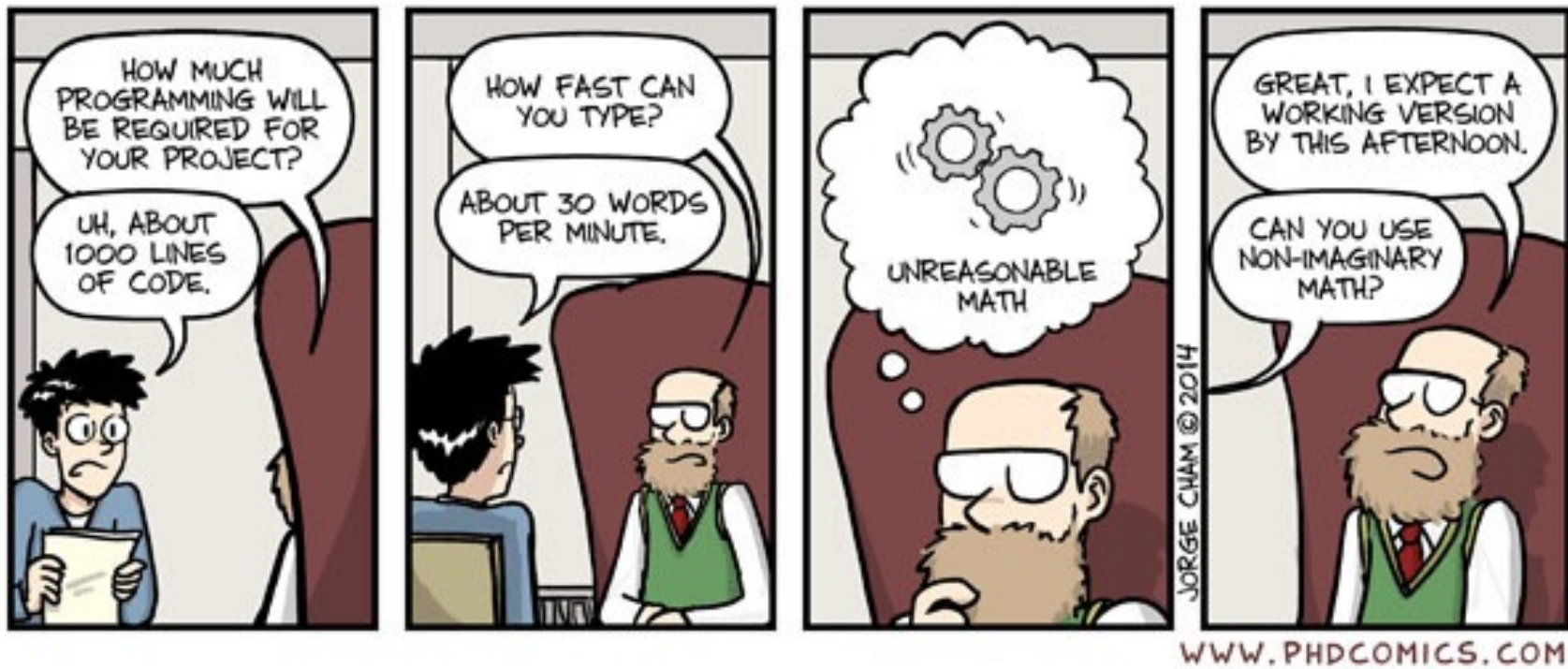
Christophe Salzmann

Laboratoire
d'Automatique

modifications

- 19.08.2024, r1 – version initiale
- 25.08.2024, r2 – typos
- 05.09.2024, r3 – Procédure d'évaluation(3), pénalités et ChatGPT
- 10.09.2024, r4 – pas de rotation, màj paramètres FindTraces()
- 17.09.2024, r5 – précision sélection traces (=tri) S.22.
- 30.09.2024, r6 – histogramme à ~~(255)~~ 256 valeurs S.21.
- 01.10.2024, r7 – éclaircissements point 6) S.22.
- 08.10.2024, r8 – Erreur si fichier .png avec plus que **10** traces. S.53
- 30.10.2024, r9 – MàJ. (précision) S.10 calcul de M
- 02.12.2024, r10- pas de warning si plus de 5 traces, S.54

Programming is not typing code!



Projet

But:

- Vous familiariser avec les 3 environnements du cours
- Mettre en œuvre les concepts vus, acquérir de la pratique
- Progresser au travers d'un exercice de plus longue haleine
- Entraîner votre *computational thinking*
- ...

Le projet est découpé en 3 parties:

- Analyse de l'image pour identifier les traces (C)
- Lecture, conversion de l'image, génération du script ML (LabVIEW)
- Redressement de la perspective et affichage (matlab)

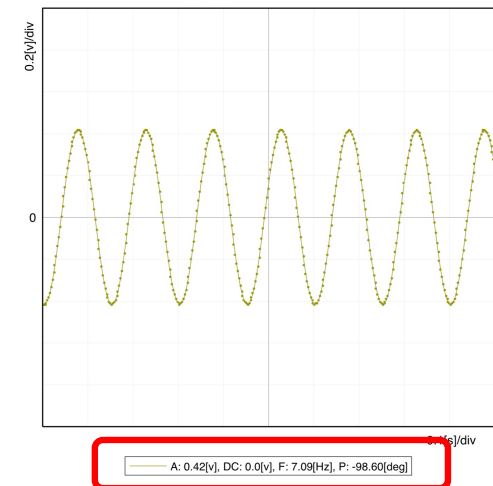
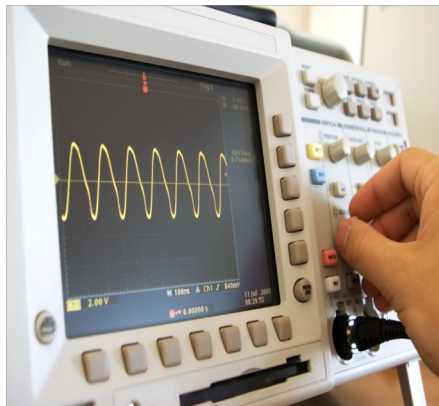
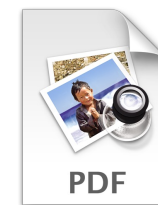
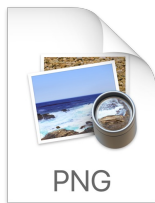
L'échange d'informations entre les environnements se fait à l'aide de fichiers sauvés sur le disque dur

Etapes du Projet

- Ci-après les étapes principales du projet. Elles ont pour but de vous aider à appréhender les algorithmes du projet.
- Des exercices et des explications supplémentaires seront fournis durant le cours.
- N'hésitez pas à poser vos questions lors des séances projets et sur le forum Ed du cours.
- Veuillez également regarder les mises à jour sur la page moodle du cours.

Récupération de données

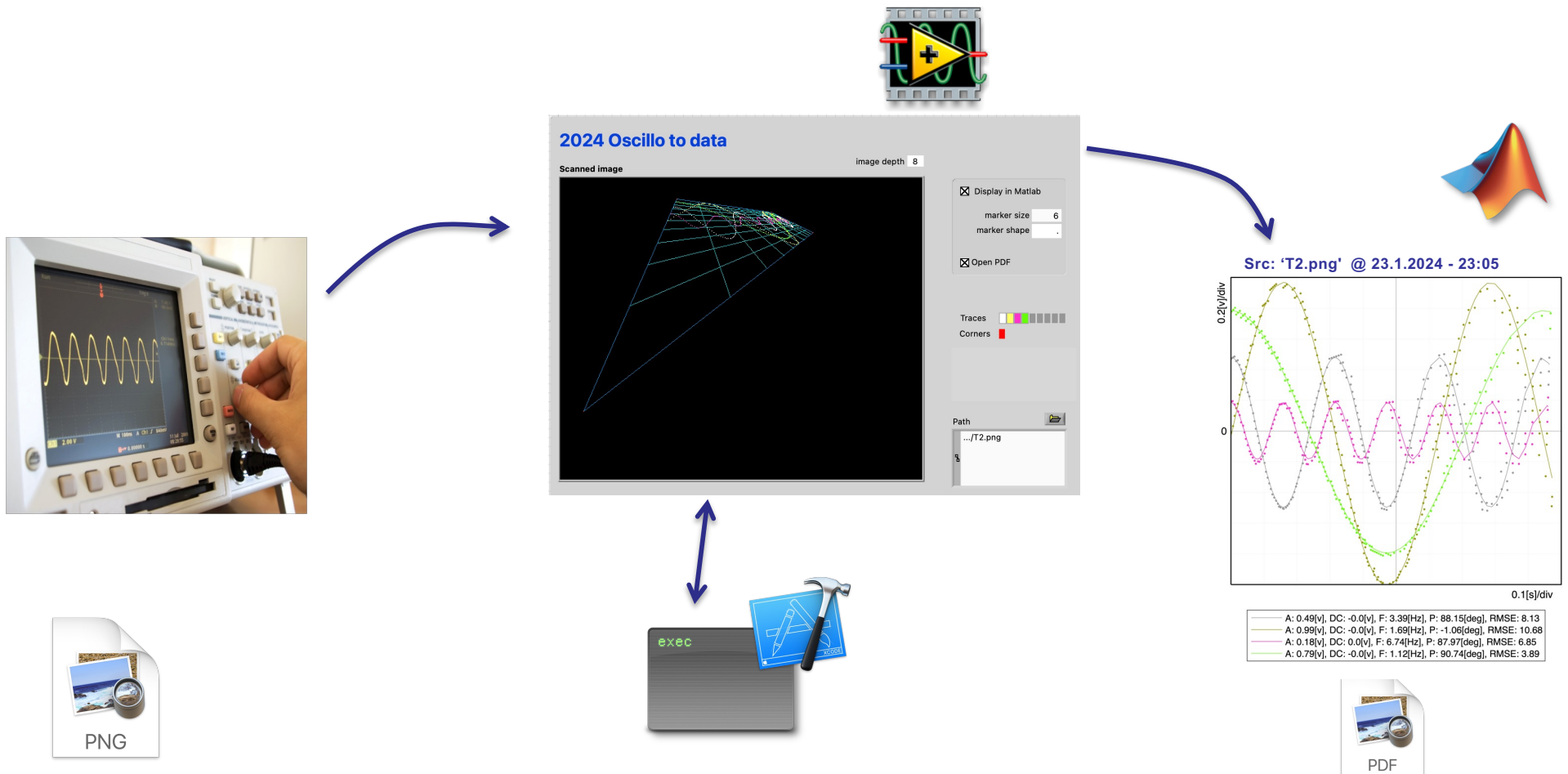
Implémentation à l'aide d'outils performants !



