Présentation du projet :

Réalisation d'un encodeur absolu à l'aide d'une webcam

Association VOYAGER 3 Astronomie
- Espace culturel de la Garenne - 44530 Sévérac -

Association (Loi 1901) pour l'initiation et la pratique de l'astronomie,
Agrément Jeunesse & Education Populaire N°44-08-12

Commission technique de VOYAGER 3 Astronomie
Sébastien POIRIER

SOMMAIRE

Guide de conception

Programmation

I) Calcul préliminaire de la résolution théorique

II) Codage de l'information
- 2.1) Choix du codage
- 2.2) Principe de lecture du code par la webcam
- 2.3) Conversion en valeur numérique
- 2.4) Algorithme de génération du code Gray
- 2.5) Géométrie du fichier d'impression

III) Distorsions dues à l'optique de la webcam
- 3.1) Présentation
- 3.2) Principe de la correction
- 3.3) Algorithme de détection d'un point
- 3.4) Algorithme de détection de la grille
- 3.5) Algorithme de correction des distorsions

IV) Lecture du code Gray par la webcam
- 4.1) Détection des lignes guides
- 4.2) Inclinaison et échelle du code
- 4.3) Première estimation pour la lecture du code
- 4.4) Seconde estimation pour la lecture du code
- 4.5) Bits de poids faibles
- 4.6) Bits de poids forts
- 4.7) Moindres carrés sur un ensemble de bits
- 4.8) Détermination « haute précision » de la valeur lue
- 4.9) Détrompeur

V) Étalonnage du code Gray
- 5.1) Présentation
- 5.2) Principe
- 5.3) Erreur d'assemblage et conséquence sur la précision de l'étalonnage
- 5.4) Échantillon d'image du code Gray
- 5.5) Algorithme

Mécanique

VI) Effets de la dilatation
- 6.1) Présentation
- 6.2) Contrainte sur la fixation de la bande imprimée
- 6.3) Effet sur la lecture du code

VII) Effet de la rotation

Hardware

VIII) Webcams et contrôleur USB

I) Calcul préliminaire de la résolution théorique

L'idée de la réalisation est issue d'un réflexion sur la détermination du positionnement d'un télescope.

Une résolution de 640x480 est une valeur courante pour les webcams. Pour la suite on considérera cette résolution. La plage de réglage de la netteté de l'image de la webcam permet d'observer clairement un objet de l'ordre de 9 mm de largeur

Or $9 : 640 = 0{,}014$ à 10^{-3} près.

Ainsi, la résolution spatiale de la webcam dans ce cas est de 14 μm.

Afin de réaliser l'encodeur, La webcam sera positionnée de façon à observer des graduations sur un disque. Notre objectif étant de réaliser un encodeur avec le minimum de frais, l'impression sera effectuée sur une simple imprimante.

Compte tenu des dimensions du télescope, l'utilisation de 4 bandes de feuille A4 (297mm de longueur) est retenue.

$4 \times 297 / 3{,}14 = 378$ à l'unité prés

Le disque d'encodeur peut donc avoir un diamètre de 378 mm.

Nous utiliserons la valeur de 360mm afin de tenir compte des marges d'impression. Ainsi le périmètre du disque sera de 1131 mm.

Par ailleurs,

$1131 / 0{,}014 = 80785$ à l'unité près par défaut

or $2^{16} < 80785 < 2^{17}$

La résolution angulaire théorique de l'encodeur absolu sera donc de <u>16 bits</u>

Un encodeur absolu de 16 bits pour le prix d'une webcam présente un coût dérisoire par rapport à ce qui est proposé sur le marché.

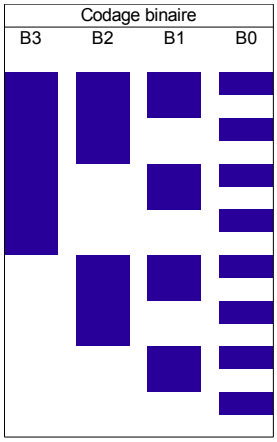
Naturellement, ce calcul est préliminaire Quelle résolution pouvons nous réellement atteindre avec cette méthode ?

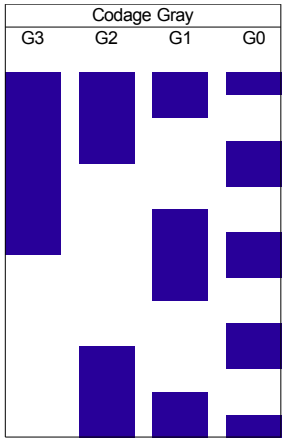
II) Codage de l'information

2.1) Choix du codage

Dans le cadre des encodeurs absolus du commerce, le choix se porte naturellement sur un codage binaire réfléchi – aussi nommé *code Gray* -. Ce codage permet d'éviter les erreurs de lecture. En effet, ce code est construit de telle façon à ce qu'un seul bit change d'état (0 ou 1) entre deux valeurs.

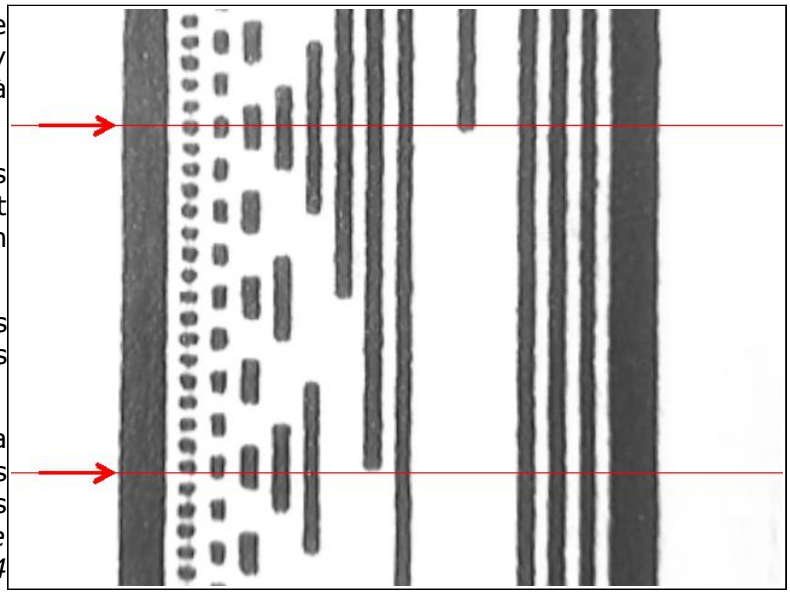
Nombre en base 10	Nombre en binaire				Nombre binaire réfléchi			
	B3	B2	B1	B0	G3	G2	G1	G0
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	1
3	0	0	1	1	0	0	1	0
4	0	1	0	0	0	1	1	0
5	0	1	0	1	0	1	1	1
6	0	1	1	0	0	1	0	1
7	0	1	1	1	0	1	0	0
8	1	0	0	0	1	1	0	0
9	1	0	0	1	1	1	0	1
10	1	0	1	0	1	1	1	1
11	1	0	1	1	1	1	1	0
12	1	1	0	0	1	0	1	0
13	1	1	0	1	1	0	1	1
14	1	1	1	0	1	0	0	1
15	1	1	1	1	1	0	0	0





2.2) Principe de lecture du code par la webcam

La webcam est orientée de telle façon à ce que le code Gray apparaisse comme sur l'image à droite.



Le code Gray est lu sur deux lignes de la webcam (ligne 96 et 384 soit un espacement de 288 pixel en verticale).

Les deux valeurs obtenues permettent d'éliminer les éventuelles erreurs de lecture.

Nous verrons par la suite que la double lecture présente d'autres avantages. Il faudra toutefois préciser ce que l'on entend par *lire suivant les lignes n°96 et n°384* pour un code incliné par rapport à la webcam.

La webcam est placée suffisamment près de la bande pour obtenir une bonne résolution spatiale. Nous verrons par la suite qu'un compromis raisonnable se situe telle que la largeur du code Gray occupe environ 70% de la largeur de la webcam. L'image sera en niveau de gris.

A cette échelle un rectangle du bit de poids faible doit être suffisamment grand pour garantir une bonne lecture. Une dimension raisonnable est choisie de l'ordre de 4 pixels de largeur suivant la perpendiculaire aux lignes de lecture.

Le code imprimé sera donc un code Gray 14 bits.

Finalement, le principe de lecture introduira deux lignes de lecture secondaire afin de réaliser un détrompeur efficace l'ordre des erreurs de lecture.

2.3) Conversion en valeur numérique

$(G_i)_{0\le i<n}$ est une suite de 0 et 1 telle que G_0 est le bit de poids faible

$G_0 G_1 G_2 G_3 \ldots G_{n-1}$ forme un nombre en binaire réfléchi

On note $B_0 B_1 B_2 B_3 \ldots B_{n-1}$ forme le nombre équivalant en binaire

La valeur décimale est donnée par la variable ***v*** et vérifiant:

$$\begin{cases} B_{n-1} = G_{n-1} \\ B_i = G_i \oplus B_{i+1} \quad \text{pour } 0 \le i \le n-2 \\ v = \sum_{i=0}^{n-1} B_i \, 2^i \end{cases}$$

La procédure en C est donnée ci-contre.

```
int GrayCodeToDecimal(int *bit, int n)
{
        int i,v,poids;

        poids=(int)(pow(2,n-1));
        v = bit[n-1]*poids;

        for(i=n-2; i>=0; i--)
        {
             poids /=2;
             bit[i] = (bit[i] + bit[i+1])%2;
             v     += bit[i]*poids;
        }

        return(v);
}
```

2.4) Algorithme de génération du code Gray

Le code Gray de l'encodeur sera imprimé. Il nous est donc nécessaire de réaliser un programme créant le fichier à imprimer.