L'image n'est pas forcément prise de face, elle aura une certaine **perspective** qu'il faudra redresser!

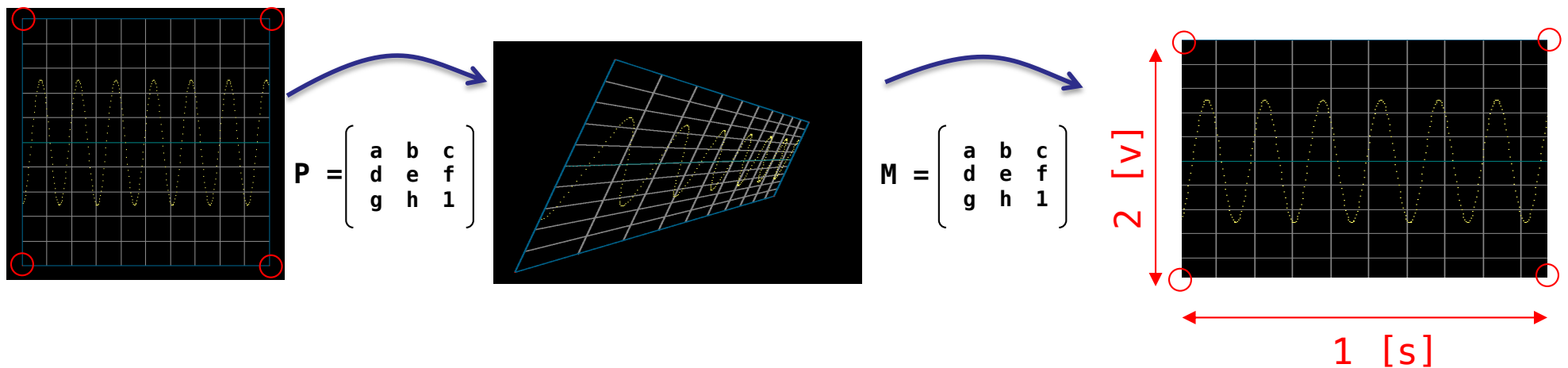
Demo – Oscillo2data

Le but de ce projet est de vous familiariser avec les 3 environnements vus au cours. Il vous permettra de mettre en œuvre les différentes phases de la création d'un programme. Chaque environnement gèrera une étape du projet.



Perspective

Les 4 angles sont identifiés par une couleur unique. Ces 4 points de référence permettront de calculer la matrice de transformation \mathbf{M} qui redressera l'image. La matrice \mathbf{P} a été utilisée pour tordre l'image initiale.



Par convention la largeur de l'image redressée représente 1 [s] et la hauteur totale est de 2[v]. Il y a 10 divisions en horizontale et verticale.

Le 0[v] est au centre de l'axe Y.

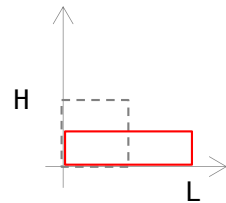
Dans le fichier PDF, la largeur et la hauteur du graphique seront identique en taille, i.e un carré de 1 [s] par 2[v].

Matrice de transformation 2D coordonnées homogènes

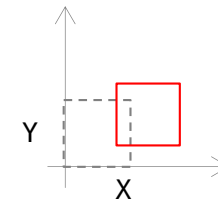
$$M = \begin{pmatrix} \boxed{a} & \boxed{b} & \boxed{c} \\ \boxed{d} & \boxed{e} & \boxed{f} \\ \boxed{g} & \boxed{h} & 1 \end{pmatrix}$$

c,f : translation
g,h : perspective
a,e : scaling
b,d : skew
a,b,d,e : rotation

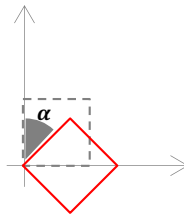
$$M = \begin{pmatrix} L & 0 & 0 \\ 0 & H & 0 \\ 0 & 0 & 1 \end{pmatrix}$$



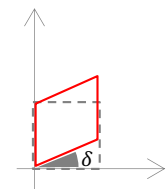
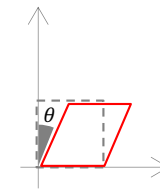
$$M = \begin{pmatrix} 1 & 0 & X \\ 0 & 1 & Y \\ 0 & 0 & 1 \end{pmatrix}$$



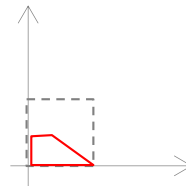
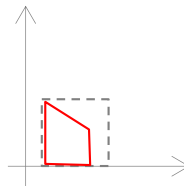
$$M = \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$



$$M = \begin{pmatrix} 1 & \tan(\theta) & 0 \\ \tan(\delta) & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$



$$M = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ P_x & P_y & 1 \end{pmatrix}$$



$$\begin{pmatrix} x_p \\ y_p \\ w_p \end{pmatrix} = M * \begin{pmatrix} x_s \\ y_s \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} x \\ y \\ w \end{pmatrix}_{\text{Homogène}} \Leftrightarrow \begin{pmatrix} x/w \\ y/w \end{pmatrix}_{\text{Cartésien}}$$

$$\text{Pixels source } \begin{pmatrix} x_s \\ y_s \end{pmatrix}_{\text{Cartésien}} \rightarrow \begin{pmatrix} x_s \\ y_s \\ 1 \end{pmatrix}_{\text{Homogène}}$$

$$\text{Pixels perspective } \begin{pmatrix} x_p \\ y_p \\ w_p \end{pmatrix}_{\text{Homogène}} \rightarrow \begin{pmatrix} x_p/w_p \\ y_p/w_p \end{pmatrix}_{\text{Cartésien}}$$

$$x_p' = (ax_s + by_s + c)/(gx_s + hy_s + 1)$$

$$y_p' = (dx_s + ey_s + f)/(gx_s + hy_s + 1)$$

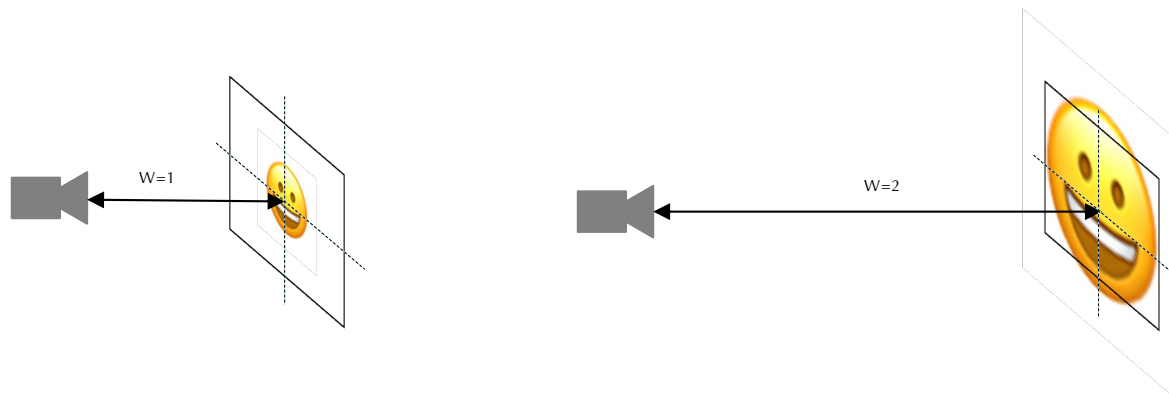
Matrice de transformation 2D coordonnées homogènes

L'utilisation des coordonnées homogènes permet de chainer les opérations en multipliant les matrices de transformations et ainsi avoir une seule transformation finale. Elles sont abondamment employées en informatique pour l'affichage 2 et 3D.

Il faut noter que ce chainage de transformations n'est pas forcément commutatif.

En plus des transformations affines (rotation, translation, scaling, skew) elles permettent de définir une perspective (termes g et h de M). Dans ce cas des parallèles dans l'image source ne sont plus forcément parallèles dans l'image de destination.

Le facteur W (coordonnées homogènes) est un facteur d'échelle qui représente l'éloignement de la scène (image)



Matrice de transformation 2D coordonnées homogènes

$$M = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix}$$

$$\text{Pixels source } (x_s, y_s) \rightarrow (x_s, y_s, 1)$$

Cartésien Homogène

$$M * \begin{bmatrix} x_s \\ y_s \\ 1 \end{bmatrix} = \begin{bmatrix} x_d \\ y_d \\ w_d \end{bmatrix}$$

$$x_d = (ax_s + by_s + c)$$

$$y_d = (dx_s + ey_s + f)$$

$$w_d = (gx_s + hy_s + 1)$$

$$(X, Y, W) \Leftrightarrow (X/W, Y/W)$$

Homogène Cartésien

$$x_d = (ax_s + by_s + c) / (gx_s + hy_s + 1)$$

$$y_d = (dx_s + ey_s + f) / (gx_s + hy_s + 1)$$

Pour avoir la forme $Ax = b$

$$ax_s + by_s + c - gx_s x_d - hy_s x_d = x_d$$

$$dx_s + ey_s + f - gx_s y_d - hy_s y_d = y_d$$

Nous avons 8 équations pour 8 inconnues, la dernière valeur de M étant 1

Pour un angle:

$$x_s a + y_s b + 1c + 0d + 0e + 0f - x_s x_d g - y_s x_d h = x_d$$

$$0a + 0b + 0c + x_s d + y_s e + 1f - x_s y_d g - y_s y_d h = y_d$$

$x_s \ y_s$ un des 4 angles sources, connus

$x_d \ y_d$ l'angles destinations désiré, connus

A répéter pour les 3 autres angles, afin d'avoir une matrice A 8x8

Reste à construire A et b pour trouver x qui est la matrice M

Matlab - Algèbre linéaire

Soit le système d'équations suivant

$$\begin{array}{rrcrcl} 10a & + & 7b & + & 8c & + & 7d & = & 32 \\ 7a & + & 5b & + & 6c & + & 5d & = & 23 \\ 8a & + & 6b & + & 10c & + & 9d & = & 33 \\ 7a & + & 5b & + & 9c & + & 10d & = & 31 \end{array}$$

Il peut être décrit sous la forme $Ax = b$,

avec

```
>> A = [10 7 8 7;...  
        7 5 6 5;...  
        8 6 10 9;...  
        7 5 9 10];  
  
>> B = [32;23;33;31];  
  
>> det(A)  
ans =  
1 % det ≠ 0 => A can be inverted
```

Il peut être résolu:

```
>> x = inv(A)*b  
x =  
1.0000 a  
1.0000 b  
1.0000 c  
1.0000 d
```



Matlab - Algèbre linéaire

Alternativement, il peut être résolu avec l'opérateur ' \backslash ' *left matrix divide*

```
>> x = A\b
x =
    1.0000
    1.0000
    1.0000
    1.0000
```

Si la matrice A n'est pas carrée, ' \backslash ' fournit une solution approximée par des moindres carrés, avec une erreur = $A*x-b$

```
>> A = [2 -2;...
        -1 1;...
         3 4];
```

```
>> rank(A)
ans =
     2
```

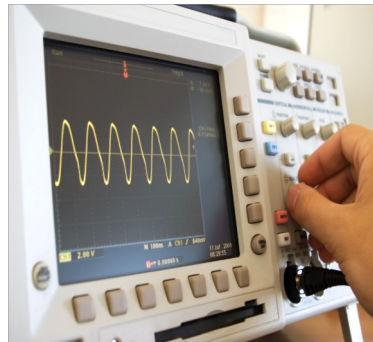
```
>> b = [4;3;2];
```

```
>> x = A\b;
x =
    0.8571
   -0.1429
```

```
>> err = A*x - b
err =
   -2.0000
   -4.0000
         0
```

Etapes du Projet

Oscillo2data.vi



①



T2.png

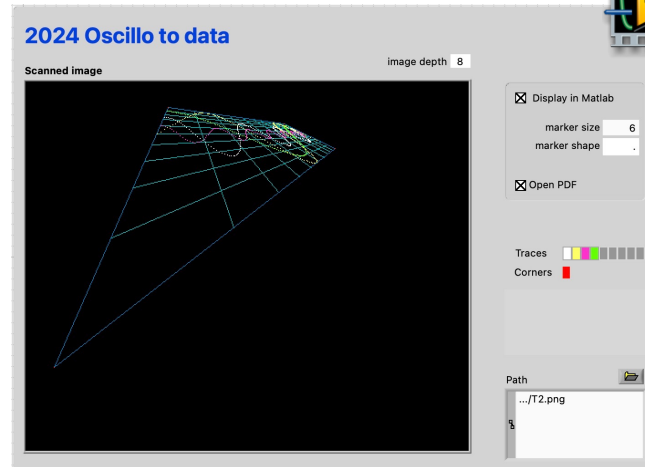
②
Pixmap.bin

idx[]

③
Traces.txt

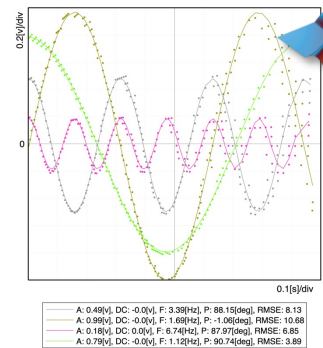


Extract



④
T2.m

Src: 'T2.png' @ 23.1.2024 - 23:05



⑤



T2.pdf

Partie 1 (C) – *Extract*

Ecrire le programme ***Extract*** en C qui va lire le fichier **Pixmap.bin** contenant les pixels d'une image représentant un/des graphiques. Les traces et autres informations utiles contenues dans cette image seront identifiées et sauvées dans le fichier **Traces.txt**

Ce fichier sera ensuite analysé par matlab.

En plus des données sauvées dans le fichier **Traces.txt**, votre programme retourne dans **stdout** une string contenant les indexes **Idx[]** des couleurs correspondant aux bords/corners et aux traces.

Extract – idée générale

Le **Pixmap** est composé de pixels de différentes couleurs.

Vous ne savez pas à l'avance quelle couleur correspond à une trace, au fond, au bord, à du bruit ou à un point de contrôle.

Pour déterminer les couleurs utiles il faut commencer par faire un **histogramme** des couleurs rencontrées.

Les couleurs étant encodées sur 8bits (CLUT) l'histogramme à $2^8 = 256$ couleurs possibles.

- Une **trace** a entre 50 et 300 pixels de couleur identique
 - Il peut y avoir entre 1 et 5 traces dans une image
 - S'il y a plus de 5 traces, choisir les 5 avec le plus grand nombre de points
- Il y a exactement 4 **points de contrôles**
- Les autres pixels peuvent avoir des valeurs quelconques
 - => bruit/fond/bords

Extract – idée générale

Une fois l'**histogramme** fait, il faut y rechercher :

- les couleurs des traces, ce sont les indexes de l'histogramme dont les valeur sont entre 50 et 300
- La couleur des points de contrôle, il y en a exactement 4

Il faut mémoriser ces couleurs et analyser à nouveau **Pixmap** en y extrayant les coordonnées x/y des pixels correspondant à une trace ou point de contrôle.

Ces coordonnées seront ensuite sauvées dans le fichier **Traces.txt**

Extract – format de Pixmap.bin

Le fichier **Pixmap.bin** correspondant est une suite de valeurs au format binaire structuré comme ci-dessous.

La profondeur, la largeur et la hauteur de l'image sont encodés sur 16 bit non signé **unsigned short**.

Les pixel sont encodés sous la forme d'un byte ou char non signé **unsigned char**.

Profondeur de l'image en bit

Largeur de l'image en pixels

Hauteur de l'image en pixels

pixel 0

pixel 1

..

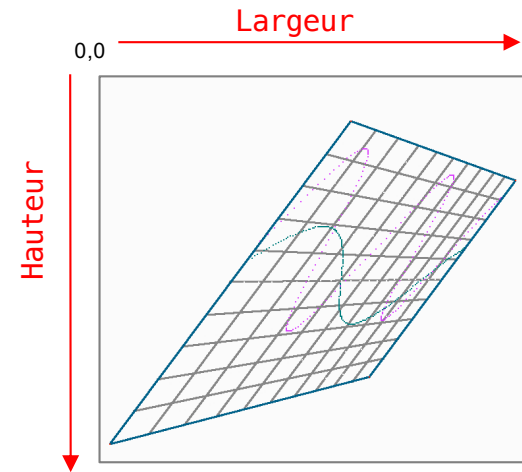
..

last pixel (il y a Largeur x Hauteur pixels dans le fichier)

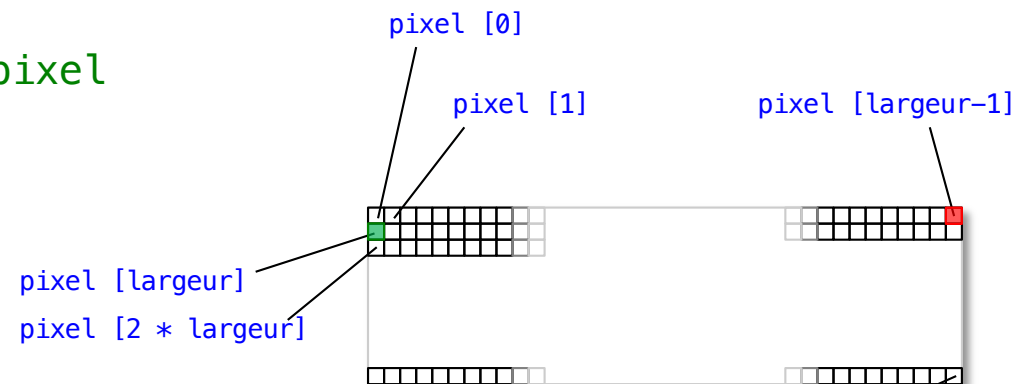
Extract – format de Pixmap.bin

Exemple du contenu du fichier **Pixmap.bin** correspondant à l'image ci-contre :