La génération d'un code gray consiste à générer des lignes de bits vérifiant l'algorithme suivant:

On note ***e*** la largeur d'un bit de poids faible

Pour $0 \le b < N$ avec ***N*** nombre de bit du code

- Pour n=0, $x_{0,b} = 0$ $\quad x'_{0,b} = e\,2^b$
- Si b<N-2

  Pour $1 \le n \le 2^{N-b-3}$, $x_{n,b} = (4n-1)\,e\,2^b$

  $x'_{n,b} = (4n+1)\,e\,2^b$

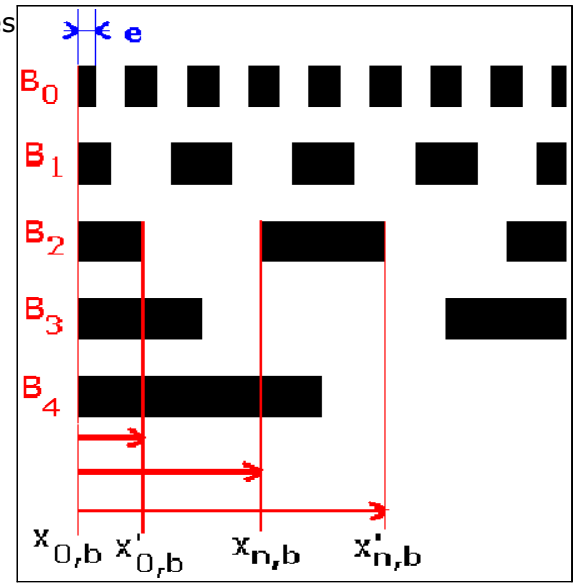
  Pour $n = 2^{N-b-3}+1$, $x_{n,b} = (4n-1)\,e\,2^b$

  $x'_{n,b} = 4n\,e\,2^b$

- Si b=N-2

  Pour $n = 1$, $x_{n,b} = (4n-1)\,e\,2^b$

  $x'_{n,b} = 4n\,e\,2^b$



Notations pour le cas N=5, b=3, n=1

2.5) Géométrie du fichier d'impression

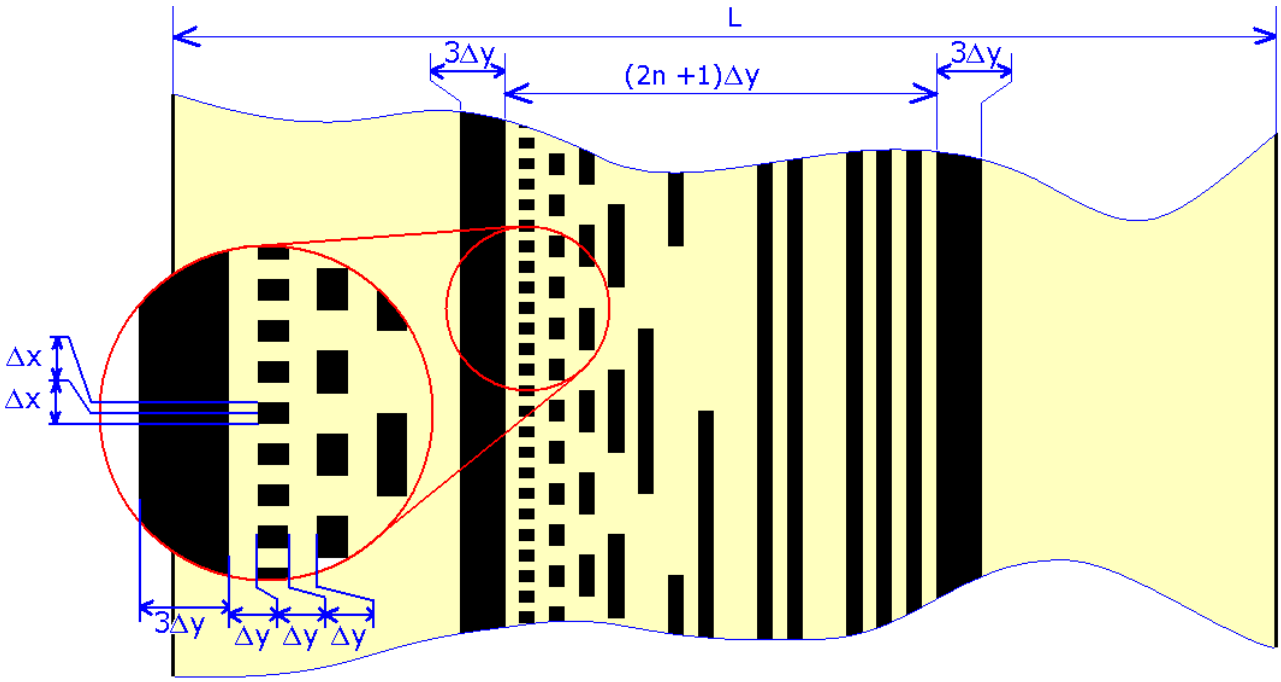
2.5.1) Remarque générale sur l'impression du code

Le programme génère un fichier au format .IA (Adobe Illustrator), présentant l'avantage d'être un format vectoriel.

La résolution de la webcam étant de l'ordre de 14µm cela correspond en résolution d'imprimante à environ 2400 dpi (25,4/2400=1,05×10^{-2} mm soit 11 µm). Dans la pratique une impression à 600 dpi est suffisante.

2.5.2) Paramétrage

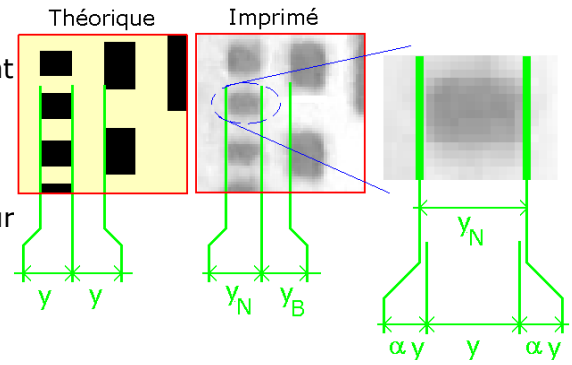
Nombre de bits de l'encodeur	: n = 14	
Longueur du bit faible	: $\Delta x = 0,2$ en 72 dpi	soit $\Delta x \approx 0,07$ mm
Largeur d'un bit	: $\Delta y = 0,58$ en 72 dpi	soit $\Delta y \approx 0,204$ mm
Largeur d'une bande	: L = 43 en 72 dpi	soit L ≈ 1,517 cm



2.5.3) Élargissement du code à l'impression

Lors de l'impression, les bandes noires ont tendance à s'élargir en empiétant sur l'espace des bandes blanches. On définit le facteur d'élargissement relatif α par la relation:

$$\alpha = \frac{1}{2}\frac{y_N}{y} \qquad \text{ainsi} \quad y_N = y(1 + 2\alpha)$$



De plus, la géométrie du code n'étant globalement pas modifiée nous avons : $y_B + y_N = y + y$

Ainsi, $y_B = y(1 - 2\alpha)$

De cette façon, il suffit de mesurer $y_B + y_N$ pour trouver la valeur de α. Il suffit d'utiliser la relation :

$$\alpha = \frac{1}{2}\frac{y_N - y_B}{y_N + y_B}$$

III) Distorsions dues à l'optique de la webcam

3.1) Présentation

Les webcams possèdent une optique permettant de focaliser l'image sur le plan du capteur photosensible.

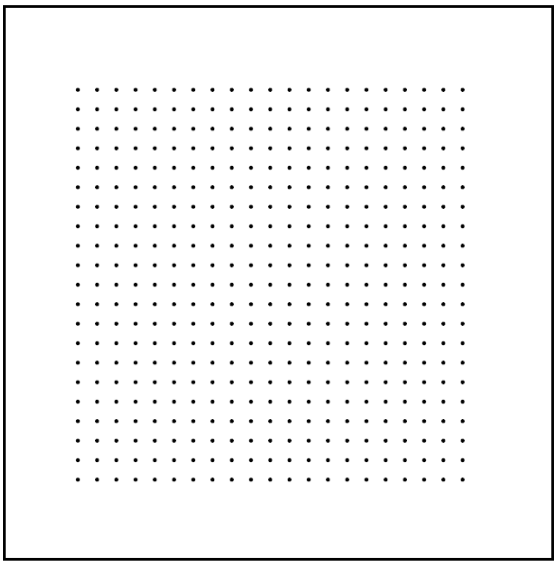
Toute optique présente l'inconvénient de déformer les images, et ceci d'autant plus que l'on s'éloigne de l'axe optique du système.

Afin de réaliser de mesure précise, nous devons nous affranchir de ces distorsions du champ. L'étalonnage des graduations du code Gray peut permettre une telle correction. Toutefois la méthode d'étalonnage que nous présenterons par la suite repose sur l'alignement d'image s présentant peut ou pas de distorsion.

Ainsi, il faut s'affranchir un maximum des distorsions avant l'étalonnage.

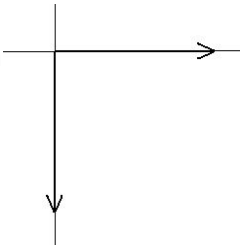
La méthode retenue est l'utilisation d'une mire de calibration. La méthode est classique, mais une mire de calibration, permettant une correction optimale est couteuse. Dans notre cas, la correction des distorsions n'a pas besoin d'être optimale, mais juste raisonnable.

Notre mire de calibration sera donc une simple grille imprimée à l'aide d'une imprimante. Les tests réalisés montrent que cela est suffisant.



La grille est générée par un programme en créant un fichier .IA (Adobe Illustrator).
La grille est constituée de 21x21 points placés sur un carré de 1,5 cm de côté.

Pour la suite les coordonnées des points de la grilles seront celles d'un repère où l'axe des l'abscisse est horizontale, orienté de la gauche vers la droite et l'axe des ordonnée est vertical, orienté du haut vers le bas. C'est le choix retenu pour les coordonnées des pixels d'une image.



Le fichier source est : ***Generation_Grille.c***

3.2) Principe de la correction

La grille est placée sur le disque portant le code gray.

La webcam est supposé fixée par rapport à ce disque. On suppose que le disque comporte une excentricité négligeable (ce défaut entrainerait que le code Gray soit plus ou moins proche de l'optique de la webcam) introduisant un effet de flou.

La webcam fait l'acquisition d'une image de la grille. Cette nous apparaît déformé (image 1)
Un algorithme repère automatiquement les points de la grille (image 2).
Un seconde algorithme mémorise les défauts et les corriges (image 3).

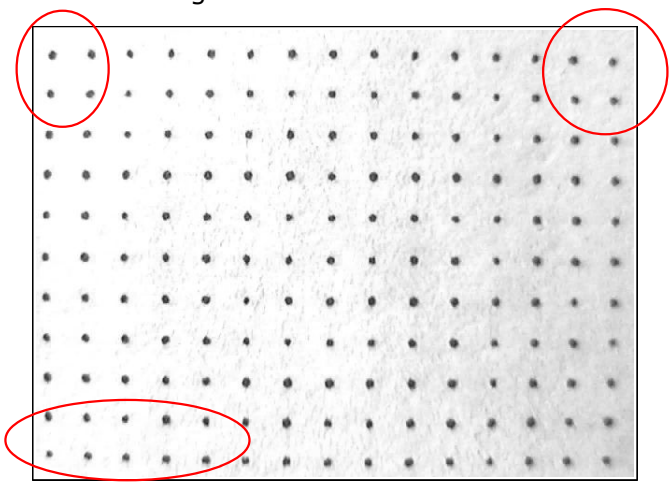


Image 1: Image avec distorsions

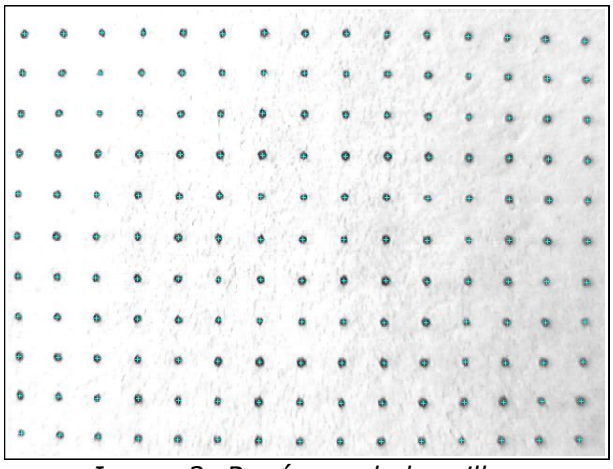


Image 2: Repérage de la grille

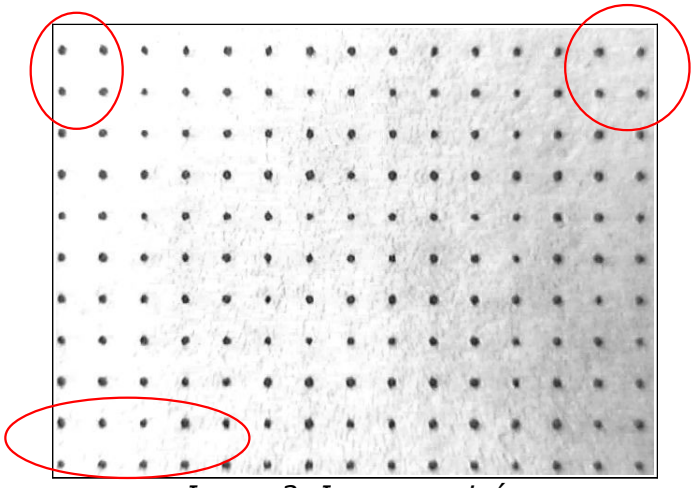
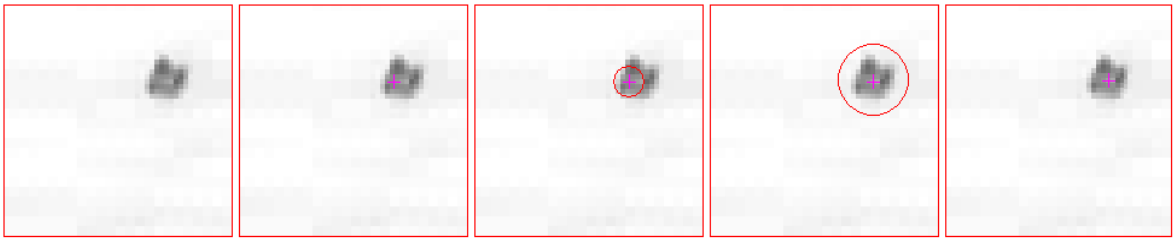


Image 3: Image corrigée

3.3) Algorithme de détection d'un point

La méthode de détection d'un point peut se décrire suivant 5 étapes:



Etape 1 Etape 2 Etape 3 Etape 4 Etape 5

Préalablement à la détection des points, on réalise un traitement de l'image en déterminant une image moyenne en subdivisant celle-ci en 100 zones. Chaque zone à la couleur correspondant à la moyenne des couleurs des pixels qu'elle contient. On soustrait l'image de la webcam par cette image moyenne, permettant ainsi de retirer le « continu ». Nous divisons l'image obtenu de nouveau par l'image moyenne afin de compenser les effets de variations de lumière (gradient, ombre...). Finalement, les niveaux de l'image ainsi obtenu sont ajustés afin de couvrir la plus grand dynamique.

Pour la détection à proprement parler, on se donne une zone de l'image carrée dont les côtés mesure $2R_{max}$

On estime la position du point en déterminant les moyennes des coordonnées des points de la zone image en pondérant en fonction de la couleur des pixels. Pour la fonction de pondération on considère le cube de la valeur du pixel. L'expérience montre que ce choix donne de bons résultats.

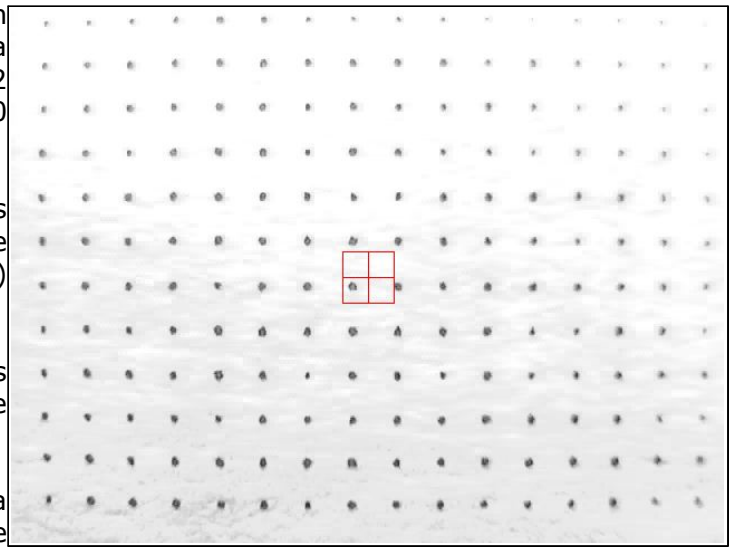
$$\begin{cases} \bar{x} = P_T^{-1} \sum_i \sum_j P_{ij}^3 \, x_{ij} \qquad \text{avec} \quad P_T = \sum_i \sum_j P_{ij}^3 \\ \bar{y} = P_T^{-1} \sum_i \sum_j P_{ij}^3 \, y_{ij} \end{cases}$$

Finalement, les coordonnées retenues pour le point de la grille correspondent à ($E(\bar{x}+0,5)$; $E(\bar{y}+0,5)$) .

3.4) Algorithme de détection de la grille

3.4.1) Centre de la grille

Dans un premier temps on repère un point à proximité du centre de l'image cette recherche se fait en aveugle.



Compte tenu du réglage de la résolution spatiale (voir §2.2) et de l'échelle de la grille (voir §3.1), la distance entre 2 points de la grille est de l'ordre de 50 pixels.

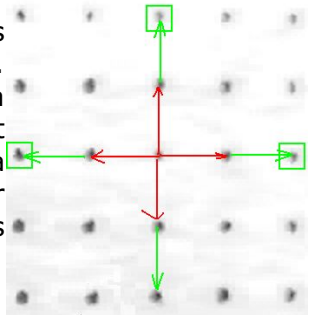
On recherche donc le point le plus proche du centre de l'image (coordonnées du centre (320;240)) dans 4 carrés de 25 pixels par 25 pixels.

L'algorithme de détection des points présenté dans le paragraphe §3.3 donne des coordonnées pour chaque carrés.

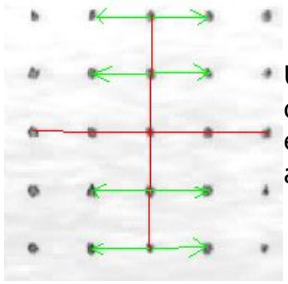
Le point recherché correspond à la solution ayant le plus grand contraste parmi les 4 obtenus.

3.4.2) Détection de l'ensemble de la grille

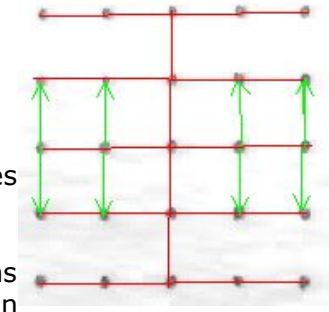
On suppose qu'au moins un point a été détecté, ne serait-ce que celui correspondant au centre de la grille du paragraphe précédent.



A partir de ce point initiale, nous allons rechercher l'ensemble des points situé suivant un axe horizontale et un axe verticale de la grille. Le vecteur servant à déterminer la position du point centrale de la prochaine zone de recherche a, par défaut, 25 pixels et orientée suivant les axes naturelles de l'image (horizontal ou vertical). Sa norme et sa direction évoluent suivant la position réelle du point détecté avec pour hypothèse que la grille est peut déformée localement, ce qui est le cas pour notre problème.