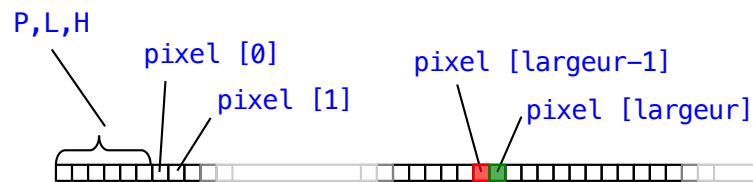
```
8      // profondeur (short)
370    // largeur en pixels (short)
350    // hauteur en pixels (short)
11     // valeur pixel 0 (uchar)
10     // valeur pixel 1 (uchar)
...
...
11     // valeur du dernier pixel
```



Organisation des pixels en mémoire



Organisation des pixels dans le fichier



Extract – format de Traces.txt

Le fichier texte de sortie **Traces.txt** contient

- i) Les coordonnées des 4 *corners*
- ii) Les coordonnées des points formant une trace, il peut y avoir de 1 à 5 traces

La syntaxe employée suit le formalisme matlab:

```
Corners= [  
  xCorner1, yCorner1;  
  xCorner2, yCorner2;  
  xCorner3, yCorner3;  
  xCorner4, yCorner4;  
];
```

```
C0= [  
  xC01, yC01;  
  xC02, yC02;  
  ...  
];
```

```
C1= [  
  xC11, yC11;  
  xC12, yC12;  
  ...  
];
```

```
...
```

```
T = {C0 C1 ...};
```

Extract – format de Traces.txt

Exemple :

```
Corners= [  
66, 315;  
193, 237;  
171, 107;  
18, 30;  
];
```

```
C0= [  
72, 226;  
...  
71, 225;  
];  
...
```

```
C1= [  
72, 226;  
73, 226;  
...  
];
```

```
...
```

```
T = {C0 C1};
```

Extract - Etapes

1) Lecture du fichier **Pixmap.bin**

- Lire les informations concernant la taille de l'image
- Valider les bornes, par. ex. taille négative!
- Reporter une erreur en cas de problème

2) Lire les pixels de l'image et les stocker dans un tableau dynamique (malloc)

- Lire tous les pixels en une fois et les stocker dans un tableau dynamique **Pixmap**
- Valider la bonne lecture des pixels et reporter une erreur en cas de problème (ex. pas assez de pixels)

3) Création de l'histogramme des couleurs

- L'histogramme est un tableau de 256 cases créé dynamiquement, chaque case correspond à une couleur, la valeur de la case indique le nombre de pixels de la couleur de la case.
- Parcourir le **Pixmap** et mettre à jour l'histogramme en fonction de la valeur des pixels

4) Parcourir l'histogramme et identifier les entrées/couleurs d'intérêt

- Il y a exactement 4 points de contrôle, i.e. une entrée du Pixmap qui a une valeur de '4'. Attention il ne peut y avoir qu'une entrée de l'histogramme avec une valeur de '4', s'il n'y en a pas ou plus c'est une erreur.
- Il ensuite identifier les indexes des courbes. Ils ont un nombre de valeurs entre 50 et 300.
- Il y a entre 1 et 5 indexes de courbes, s'il y a plus que 5 indexes de courbe, choisir les 5 premiers avec le plus grand nombre de points.

Extract – Etapes suites

5) Parcourir à nouveau le Pixmap et mémoriser les coordonnées des couleurs d'intérêt

- Sauvegarder les coordonnées des 4 points de contrôle, i.e. les points dont l'index de l'histogramme vaut '4'.
- Sauvegarder les coordonnées des points des traces. Il y a entre 0 et 5 traces de 50 à 300 points chacune. Si vous avez plus que 5 traces, gardez les 5 avec le plus grand nombre de pts (dans les bornes données). Pour cela vous devez trier (ex. insert sort) de manière décroissante les traces par le nombre de pts.

6) Sauvegarder les données

- Sauvegarder les coordonnées des corners et traces dans le fichier **Traces.txt**
- Selon le format du slide "*Extract* – format de Traces.txt" soit

```
Corners= [... ]
C0 = [];
...
T = {C0 }
```

7) Retourner les indexes des couleurs de corners et des traces dans **stdout**

- Format:

```
C: cc           index de la couleur des corners
T: tt0 tt1 tt2  indexes des couleurs des traces
```

Ne pas oublier de tester les erreurs!

Extract - Validation

Une fois votre code compilé, vous validez votre programme en effectuant des tests avec différentes valeurs. En premier, testez avec des valeurs plausibles et valides et ensuite avec des valeurs fausses ou impossibles.

Vous contrôlerez par ailleurs que votre programme suit les spécifications définies précédemment.

Ex.

- Largeur ou hauteur invalides ou en dehors des bornes raisonnables (100 ..1000)

```
4294967196    // hauteur en pixels en dehors des bornes, unsigned int correspondant à -1000
```

- Vous avez le nombre correct (largeur x hauteur) de pixels

```
440           // largeur en pixels
430           // hauteur en pixels
```

-> vous devez avoir $440 \times 430 = 189200$ pixels dans votre fichier, **pas plus, pas moins**

- Problème lors de l'ouverture ou la création des fichiers

etc.



Pixmap.bin

Lecture de valeurs binaires

Employez la fonction **fread()** pour les lire dans des fichiers binaires, format :

```
#include <stdio.h>
```

```
size_t fread(void *ptr, size_t size, size_t nmemb, FILE *stream);
```

Avec

***ptr**: un pointeur sur ma variable/buffer de **nmemb** éléments de taille **size**

nmemb: un entier indiquant le nombre d'éléments (de taille **size**) à lire

size: un entier indiquant la taille d'un élément en byte(s)

***stream**: un pointeur sur le descripteur de fichier/stream

Retourne

le nombre d'éléments lus (si **size** est 1) ou une erreur/fin de fichier

La fonction **fread()** va lire les **nmemb** élément (de taille **size**) **suivants** dans le fichier ***stream** et va les mettre dans le buffer/variable ***ptr** passé en paramètre.

Ex.

```
unsigned int Largeur=0;
```

```
ret = fread(&Largeur,sizeof(unsigned int),1,fp);
```

```
// Copie dans Largeur les 4 prochains bytes (size(unsigned int)=4) lu dans fp.
```

```
unsigned int Pixmap[1000];
```

```
ret = fread(Pixmap,sizeof(unsigned int),1000,fp);
```

```
// Copie dans Pixmap les 1000 prochains pixels (unsigned int) lu dans fp.
```

```
// Pixmap doit être assez grand pour stocker les données lues!
```

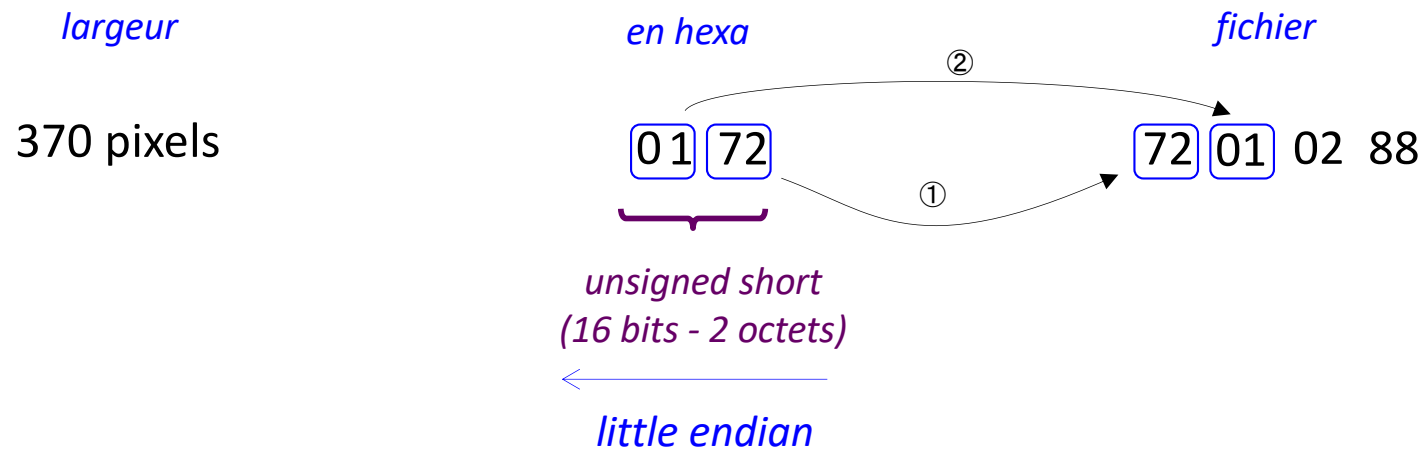


Pixmap.bin

Big et Little Endian

Il y a deux manières d'organiser les octets dans un fichier, soit l'octet de poids le plus fort est enregistré à l'adresse mémoire la plus petite (Big Endian), soit l'inverse (Little Endian). Un octet est représenté par 2 caractères hexadécimal (0..9, A..F), un short (entier 16 bit) non-signé comprend 2 octets, soit 2 caractères hexadécimal (ou 16 bits).