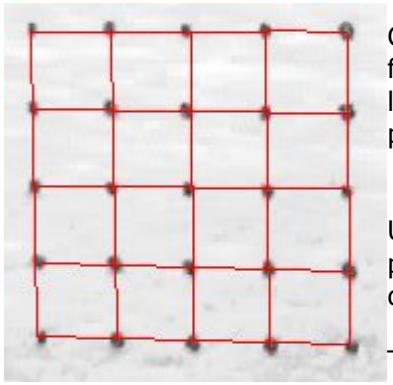
Une fois que l'ensemble des points de la grille de l'image se trouvant sur ce premier axe verticale, une détection des points des axes horizontaux est lancée à partir de chaque point de ce premier axe verticale.



Finalement, nous suivons la même procédure pour les points des axes verticaux.

Pour chaque point détecté, nous mémorisons ses coordonnées dans l'image ***(x,y)*** ainsi que ses coordonnées au sein de la grille ***(i,j)***, en

prenant comme point d'origine (*coordonnées (0;0) dans la grille)* le point à proximité du centre de l'image.



Cette procédure conduit à potentiellement détecter un point deux fois. Elle permet aussi d'éviter manquer un point de la grille dans le cas d'une grille inclinée ou déformée, où un axe apparaît partiellement ou de façon discontinue.

Une procédure d'identification croisée *(Cross identification)* permet de supprimer un point redondant. En effet chaque point détecté peut avoir des coordonnées sur l'image différente

Toutefois, si deux points P_1 et P_2 ont des coordonnées identiques *(i,j)* sur la grille, il s'agit en réalité d'un seul et unique point. La procédure d'identification croisée les associés et attribue à ce point effectif, la moyenne des deux coordonnées (x_1, y_1) et (x_2,y_2) pondérées par les poids $P_{T,1}$ et $P_{T,2}$ (voir §3.3) associée à chacun des points.

Le point effectif a donc pour coordonnées sur la grille: (i,j)
et pour coordonnées sur l'image (x_e, y_e) donc par :

$$x_e = (P_{T,1} + P_{T,2})^{-1} (P_{T,1} \ x_1 + P_{T,2} \ x_2)$$

$$y_e = (P_{T,1} + P_{T,2})^{-1} (P_{T,1} \ y_1 + P_{T,2} \ y_2)$$

Finalement l'ensemble des points détecté est sauvegardé dans un tableau ou l'ordre des points est ordonnée de la façon suivante:

- Les premiers points de la liste sont ceux ayant la plus petite abscisse sur la grille.
- L'ordre de ces points est celui de l'ordre croissant suivant les ordonnées sur les grille.

- on itère jusqu'à ce que l'ensemble des points détectés soit classé.

3.5) Algorithme de correction des distorsions

3.5.1) Présentation

La procédure des détection de la grille (§3.4) permet d'associer deux types de coordonnées à chaque point observé sur la webcams :

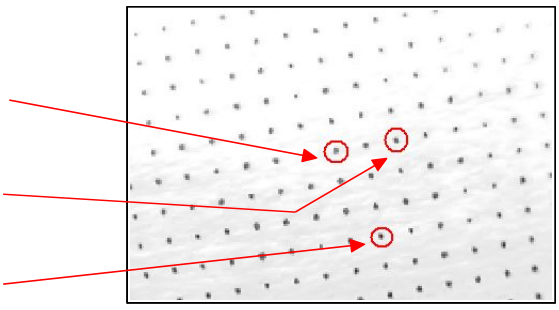
- Des coordonnées correspondent à la position du point sur l'image
- Des coordonnées correspondent à la position du point sur la grille (voir §3.1 et §3.4.2 pour le choix du repère)

Exemples :

Coordonnées sur l'image (318; 242)
Coordonnées sur la grille (0,0)

Coordonnées sur l'image (410;229)
Coordonnées sur la grille (2;0)

Coordonnées sur l'image (382;386)
Coordonnées sur la grille (3;1)



Les coordonnées sur l'image sont des coordonnées que l'on peut qualifier de réel. C'est celles obtenues en observant la grille via la webcam. Les coordonnées sur la grille sont des coordonnées que l'on peut qualifier d'idéales.

Le principe de la correction s'effectue en 3 phases:

1) Détermination de la rotation et du changement de l'image avant et après la correction.
2) Rotation et dilatation appliquées aux coordonnées de la grille des points en fonction des valeurs obtenues à l'étape 2.
3) Application du modèle de correction à l'ensemble de l'image.

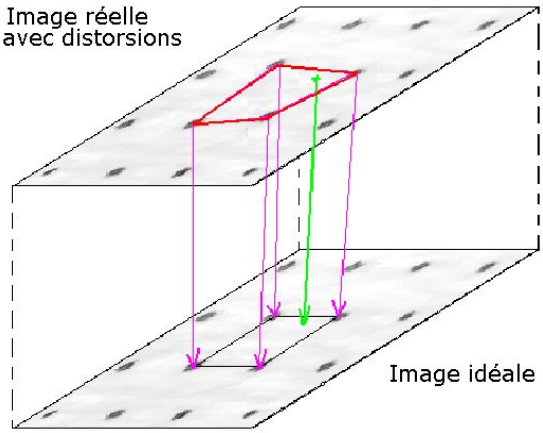
La condition d'arrêt s'effectue lorsque le changement d'échelle et de la rotation de l'image sont négligeable. Dans la majorité des cas, 2 itérations sont suffisantes.

L'association du double jeu de coordonnées pour les pixels correspondants au centre d'un point a été effectuée lors de la détection de la grille.

Sur la figure ci-contre l'association des coordonnées est représentée par les flèches mauves ayant pour origine un des sommets du quadrilatère représenté en rouge sur l'image réelle.

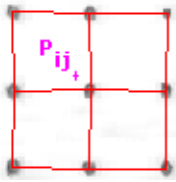
Toutefois, pour les pixels situés à l'intérieur du quadrilatère l'association de coordonnées n'est pas explicitée.

La correction retenue est réalisée par un interpolation bi-linaire (ou extrapolation le cas échéant pour des pixels des bords de l'image)



L'image réelle est découpée suivant le pavage imposé par le point de la grille. Ainsi chaque pixel de l'image est à l'intérieur d'un quadrilatère ayant pour sommet des points de la grille.

Pour un pixel P_{ij} de l'image réelle donnée, il est nécessaire de connaître le quadrilatère le contenant. Cela revient à trouver les points de la grille les plus proches du pixel P_{ij}.



Afin de tenir compte des différents cas, il faut rechercher les 9 points les plus proches du pixel P_{ij} afin de définir 4 quadrilatères dont un contient nécessairement P_{ij} .

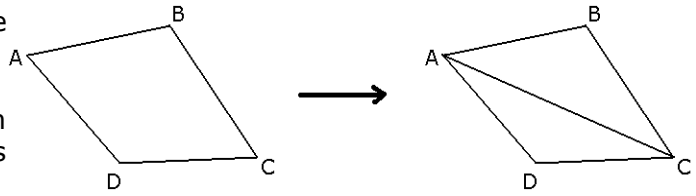
La distance entre les points de la grille et du pixel P_{ij} .est calculée à l'aide de la distance euclidienne classique appliquée sur les coordonnées de l'image.

Il faut toutefois créer une procédure permettant de tester si un point se trouve à l'intérieur d'un quadrilatère. Pour simplifier le problème, on ne considére que le cas de quadrilatère convexe, ce qui correspond au cas de faibles distorsions comme c'est le cas pour l'optique d'une webcam.

3.5.2) Point à l'intérieur d'un quadrilatère convexe

- Soit M un point donc on connait les coordonnées (x_M; y_M) dans un repère quelconque.

A, B, C et D sont 4 points définissant un quadrilatère convexe dont on connait les coordonnées dans le même repère.



Si ABCD est convexe alors la diagonales [AC] est à l'intérieur de ce quadrilatère. Ainsi ACD et ACB, sont deux triangles pavant ABCD.

- Si M est un point de la surface de ABCD alors soit M est un point de la surface de ACD, soit M est un point de la surface de A de ACB. Vérifier qu'un point à l'intérieur de ce quadrilatère convexe revient à vérifier si ce point est à l'intérieur de l'un des deux triangles.

- On considère le triangle ABC (Pour le car ADC, il suffira de substituer le point B par D)

Un point du plan peut être défini de façon unique comme étant un barycentre de 3 points avec la données des 3 coefficients de pondération.

Ainsi soit M le barycentre de (A ; α) (B ; β) (C ; γ) avec $\alpha + \beta + \gamma = 1$

Trouver les valeurs de α, β et γ revient à résoudre le système

D'où
$$\begin{cases} \alpha\, x_A + \beta\, x_B + \gamma\, x_C = x_M \\ \alpha\, y_A + \beta\, y_B + \gamma\, y_C = y_M \\ \alpha + \beta + \gamma = 1 \end{cases}$$

Ce système est équivalant à l'égalité matricielle :
$$\begin{pmatrix} x_A & x_B & x_C \\ y_A & y_B & y_C \\ 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix} = \begin{pmatrix} x_M \\ y_M \\ 1 \end{pmatrix}$$

Ce système admet donc une solution pour $d \neq 0$, avec $d = (x_B - x_A)(y_C - y_A) - (y_B - y_A)(x_C - x_A)$

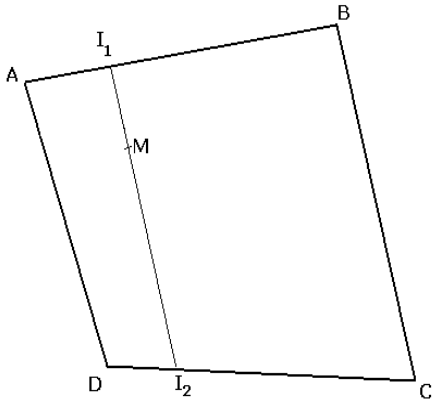
Donnée par :
$$\begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix} = d^{-1} \begin{pmatrix} y_B - y_C & x_C - x_B & x_B y_C - x_C y_B \\ y_C - y_A & x_A - x_C & x_C y_A - x_A y_C \\ y_A - y_B & x_B - x_A & x_A y_B - y_A x_B \end{pmatrix} \begin{pmatrix} x_M \\ y_M \\ 1 \end{pmatrix}$$

d'où
$$\begin{cases} \alpha = [\,(y_B - y_C)\, x_M + (x_C - x_B)\, y_M + x_B y_C - x_C y_B\,]/d \\ \beta = [\,(y_C - y_A)\, x_M + (x_A - x_C)\, y_M + x_C y_A - x_A y_C\,]/d \\ \gamma = [\,(y_A - y_B)\, x_M + (x_B - x_A)\, y_M + x_A y_B - y_A x_B\,]/d \end{cases}$$

- Ainsi, M est à l'intérieur du triangle ABC si et seulement si: $0 \leq \alpha \leq 1$, $0 \leq \beta \leq 1$ et $0 \leq \gamma \leq 1$

3.5.3) Correction bi-linéaire

*Dans cette partie les expressions en gras telles que que **AB** définissent des vecteurs.*



A, B, C et D sont 4 points définissant un quadrilatère convexe dont on connait les coordonnées dans le repère de l'image.

Soit a et b deux réelles tels que $0 \leq \alpha \leq 1$ et $0 \leq \beta \leq 1$.

On définit le point I_1 par : $\mathbf{AI_1} = \alpha\,\mathbf{AB}$
On définit le point I_2 par : $\mathbf{DI_2} = \alpha\,\mathbf{DC}$
On définit le point M par : $\mathbf{I_1M} = \beta\,\mathbf{I_1I_2}$

*Remarque: On peut de la même façon définir M en changeant les rôles de **AB** et **AD**. Les images ayant peu de distorsion localement, en d'autre terme ABCD étant proche d'un carré, cela change peu de chose quand à la position du point M ainsi définie. Par ailleurs, une fois le choix défini la position de M est bien déterminé de façon unique.*

Ainsi,

$$\mathbf{I_1A} + \mathbf{AM} = \beta\,\mathbf{I_1A} + \beta\,\mathbf{AI_2}$$

$$\mathbf{AM} = (\beta - 1)\;\mathbf{I_1A} + \beta\,\mathbf{AI_2}$$

$$\mathbf{AM} = (1 - \beta)\,\alpha\,\mathbf{AB} + \beta\,\mathbf{AD} + \beta\,\mathbf{DI_2}$$

$$\mathbf{AM} = (1 - \beta)\,\alpha\,\mathbf{AB} + \beta\,\mathbf{AD} + \alpha\,\beta\,\mathbf{DC}$$

$$\mathbf{AM} = \alpha\,\mathbf{AB} + \beta\,\mathbf{AD} + \alpha\beta\,(\,\mathbf{DC + BA}\,)$$

Le point M est défini de façon unique par la donnée de $(\alpha\,;\beta)$. On peut remarquer que dans le cas d'un carré ces coordonnées correspondent à celles attribuées au point M dans un repère orthonormal.

- Si un point M (x_M; y_M) est à l'intérieur d'un quadrilatère ABCD, qu'elles sont les valeurs de α et β associées à ce point ?

Ce problème consiste à résoudre :

$$\mathbf{AM} = \alpha\,\mathbf{AB} + \beta\,\mathbf{AD} + \alpha\beta\,(\,\mathbf{DC - AB}\,) \text{ avec } \alpha \text{ et } \beta \text{ pour inconnues}$$

Ce qui équivaut à résoudre :

$$\begin{cases} x_M - x_A = \alpha\,(x_B - x_A) + \beta\,(x_D - x_A) + \alpha\beta\,(x_A - x_B + x_C - x_D) \\ y_M - x_A = \alpha\,(y_B - y_A) + \beta\,(y_D - y_A) + \alpha\beta\,(y_A - y_B + y_C - y_D) \end{cases}$$

On note

$a_{00} = x_A - x_A$ $\quad$ $a_{10} = y_A - y_A$
$a_{01} = x_B - x_A$ $\quad$ $a_{11} = y_A - y_A$
$a_{02} = x_D - x_A$ $\quad$ $a_{12} = y_A - y_A$
$a_{03} = x_A - x_B + x_C - x_D$ $\quad$ $a_{13} = y_A - y_B + y_C - y_D$

Le système devient ainsi :

$$\begin{cases} a_{00} + \alpha\,a_{01} + \beta\,a_{02} + \alpha\beta\,a_{03} = 0 \\ a_{10} + \alpha\,a_{11} + \beta\,a_{12} + \alpha\beta\,a_{13} = 0 \end{cases}$$

$$\begin{cases} a_{00} + \beta\, a_{02} + \alpha\,(a_{01} + \beta\, a_{03}) = 0 \\ a_{10} + \beta\, a_{12} + \alpha\,(a_{11} + \beta\, a_{13}) = 0 \end{cases}$$

d'où $(a_{00} + \beta\, a_{02})(a_{11} + \beta\, a_{13}) - (a_{10} + \beta\, a_{12})(a_{01} + \beta\, a_{03}) = 0$

$(a_{02}\, a_{13} - a_{12}\, a_{03})\,\beta^2 + (a_{00}\, a_{13} + a_{02}\, a_{11} - a_{10}\, a_{03} - a_{12}\, a_{01})\,\beta + a_{00}\, a_{11} - a_{10}\, a_{01} = 0$

On note $A = a_{02}\, a_{13} - a_{12}\, a_{03}$ $\quad B = a_{00}\, a_{13} + a_{02}\, a_{11} - a_{10}\, a_{03} - a_{12}\, a_{01}$ $\quad C = a_{00}\, a_{11} - a_{10}\, a_{01}$

d'où $A\,\beta^2 + B\,\beta + C = 0$

on note $\Delta = B^2 - 4AC$ Cette équation admet au moins une solution réelle, M étant à l'intérieur du quadrilatère.

donc $\Delta \geq 0$

Si $A \neq 0$ $\quad \beta = (-B - \sqrt{\Delta})(2A)^{-1}$ ou $\beta = (-B + \sqrt{\Delta})(2A)^{-1}$

Si $A = 0$ $\quad$ Si $B \neq 0$ alors $\beta = -C\, B^{-1}$

Si $B = 0$ alors on ne gère pas ce cas puisque M étant à l'intérieur du quadrilatère il existe une unique solution. Ce cas est donc impossible.

On obtient α de la façon suivante:

Si $(a_{01} + \beta\, a_{03}) \neq 0$ $\quad$ alors $\alpha = -(a_{00} + \beta\, a_{02})(a_{01} + \beta\, a_{03})^{-1}$

Si $(a_{11} + \beta\, a_{13}) \neq 0$ $\quad$ alors $\alpha = -(a_{10} + \beta\, a_{12})(a_{11} + \beta\, a_{13})^{-1}$

M étant à l'intérieur du quadrilatère il existe une unique solution, α est définie dans l'un des deux cas précédent.

β admettant deux valeurs possibles on obtient deux couples de solutions vérifiant le système. On note ces couples $(\alpha_1;\beta_1)$ et $(\alpha_2;\beta_2)$. La solution recherchée est telle que $0 \leq \alpha_i \leq 1$ et $0 \leq \beta_i \leq 1$. M étant à l'intérieur du quadrilatère, la solution existe. Soit i=1 soit i=2.

- Finalement afin de réaliser la correction, on appliquant

$$x_{M'} = x_{A'} + \alpha\,(x_{B'} - x_{A'}) + \beta\,(x_{D'} - x_{A'}) + \alpha\beta\,\underbrace{(x_{A'} - x_{B'} + x_{C'} - x_{D'})}_{\mathbf{0}}$$

$$y_{M'} = x_{A'} + \alpha\,(y_{B'} - y_{A'}) + \beta\,(y_{D'} - y_{A'}) + \alpha\beta\,\underbrace{(y_{A'} - y_{B'} + y_{C'} - y_{D'})}_{\mathbf{0}}$$

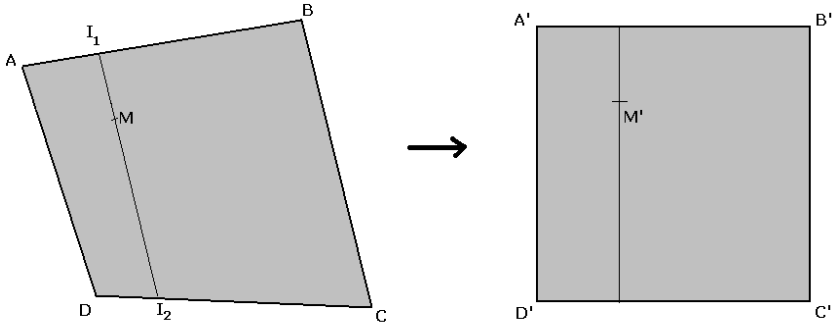


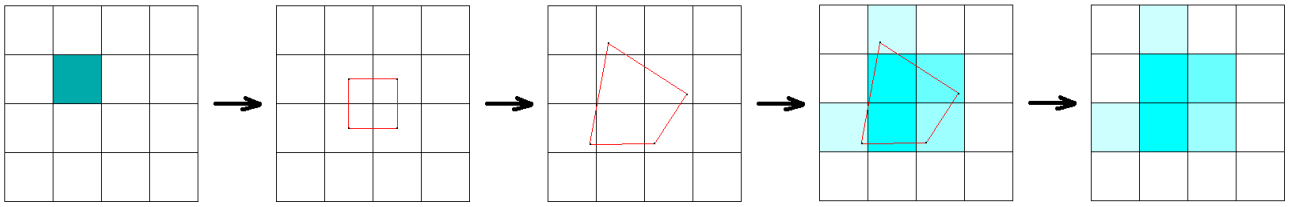
Image avec distorsions $\qquad$ Image corrigé

3.5.4) Gestion de la non-bijectivité de la transformation discrétisée

Dans le paragraphe précédent nous avons considéré une correction bilinéaire avec des variables réelles. Compte tenu du pavage des quadrilatères et la transformation employée à l'intérieur de chaque quadrilatère, la transformation considérée est bijective.