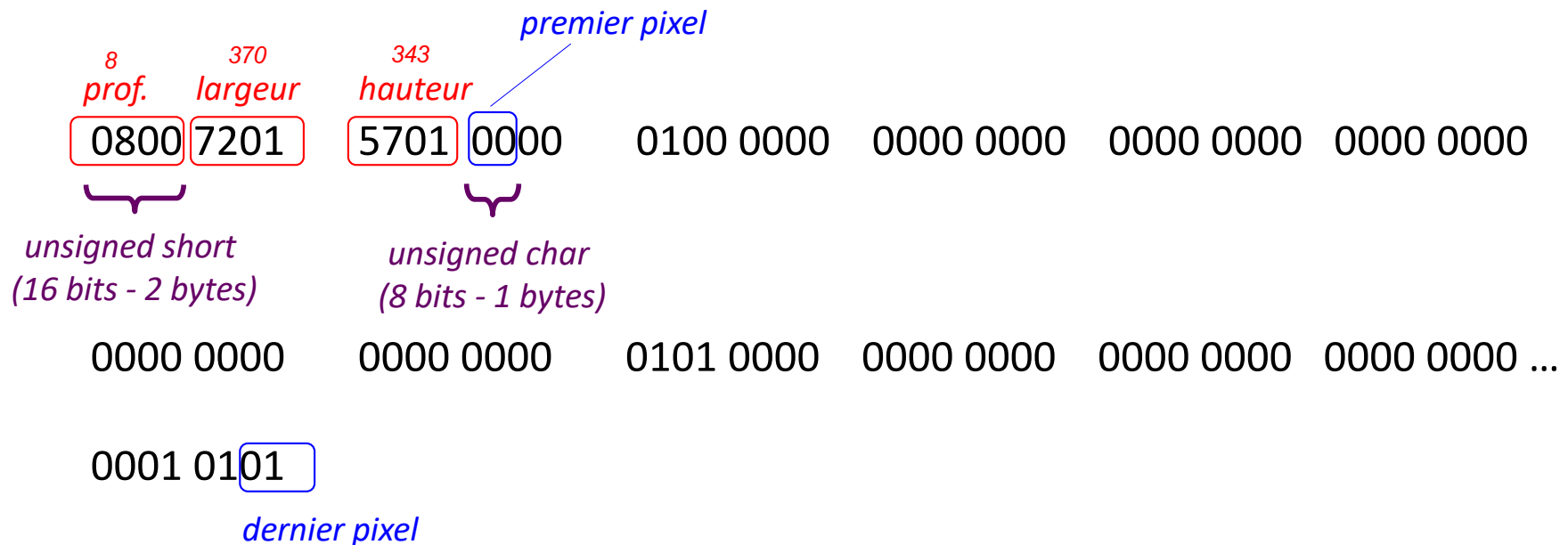
Exemple:





Extract – format de Pixmap.bin

Pixmap.bin affiché au format hexadecimal est composé de short int non signés: entiers non-signés **unsigned short** pour la **profondeur**, la **largeur** et la **hauteur**, ici *l'endianess* est importante
puis de **unsigned char** pour les pixels 0 à *last pixel*, *l'endianess* n'a pas d'importance!



Paramètres de la ligne de commande

Des paramètres sont passés via la ligne de commande à votre exécutable C.

Ex. `./Extract '/path1/Pixmap.bin' '/path2/Traces.txt'`

Ces paramètres sont des chaînes de caractères. Il y a 2 paramètres:

- Le chemin du fichier **Pixmap.bin** '`chemin/complet/entre/apostrophe`'
- Le chemin du fichier **Traces.txt** '`chemin/complet/entre/apostrophe`'

Ces paramètres sont disponibles via `int main(int argc, const char * argv[])`

`argv[]`, un tableau de chaînes de char., contient les paramètres de la ligne de commande, `argc` indique le nombre de paramètres

`argv[0]`, indique toujours le nom du programme, dans notre exemple **'Extract'**

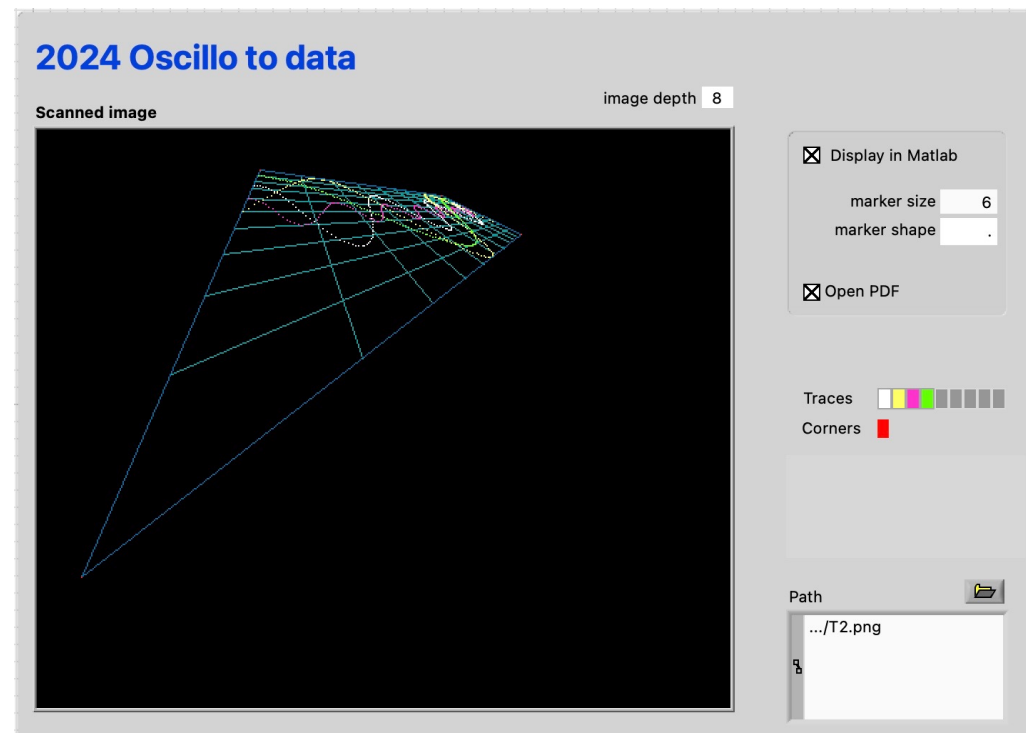
`argv[1]`, contient le chemin complet vers **Pixmap.bin**

...

Partie 2 – *Oscillo2Data.vi*

Le programme LabVIEW est le chef d'orchestre. Il va faire le lien avec les deux autres programmes en générant les lignes de commandes nécessaires pour l'appel de vos programmes C et matlab.

Il va également fournir l'interface graphique qui va permettre aux utilisateurs de sélectionner les différentes options.



Oscillo2data.vi - Etapes

1) Dessiner votre interface utilisateur

- Sélection du chemin contenant le graphique
- Affichage du graphique dans une image
- Les différentes options d'affichage pour matlab doivent être accessibles
- Affichage des erreurs éventuelles

2) Lecture d'une image **.png**

- Lecture de l'image courante, ex. **test1.png**
- Extraction des pixel de l'image (.png -> pixmap **couleur**)
- Ecriture des pixels dans **Pixmap.bin**
- Lancement de votre exécutable C via la ligne de commande (avec paramètres)
- Lecture et mémorisation de **Traces.txt**
- Récupération de l'index de la couleur des corners et des indexes des couleurs de traces

Oscillo2data.vi - Etapes

3) Creation et sauvegarde du script Matlab ***nomdufichierPNG.m***

- Générer le script Matlab en tenant compte des paramètres spécifiés dans l'interface utilisateur
- Ecriture du fichier script ***nomdufichierPNG.m***
- Lancer matlab avec le fichier script ***nomdufichierPNG.m***

nomdufichierPNG.xxx

Le nom de votre script matlab sera le même que le nom de la figure source et du fichier résultant PDF, uniquement l'extension du fichier change.

Ne pas oublier la gestion des erreurs

Partie 3 – *Matlab script file*

Le redressement des pixels l'image et l'identification des courbes se fait à l'aide de Matlab.

L'algorithme est le suivant:

1. Ordonner les 4 bords de l'image (corners)
2. Calculer la matrice de perspective inverse en se basant sur les 4 bords
3. Redresser les traces identifiées
4. Identifier les paramètres des traces (sinus/polynômes)
5. Afficher les points et les courbes identifiées
6. Afficher les valeurs clés des courbes identifiées (amplitude, phase, etc).
7. Sauvegarder au format PDF le plot redressé et les informations s'y rapportant

Partie 3 – *Matlab script file*

Exemple

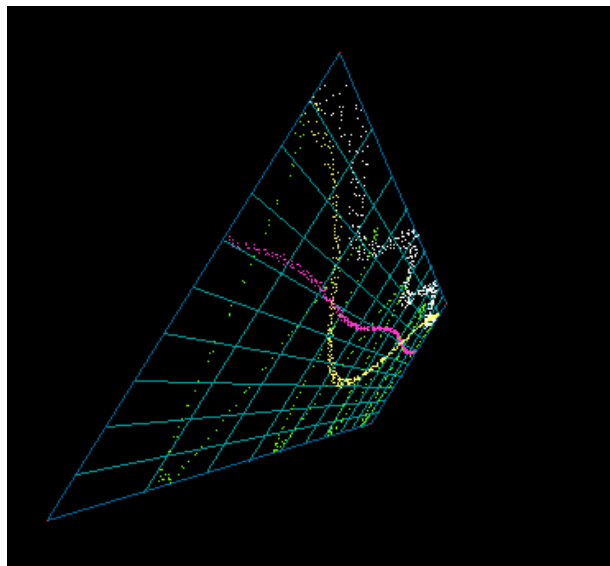
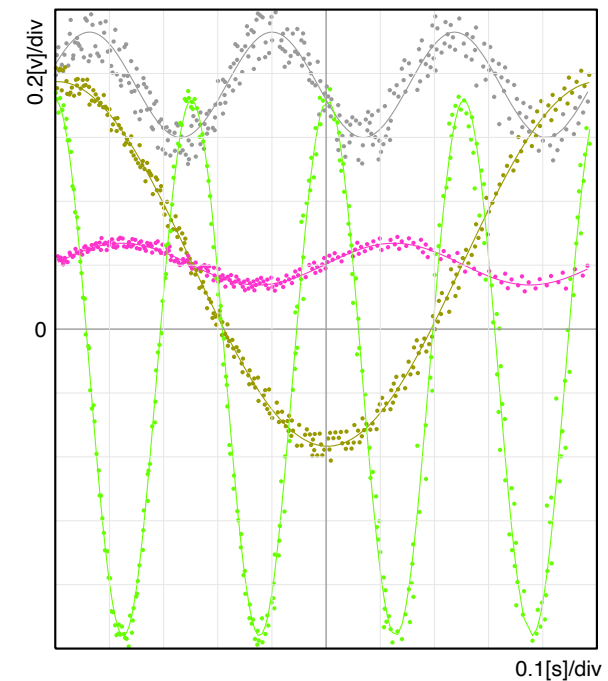