Toutefois, une image de webcam est un ensemble discret, les pixels état les surfaces élémentaires. Un pixel de l'image d'origine ne se transforme forcément en un pixel sur l'image corrigée. Ainsi pour le cas discret, la transformation n'est pas bijective.

On peut schématiser ainsi la modification de la transformation afin de tenir compte de la nature discrète des images:



Étape 1 Étape 2 Étape 3 Étape 4 Étape 5

On considère 4 pixels adjacents (1), On applique la transformation du §3.5.3 aux coordonnés de ces pixels (2 et 3). Les coordonnées des 4 points obtenus sont des valeurs a priori réelles (3). Afin de tenir compte de cet effet, on réparti la couleur initiale sur toute la zone obtenue (4). Chaque pixel concerné reçoit une faction de la couleur initiale (5). La somme des fractions vaut 1. La valeur de la fraction est déterminée à partir les coordonnées réelles du point et non de la valeur de la couleur de ce dernier.

L'effet de la non-bijectivité est minimisé par une répartition des couleurs.

Pour chaque pixel de l'image d'arrivée, on mémorise les coordonnées des pixels qui ont contribués à sa couleur ainsi que les coefficients correspondant à la fraction de couleur apporté par les différents pixels. La fraction ne dépendant pas de la couleur des pixels, le modèle ne dépend que de la distorsion de la webcam. Ainsi,Il suffit, pour chaque pixel de l'image d'arrivée,de mémoriser la liste des pixels de l'image d'origine et les fractions associées.

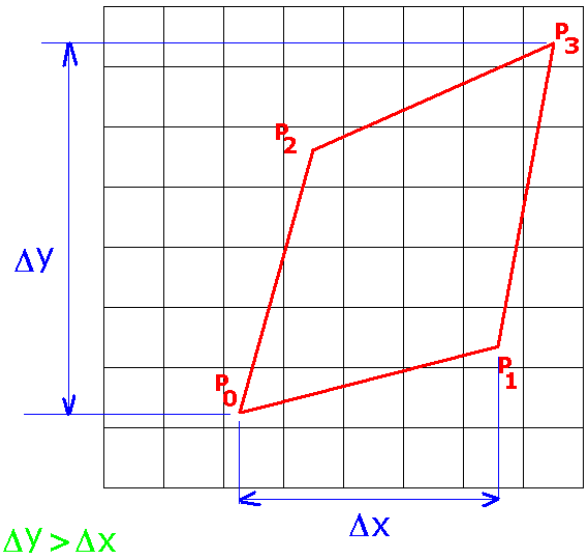
3.5.5) Calcul de la fraction de couleur à répartir sur chaque pixel

Cette partie n'est utilisée que lorsque l'image d'un pixel est répartie sur plus de 1 pixels, c'est à dire que la différence la plus grande des abscisses des sommets du quadrilatère est supérieur à 2 ou que la différence la plus grandes des ordonnées des sommets du quadrilatère est supérieur à 2.

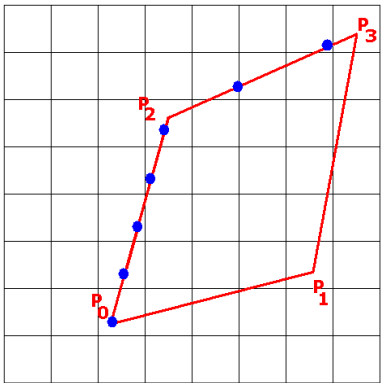
On considère un quadrilatère non convexe. Quitte à réordonner les points et à permuter les rôles des abscisses est des ordonnées, on peut toujours se ramener au cas où:

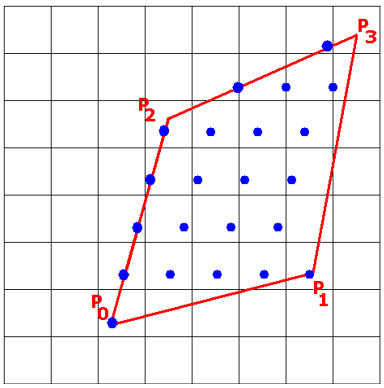
- Le point P_0 est le point de plus petite ordonnée parmi les 4 sommets
- Le point P_3 est le point de plus grande ordonnée parmi les 4 sommets
- Le point P_1 a un abscisse plus petite que P_2
- la différence la plus grande des abscisses des points et plus petite que la différence la plus grande des ordonnées des points

De telles conditions sont vérifiées par la figure ci-contre.

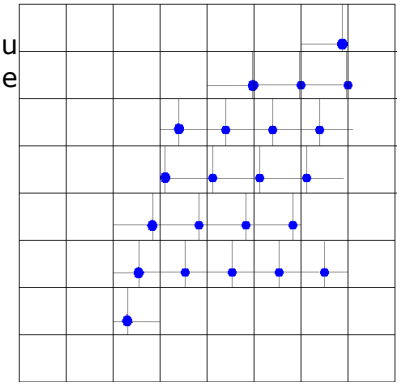


On repère les coordonnées contenues dans le quadrilatère par pas unitaire suivant l'axe des abscisses et des ordonnées en partant de P_0 comme point d'origine. A chaque pas en ordonnées, la première abscisse (valeur la plus petite) est choisie pour que le point associé à cette abscisse se trouve sur un côté du quadrilatère.

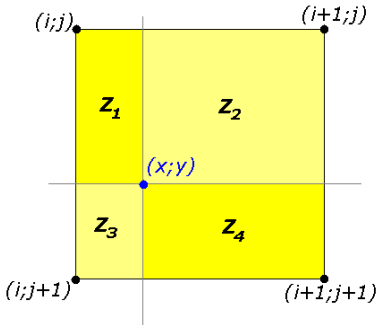




Chaque points ainsi définie est un donc un point de la surface du quadrilatère. Chacun d'entre eux définisse 4 partie d'un pixel de l'image, comme représenté sur la figure ci-contre.



Pour chaque point on peut donc définir 4 aires que l'on note Z_1, Z_2, Z_3 et Z_4.



On peut remarquer que $Z_1 + Z_2 + Z_3 + Z_4 = 1$

et $Z_1 = (x-i)(y-j)$ $Z_2 = (i+1-x)(y-j)$

$Z_3 = (x-i)(j+1-y)$ $Z_1 = (i+1-x)(j+1-y)$

On note $\boldsymbol{C_p}$ la couleur du pixel $\boldsymbol{p}$ obtenue par transformation présentée dans la paragraphe §3.5.3 et §3.5.4.

Le fraction de couleur sont prisent comme étant Z_1, Z_2, Z_3 et Z_4.

Ainsi la contribution de pixel $\boldsymbol{p}$ à la couleur des pixels $p_{(i;j)}$, $p_{(i+1;j)}$, $p_{(i;j+1)}$ et $p_{(i+1;j+1)}$ est donnée par:

$c_{i,j} = Z_1 C_p$ avec $c_{i,j}$ la contribution à la couleur du pixel $p_{(i;j)}$

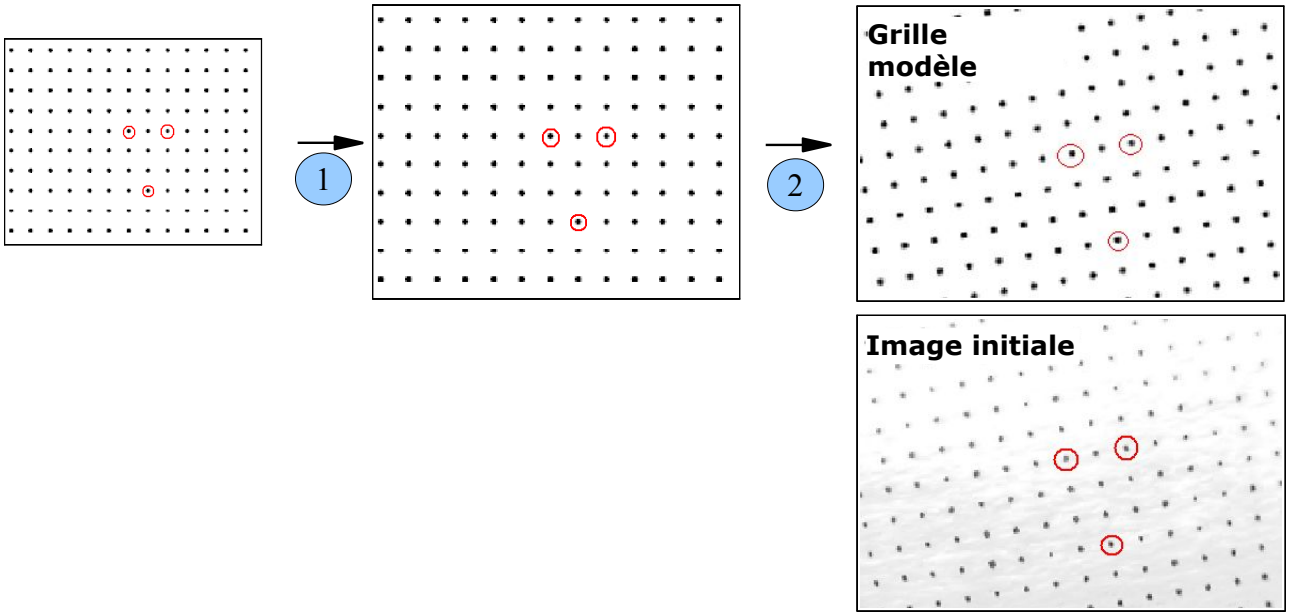
$c_{i+1,j} = Z_2 C_p$ $c_{i+1,j}$ la contribution à la couleur du pixel $p_{(i+1;j)}$

$c_{i,j+1} = Z_3 C_p$ $c_{i,j+1}$ la contribution à la couleur du pixel $p_{(i;j+1)}$

$c_{i+1,j+1} = Z_4 C_p$ $c_{i+1,j+1}$ la contribution à la couleur du pixel $p_{(i+1;j+1)}$

3.5.6) Rotation et dilatation de la grille

Afin de conserver la même échelle (1) et la même orientation du champ de l'image avant et après la correction, les coordonnées de la grille doivent être adaptées.