Image source



Src: 'T4b.png' @ 18.08.2024 – 23:55:28



—	A: 0.17[V], DC: 0.8[V], F: 4.00[Hz], P: 22.48[deg]
—	A: 0.57[V], DC: 0.2[V], F: 1.35[Hz], P: 88.63[deg]
—	A: 0.07[V], DC: 0.2[V], F: 2.69[Hz], P: 1.79[deg]
—	A: 0.84[V], DC: -0.1[V], F: 5.36[Hz], P: 90.03[deg]

PDF final

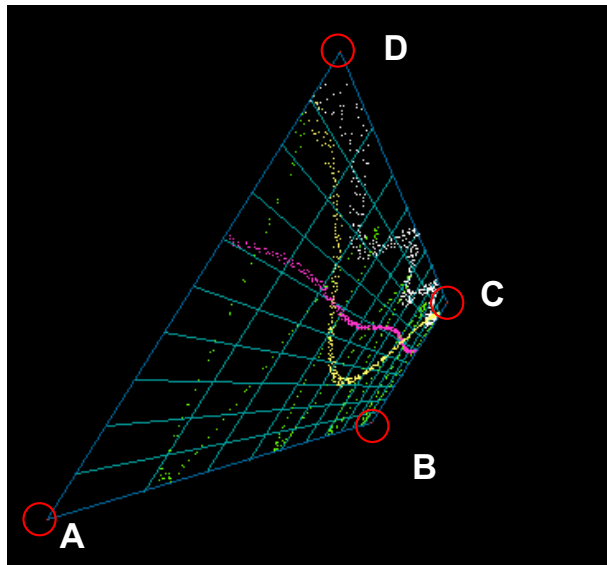
Partie 3 – *nomdufichierPNG.m*

Réordonner les 4 points de contrôle ABCD (aux angles)

Les 4 points de contrôle sont dans un ordre quelconque.

Il faut les ordonner dans le sens inverse des aiguilles d'une montre.

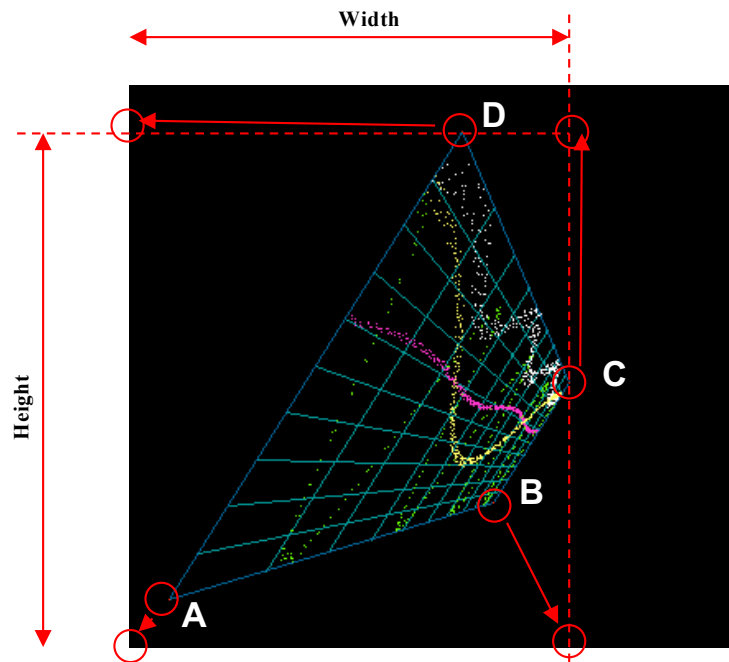
Par convention le premier point (A) est celui qui est le plus proche de l'angle en bas à gauche.



Partie 3 – *nomdufichierPNG.m*

Perspective inverse

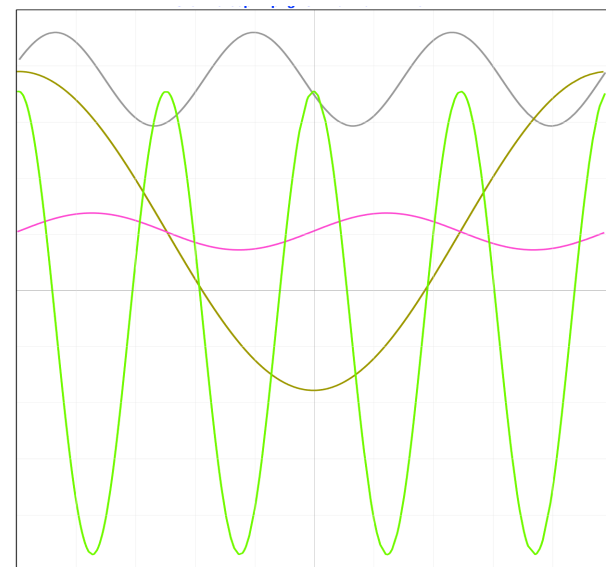
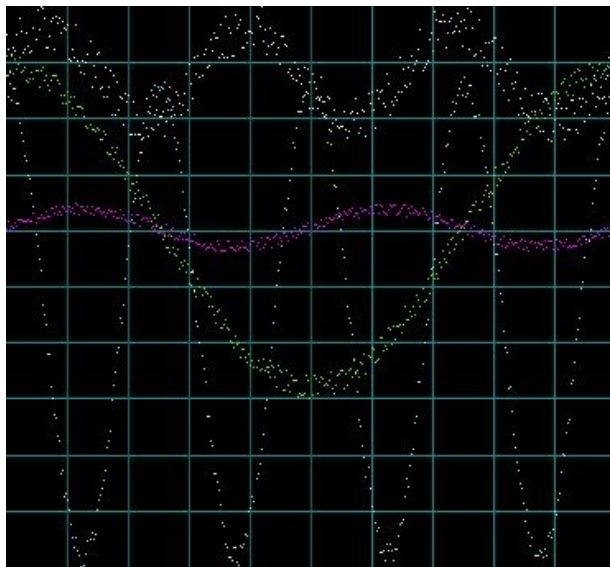
Calculer la matrice de transformation (3x3) **computeM()** qui mappe les 4 points de contrôle vers les 4 angles de l'image



Partie 3 – *nomdufichierPNG.m*

Identifier les courbes et *fitter* un sinus ou un polynôme de degré 3 (sin1, poly3)

Au besoin enlever la composante continue (DC) des signaux



Partie 3 – *nomdufichierPNG.m*

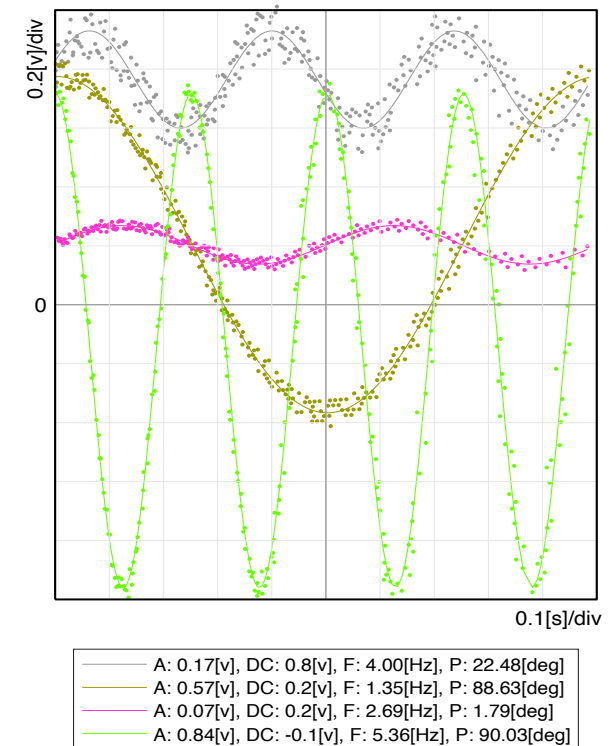
Calculer les paramètres des sinus/polynomes.

Sauvegarder les graphiques et les données dans un fichier PDF

En plus des courbes et des points, le PDF affiche:

- Le nom du fichier source
- La date et l'heure de la création du fichier
- Les paramètres des courbes
 - Si sinus (**A**mplitude, **O**ffset/**DC** , **F**réquence, **P**hase)
 - Si polynôme (coefficients du polynôme)
- La ligne 0[v] et la ligne 0.5 [s]
- Les 9 lignes de séparation horizontales et verticales
- Les échelles X et Y

Src: 'T4b.png' @ 18.08.2024 – 23:55:28



Partie 3 – *Matlab*

Le nom du fichier matlab contenant votre code est similaire au nom du fichier png source, l'extension du fichier '**.png**' étant remplacée par '**.m**'.