Afin de déterminer l'échelle et l'angle de rotation, on procède de façon itérative:

- Initialement, on utilise les coordonnées naturelles de la grille (celles associées aux points lors de la détection de la grille (§3.4)
- On a appliqué le modèle de correction au pixel centrale, P_1, de l'image de coordonnées (x_0; y_0) et a un autre pixel, P_2, proche du centre de l'image (par exemple de coordonnées (x_0+10; y_0).

  La distance entre les deux points sur l'image d'origine est donc de 10 pixels (x_0+10-x_0=10)

  On note ***e*** le rapport de la distance entre les points pixels sur l'image après la correction et la distances entre les deux pixels avant la correction.

  De même, on note θ note l'angle entre la direction de la droite passant par les deux points après et avant la correction

  Finalement, on note (x_1;y_1) et (x_2;y_2) les coordonnées des points P_1 et P_2 après la correction.

  Soit $d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$

  Ainsi, $e = \frac{d}{10}$ $\cos\theta = \frac{(x_2 - x_1)}{e}$ et $\sin\theta = \frac{(y_2 - y_1)}{e}$

  On applique la correction suivante aux coordonnées de la grille pour chaque point détecté lors de la procédure de détection de la grille:

  On note (x;y) les coordonnées initiales et (x'; y') les coordonnées finales du point.

  Soit $r_x = x - x_1$ et $r_y = y - y_1$

  Ainsi, $x' = x_0 + (r_x \cos\theta + r_y \sin\theta) / e$
  $y' = y_0 + (-r_x \sin\theta + r_y \cos\theta) / e$

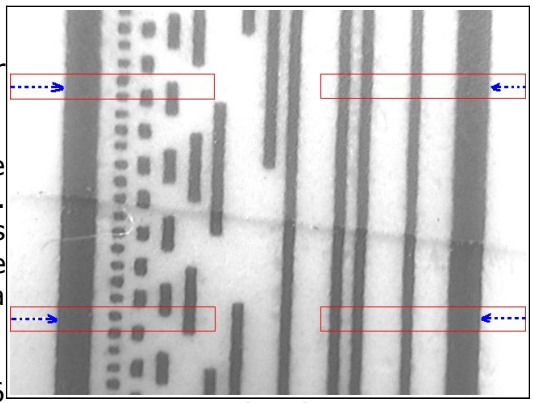
  On réitère 3 fois afin de converger vers une orientation et une échelle correcte de la grille.

- Finalement, on applique la procédure d'écrite dans §3.5.3 et §3.5.4 à l'ensemble des points de l'image.

IV) Lecture du code gray par la webcam

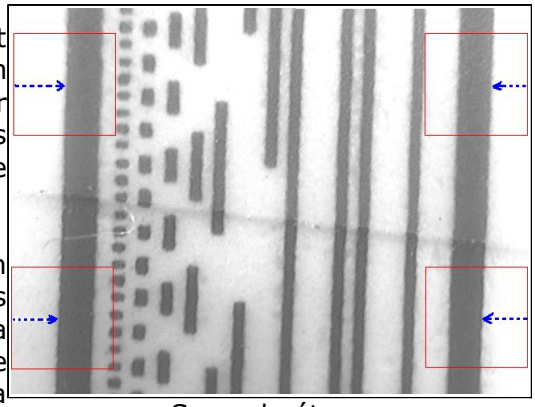
4.1) Détection des lignes guides

La bande imprimée comporte des bandes plus large de part et d'autre du code Gray. On les désignera par le terme de lignes guides.


Première étape

Toute la lecture du code gray repose sur la détection de ces lignes. L'algorithme de détection doit être robuste. Ainsi, nous utilisons une détection mettant en jeu des transformées de Fourrier rapide (FFT) permettant de réaliser un filtrage spatiale des bruits dû par exemple à la poussière, à la détérioration du code...

Dans une première étape, on définit 4 cadres de 256 par 32 pixels centrées respectivement sur la ligne 96 et 384, dont un côté est adjacent au bord de l'image.


Seconde étape

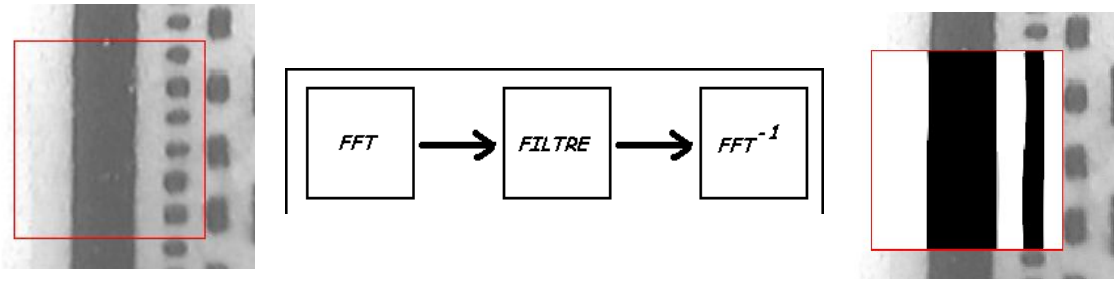
Le principe de détection des bords des lignes guides est de partir du bord de ligne jusqu'au moment où l'on observe un transition d'une couleur claire à une couleur sombre. Les trajectoires suivies sont repérées par les flèches bleues. Les pixels correspondant à cette transitions sont sur les bords extérieurs.

Lors d'un seconde étape, on réitère la méthode en considérant des cadres des 128 par 128 pixels respectivement centrés sur les pixels trouvés lors de la première étape. Cette seconde étape est nécessaire afin d'éviter de erreurs d'interprétations dû à la création d'artéfact due à la non périodicité de l'image lors du filtrage spatial.

- Transformés de Fourrier et Filtrage du bruit

Afin d'augmenter la qualité de la méthode présentée ci-dessous, nous réalisons un filtrage spatiale. Ainsi, la largeur des lignes guides sont choisies afin que leurs contours, une fois filtrées, ne sont trop déformés par leur environnement spatial.

Dans un premier temps chaque partie de l'image correspondant aux cadres rouges subit une FFT, on applique ensuite un filtrage dans l'espace des fréquences puis on applique une FFT inverse pour obtenir l'image filtrée.



Le filtre dans l'espace des fréquences à la forme suivante:

Soit r définit par $r = (f_x^2 + f\, f_y^2)^{1/2}$ avec $f_x = i_{max} - i$ et $f_y = j_{max} - j$

i_{max} et j_{max} correspondent aux demies-dimensions de l'image

i et j sont les coordonnées du pixel sur l'image de l'espace des fréquences

<u>Remarque</u>: Le pixel $i=i_{max}$ et $j = j_{max}$ correspond à l'amplitude de la valeur moyenne de l'image.

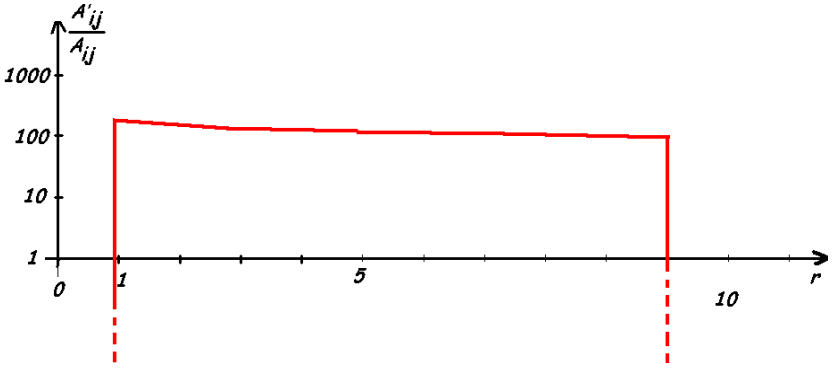
Finalement, ***f*** est un paramètre permet d'amplifier le filtrage suivant l'axe verticale par rapport rapport à l'axe horizontale, tenant ainsi compte de l'orientation principale des lignes guides.

On note A_{ij} l'amplitude (ie la valeur) associée au pixel (i,j) dans l'espace des fréquences avant la filtration et A'_{ij} après la filtation:

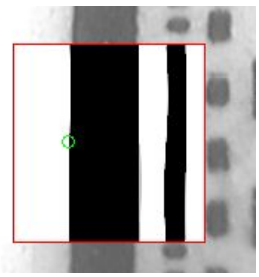
$$\begin{cases} \text{si } r < 1 \text{ ou si } r>e \quad A'_{ij} = 0 \\ \text{si } 0<r\leq e \quad A'_{ij} = G\left(\frac{e}{r}\right)^{\frac{1}{4}} A_{ij} \end{cases} \qquad \text{avec} \qquad \begin{array}{l} \boldsymbol{e} \text{ paramètre d'échelle} \\ \boldsymbol{G} \text{ le gain constant} \end{array}$$

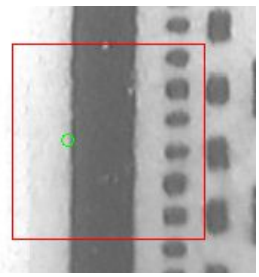
L'expérience montre que les valeurs f=30, e=10 et G=100 sont bien adaptées à notre problème.

Graphiquement ce filtre ce traduit par:

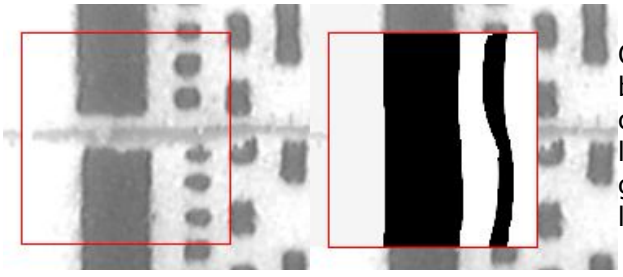


Ainsi, le filtre élimine la valeur moyenne de l'image, ce qui a pour effet d'éliminer le niveau de fond et de le ramener à zéro et les hautes fréquences correspondants aux bruits.