En plus du programme principal, écrire les fonctions suivantes:

```
function [A,B,C,D] = orderCorners (Corners)
```

Qui ordonne les 4 coins dans le sens contraire des aiguilles d'une montre, avec le point A en bas à gauche.

```
function M = computeM(A,B,C,D, Width, Height)
```

Qui calcule la perspective inverse en se basant sur les 4 coins ABCD, la largeur et la hauteur de l'image.

Les points ABCD sont *mappés* vers les 4 angles [(0,0) (W,0) (W,H) (0, H)]

Partie 3 – *nomdufichierPNG.m*

```
function [x y] = applyM(M, X, Y)
```

Qui applique la matrice de transformation M aux points contenus dans les vecteurs X Y.

```
function [ParmStr, yfit] =  
FindTrace(X, Y, Width, Height)
```

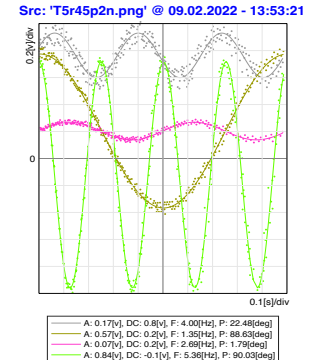
Qui identifie les paramètres caractéristiques (**A**mplitude, **F**réquence, **P**hase, Offset/**DC** ou les **4** coefficients du **p**olynôme) ainsi que l'erreur RMSE des courbes en *fittant* un sinus ou un polynôme sur une trace donnée. Cette fonction retourne également le vecteur **yfit** qui contient le sinus/polynôme identifié évalué aux points de x. **ParamStr** contient une string avec les paramètres pour la légende de l'affichage.

Le choix entre sinus ou polynôme se fait en sélectionnant le fit qui a la plus petite erreur RMSE (via param gof ou `rmse=sqrt(mean((y-yhat).^2))`)

Par convention la largeur représente 1 [s] et la hauteur 2 [v]

Partie 3 – Résolution et affichage

Générer (dans LabVIEW) le script Matlab ***nomdufichierPNG.m***, ce script contient les bords, les traces et les informations utiles s'y rapportant ainsi que les fonctions Matlab pour le calcul, l'affichage et la sauvegarde du résultat dans un fichier au format PDF.



Etapes:

Testez votre script depuis Matlab avec des données connues

Lorsqu'il fonctionne correctement générer ces données depuis LabVIEW

Validation:

Est-ce que les paramètres choisis correspondent à l'affichage ?

Est-ce que les fichiers sont présents ?

Oscillo2Data.vi - Validation

Votre programme LabVIEW a la responsabilité de *commander* les deux autres programmes. Il doit être à même de récupérer les erreurs et de les traiter.

Vous devez particulièrement tester les situations suivantes:

- **Extract** manquant ou pas dans le bon chemin
- Erreur lors de l'exécution de **Extract**
- Pas d'image **.png** dans le chemin fourni
- Problème lors de la création du script matlab **nomdufichierPNG.m**
- Matlab manquant ou pas dans le bon chemin
- Erreur dans le script matlab

Une liste complète des erreurs à traiter est fournie ci-après

Evaluation

Les aspects suivants seront principalement évalués:

- Exécution correcte
i.e. je lance le programme ***Oscillo2Data.vi*** et je regarde le déroulement sans faire d'intervention, si tout est OK vous êtes assuré d'avoir la moyenne au minimum
Ensuite un programme teste votre code C de manière automatique avec des valeurs valides puis invalides et analyse comment les erreurs sont gérées
Dans une moindre mesure je fais de même pour le code matlab et LabVIEW
- "Elégance" et lisibilité du code
- Exactitude du code
- Documentation

Plus d'information ultérieurement

Rendu sur Moodle

- Déposer l'archive .zip (avec vos noms) via la page moodle du cours
- Nombre illimité de soumissions, uniquement la dernière est retenue
- Date de remise: **xx 2024 à 23h55**

Soumission projet 2024



Soumission du projet 2024 sous la forme d'une archive .zip contenant l'entier de vos fichiers.




La taille maximum est de 50 MB.

Nommez votre archive avec les noms des membres du groupe.


File submissions

Maximum size for new files: 50MB, maximum attachments: 1





Files



You can drag and drop files here to add them.

Save changes

Cancel

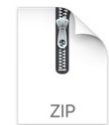
Contenu de l'archive .zip

Rendu sur Moodle de vos fichiers dans une archive .zip aux noms des membres du groupe, avec:

- Brève documentation au format pdf
- Le(s) fichier(s) source(s) c et fichier(s) projet relatif
- Le programme/exécutable (Extract) C compilé pour mac ou windows
- Le(s) VI(s) LabVIEW
- Le(s) script(s) Matlab
- Un jeu de fichiers (*.bin, .txt, .m, .pdf, ...*) générés par vos différents programmes

Contrôlez que **TOUS**** les fichiers se trouvent dans l'archive avant de la soumettre.**

Soumettre l'archive et vérifier ce que vous avez uploadé!




Nom1_Nom2.zip

**Fichier
à soumettre**

Procédure d'évaluation

(A tester dans votre environnement avant la soumission!)

- Décompression de l'archive .zip
- Ouverture du fichier LabVIEW
- Choix de paramètres dans l'interface LabVIEW
- Exécution du VI en cliquant sur la flèche 

Après exécution de la partie (C et LabVIEW) de manière correcte, est-ce que Matlab génère correctement le fichier pdf contenant la solution avec les informations demandées?

oui => vous êtes assuré/e d'avoir la moyenne pour la partie projet
(si vous n'avez pas triché avec votre code!)

non => analyse de votre code et des documents fournis

Procédure d'évaluation (1)

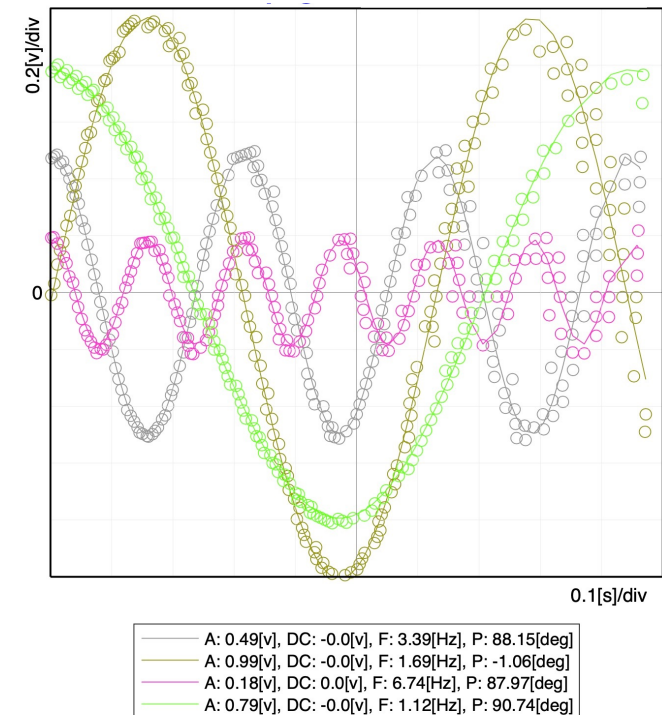


Fichier résultat au format PDF

En plus des courbes et des points, le PDF affiche:

- Le nom du fichier source
- La date et l'heure de la création du fichier
- Les paramètres des courbes
Amplitude, Offset/**DC** , Fréquence, **P**hase
ou
Coefficients du polynôme
- La ligne 0[v] et la ligne 0.5 [s]
- Les 9 lignes de séparation horizontales et verticales
Note: elles sont en gris trop clair dans la figure de droite
- Les échelles X et Y

Src: 'T2.png' @ 23.1.2024 - 23:05



Procédure d'évaluation (2)

C

- Votre code C est analysé en premier via le(s) VI(s) LabVIEW fourni et ensuite de manière indépendante en entrant automatiquement des valeurs pour *Pixmap.bin* et en observant le fichier généré *Traces.txt* ainsi que les messages éventuels dans *stderr* et *stdout*
- **Le comportement de votre programme C est comparé avec les spécifications fournies durant le cours ainsi qu'avec votre documentation!**
- **Attention aux chemins relatifs, noms de fichiers, etc.**
- Analyse du(des) fichier(s) source C

LabVIEW

- Test du bon déroulement du programme LabVIEW, test avec plusieurs fichiers et paramètres (valides et invalide), affichage du résultat dans matlab, **gestion des erreurs**, etc.
- Analyse de la gestion des erreurs en cas de problème avec la partie C ou LabVIEW (message(s) d'erreur et/ou autres mécanismes)
- **Test avec exécutable C manquant**
- **Comment gérez vous le cas d'une image invalide (ex. jpg sélectionné) ou manquante ?**
- **Comment gérez vous le cas d'une image dont la taille est en dehors des bornes?**
- Problème lors de la création du script *nomdufichierPNG.m*
- Analyse du(des) fichier(s) source LabVIEW

Matlab

- Générations correctes des fichiers
- Analyse du(des) fichier(s) source matlab

Procédure d'évaluation (3)

- Les tests se feront sur la machines virtuelles Windows ou sur mac
- Les spécifications et commentaires peuvent être en français ou en anglais
- Mettez des commentaires dans le code C, LabVIEW et matlab
- Relisez les spécifications, est-ce que vos programmes les suivent, ex. noms de fichiers, format, etc.

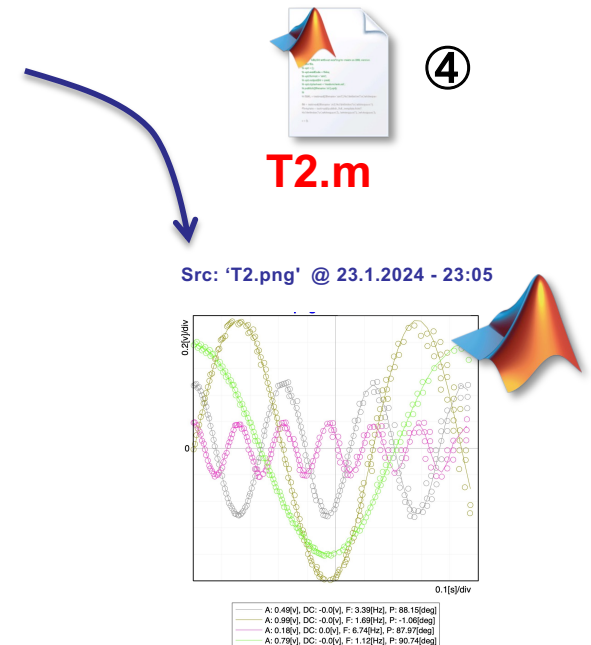
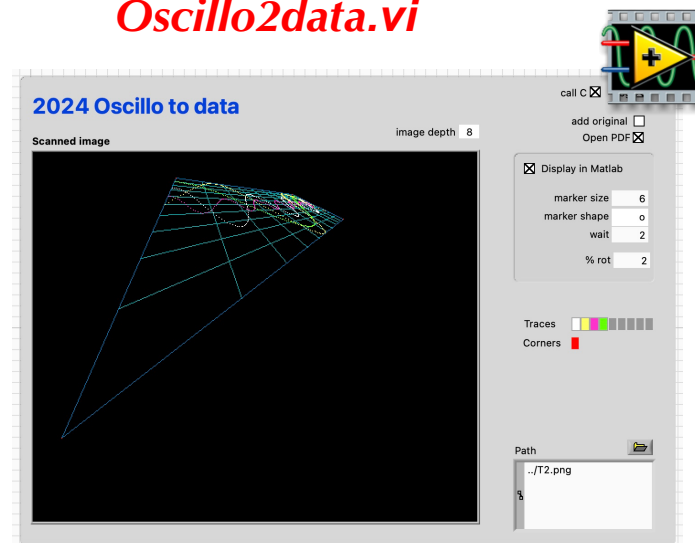
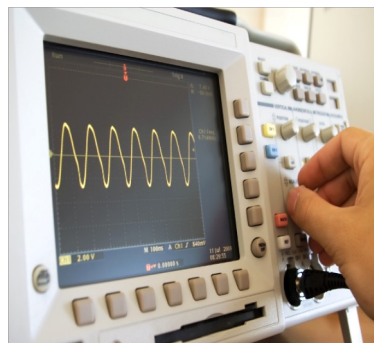
Pénalités assurées si:

- **vous mettez des chemins absolus dans vos codes C/LabVIEW/matlab!**
- **vous n'avez pas au minimum une fonction en C, un en matlab et un sousVI en LabVIEW**
- **vous omettez un/des fichiers dans l'archive zip! Testez que votre archive fonctionne correctement sur une autre machine**
- **vous vous êtes inspiré de codes trouvés sur internet sans mettre les références**
- **vous avez travaillé avec un autre groupe de manière significative sans l'indiquez**
- **vous avez employé ChatGPT ou équivalent, sans joindre le *transcript* de vos conversations en PDF**
- **vous employez des VLA**

Procédure d'évaluation (3)

Respectez impérativement les **noms** des fichiers

Oscillo2data.vi



①



T2.png

②

Pixmap.bin

③

Traces.txt



Extract

⑤



T2.pdf

Procédure d'évaluation (4)

- Les tests se feront sur la machines virtuelles Windows ou sur mac
- Les spécifications et commentaires peuvent être en français ou en anglais
- Mettez des commentaires dans le code C , LabVIEW et matlab
- Relisez les spécifications, est-ce que vos programmes les suivent, ex. noms de fichiers, format, etc.
- **Ne pas mettre de chemin absolu dans vos codes C/LabVIEW/matlab!**
-> pénalité si c'est le cas!
- **Mettez *TOUS* vos fichiers dans l'archive zip! Au besoin testez que votre archive fonctionne sur une autre machine**
- **Si vous vous êtes inspiré de codes trouvés sur internet, mettez les références. Idem, si vous avez travaillé avec un autre groupe.**

Gestion des erreurs & warnings C

Erreurs (impossible de continuer)

- Largeur/ Hauteur en dehors de bornes [**100..1000**]
- Pas assez de pixels (1 à N pixels manquants)
- Impossible de lire dans le fichier d'entrée ***Pixmap.bin***
 - fichier manquant, mauvais nom, droit d'accès
- Impossible d'écrire dans le fichier de sortie ***Traces.txt***
 - lecture seule, droit d'accès
- Pas le bon nombre de paramètres dans la ligne de commande
- 0 traces, i.e. aucune entrée dans l'histogramme avec une valeur entre [**50..300**]
- Fichier .png avec plus que **10** traces
- Pas de corners, i.e. aucune entrée dans l'histogramme avec une valeur de **4**
- Profondeur de l'image autre que **8**

Warnings (à vous de décider si vous continuez ou non)

- Trop de pixels
- ~~Plus de 5 traces~~

Test de validité des paramètres d'entrées

Votre code C doit pouvoir traiter les erreurs spécifiées dans ce document, slide «**Gestion des erreurs & warnings C**» (slide xxx) et générer un message d'erreur correspondant dans stdout, stderr.

En fonction de la manière dont votre code est écrit, certaines erreurs décrites ci-dessus ne peuvent pas être détectées, dans ce cas vous devez l'indiquer dans le fichier de documentation. (ex. pixels excédentaires dans Pixmap.bin)

Vous devez décrire **succinctement** dans vos spécifications comment vous gérez les erreurs ci-dessus. En particulier ce que vous faites en cas d'erreur, par ex. arrêt du programme ou utilisation de valeur par défaut, etc.

Ex.

Largeur de Pixmap.bin en dehors des bornes bornes [100..1000]

-> fermeture de Pixmap.bin et arrêt du programme, message d'erreur 'Width not in range' dans stderr.

Extract - Validation

Une fois votre code compilé, vous validez votre programme en effectuant des tests avec différentes valeurs. En premier, testez avec des valeurs plausibles et valides et ensuite avec des valeurs fausses ou impossibles.

Vous contrôlerez par ailleurs que votre programme suit les spécifications définies précédemment.

Ex.

- Largeur ou hauteur invalides ou en dehors des bornes raisonnables (100 ..1000)

```
4294967196    // hauteur en pixels en dehors des bornes, unsigned int correspondant à -1000
```

- Vous avez le nombre correct (largeur x hauteur) de pixels

```
440           // largeur en pixels
430           // hauteur en pixels
```

-> vous devez avoir $440 \times 430 = 189200$ pixels dans votre fichier, pas plus, pas moins

- Problème lors de l'ouverture ou la création des fichiers

etc.

Code C

- Utilisez des fonctions
- Mettez des commentaires
- Mise en page de votre code

Documentation

De manière générale votre code doit pouvoir être repris par quelqu'un d'autre ou vous-même dans 1 mois ou 10 ans...

En plus d'un algorithme correct et valide, votre code doit être:

- Documenté
- Lisible
- Facilement compréhensible
- Assorti de commentaires dans la partie compliquée
- Compliqué ↗ -> commentaires ↗↗

Voir exemple fourni sur la page moodle du cours

Documentation 1

- Document **PDF** de 1 à 3 (max. 5) pages, pas besoin d'un roman, mise en page simple et bien organisée. Il doit permettre de comprendre votre démarche et la mise en œuvre de celle-ci.
- Contenu:
 - **Nom des auteurs**
 - **Qui à fait quoi**, succinct, 1 ligne par auteur
 - **La plateforme** (linux, OSX, Win, ...)
 - **Le compilateur** employé (xcode, visual studio, etc.)
 - Nom des fichiers et leur fonction
 - Attention** certains noms sont déjà défini dans le slide "Procédure d'évaluation (3)"
 - ex.
Oscillo2data.vi: VI principal à lancer en premier

Documentation 2

- Contenu (suite):
 - Le flow de données entre les applications et à l'intérieur des applications, i.e. les fichiers créés, lus et échangés entre les applications, + stdout & stderr
ex. (éventuellement avec un schéma)
Oscillo2data.vi lance Extract.
Extract lit le fichier Pixmap.bin et génère le fichier Traces.txt, en cas d'erreur affiche "xx" dans stderr
- Descriptions gestion des erreurs:
 - Que fait votre programme en cas d'erreur, au niveau LabVIEW, C, (et éventuellement matlab)
Ex:
Erreur ***pas assez de pixel***, monPrg C ... stoppe , ne modifie pas Traces.txt et affiche "err" dans stderr
Erreur ***exécutable manquant***, monPrg **LabVIEW** fait... (détails svp)
Erreur ***trop de pixels***, ignore les pixels excédentaires et continue
...

Documentation 3

- Description **succincte** de vos algorithmes
 - Programme C
 - Ex.
 - Parcourt toutes les lignes
 - Parcourt toutes les colonnes
 - ...
 - Optimisation: mon programme fait ...
 - Programme LabVIEW (recherche bord)
 - Ex. Parcourt toutes les lignes
 - Parcourt toutes les colonnes
 - ...
- Autres informations utiles, problèmes rencontrés où commentaires
 - Ex.
 - L'exécutable doit être recompilé sous Sonona (OSX 14.6)
 - ...