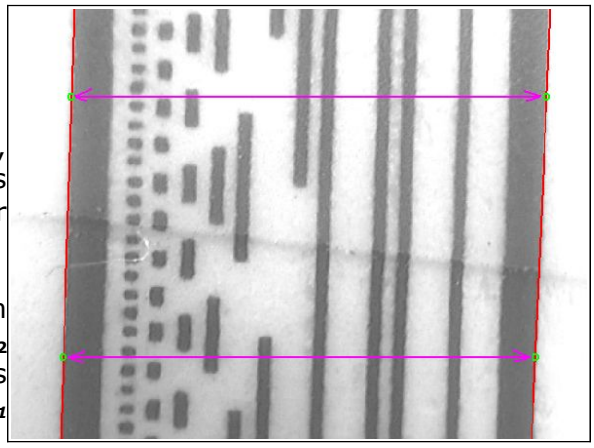


La méthode décrit au début du paragraphe est appliquée à l'image filtrée, permettant ainsi une détection faible et robuste du bord d'une ligne guide



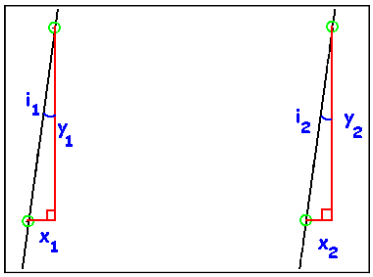
On peut illustrer la robustesse de la détection du bord de la ligne dans le cas une discontinuité de celle-ci. Les images à droite montrent une zone à la ligne guide est coupée. Pour autant, le filtre gomme celle-ci permettant de détecter la localisation du prolongement du bord de la ligne.

4.2) Inclinaison et échelle du code



En utilisant la méthode du paragraphe précédent, on connait la position des bords extérieurs des lignes guide sur la ligne 96 et 384 (cercles sur l'image de droite).

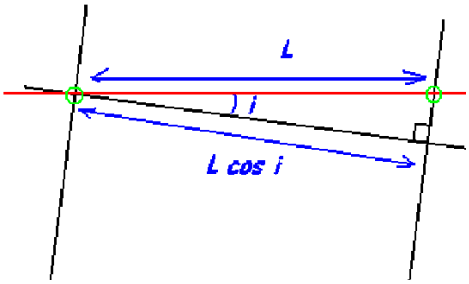
L'inclinaison ***i*** de la bande par rapport à la webcam est donnée par la moyenne des inclinaisons i_1 et i_2 des bords des lignes guides. Si la correction des distorsions optiques est correcte, les valeurs de i_1 et i_2 sont proches:



$\tan i_1 = x_1 / y_1$ $\tan i_1 = x_1 / y_1$

$i = (i_1 + i_2) / 2$

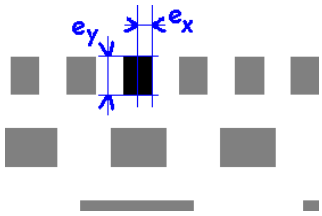
On connait par ailleurs la distance suivant l'axe horizontale des points des lignes guides. On note ***L*** cette distance.



La distance ***L'*** entre les bords extérieurs des lignes guides est donc donnée par :

$$L' = L \cos i$$

*La longueur **L'** est suivant l'axe perpendiculaire à celui du code gray*



On a $L' = \kappa\, e_y$ *e_y étant la larguer d'un bit de poids faible*

donc $e_y = \dfrac{L \cos i}{\kappa}$

κ est un coefficient de proportionnalité imposé par la géométrie de la bande lors de l'impression de la bande codée. Le fichier d'impression impose aussi le rapport R :

$R = e_y / e_x$ On obtient ainsi e_x par : $\boxed{e_x = \dfrac{L \cos i}{R\, \kappa}}$

Compte tenu du §2.5, $\kappa = \underset{\text{Nombre ligne de bit}}{n} + \underset{\text{Nombre ligne guide plus fraction d'élargissement de lignes guides}}{2 + D} + \underset{\text{Nombre d'interlignes}}{n+1}$

Ainsi, $\kappa = \kappa' + D$ avec $\kappa' = 2(n+2) - 1$

A noter que D se décompose de la façon suivante: $D = \underset{\text{Ligne guide de gauche}}{(T_g + E_g + O_G)} + \underset{\text{Ligne guide de droite}}{(T_g + E_g - O_D)}$

Le coefficient T_g provient du fait qu'une ligne guide est 3 fois plus large qu'une ligne de bit, ainsi: $T_g = 3 - 1$. E_g est l'élargissement lors de l'impression de la ligne guide en fraction de largeur d'une ligne de bit. O_G et O_D sont des offsets des lignes guides dus à l'impression.

4.3) Première estimation pour la lecture du code

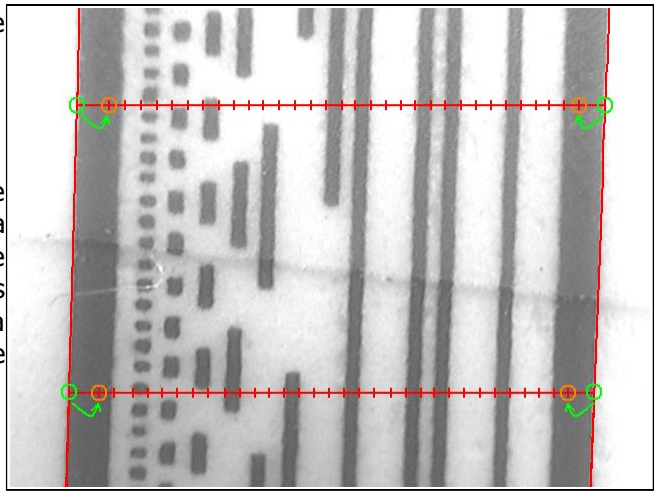
On reprend les notations introduites au paragraphe précédent.

Afin de localiser l'information qui nous intéresse, nous allons dans un premier temps localiser le ligne de bits et les espaces qui les séparent.

On raisonne de la même façon sur les deux lignes de lecture (ligne 96 et 384).

Sur ces lignes on considère l'élément de largeur des bits de données ***r*** vaut :

$$r = \frac{L}{\kappa}$$



- Dans un premier temps, on localise le bord intérieur de la ligne guide en considérant le pixel à une largeur de bit de donnée à l'intérieur de la ligne guide le décalage entre les points repérant le bord extérieur des lignes guide est repéré par les petites flèches sur la figure ci-contre. La valeur ***Δb*** de ce décalage est donnée par:

$\Delta b = (T_g + E_g + O_G)\, r$ pour la ligne guide gauche

$\Delta b = (T_g - E_g - O_D)\, r$ pour la ligne guide droite

Pour chaque ligne de lecture, on utilisera la largeur ***l*** suivante : $l = L\,\frac{\kappa'}{\kappa}$

Cette largeur correspond à la distance entre les points précédents situés sur les lignes de lecture et à l'intérieur des lignes guides.

Pour la suite, on note l'abscisse de ses points b_k.

Avec: k=0 pour la ligne guide de gauche et ligne de lecture du haut
k=4 pour la ligne guide de droite et ligne de lecture du haut
k=1 pour la ligne guide de gauche et ligne de lecture du bas
k=5 pour la ligne guide de droite et ligne de lecture du bas

- La seconde étape consiste à délimiter la localisation des lignes de bits et des bandes blanches le séparant. Dans un cadre idéal, représenté par le graduation des lignes de lectures de l'image ci-dessous, chaque zone commence et termine à un multiple entier de ***r.***

Les abscisses des pixels limites de ces zones sont donc:

$x_{n,0,k} = b_k + n\, r$ pour la limite gauche et $x_{n,1,k} = b_k + (n+1)r$ pour la limite droite

avec $0 \le n < \kappa'$

Toutefois, afin de rendre plus robuste la lecture, il faut tenir compte de l'incertitude sur la détermination des bords extérieurs des lignes guides et l'élargissement des lignes de bit lors de l'impression.

On note $\boldsymbol{E_b}$, l'élargissement lors de l'impression des lignes de bits en fraction de largeur de celle-ci *(Remarque: en principe $E_b = E_g$)*. On note $\boldsymbol{d_p}$ l'incertitude sur la position des lignes guides en faction de largeur d'une ligne de bit.

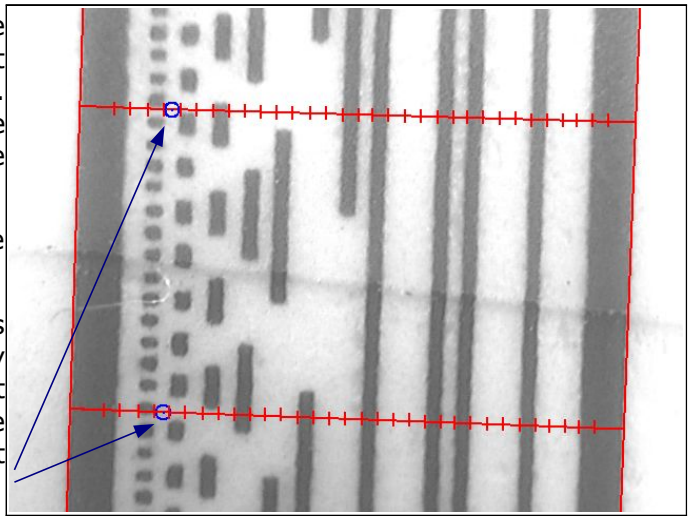
Ainsi, pour la limite gauche : $x_{n,0,k} = b_k + (\ n\ + j\, E_b + d_p\ )\, r$
pour la limite droite : $x_{n,1,k} = b_k + (\ n+1 - j\, E_b - d_p\ )\, r$

avec j = 0 si la zone ne correspond pas à une zone blanche *(i.e. sans impression)*
et j = 1 sinon.

4.4) Seconde estimation pour la lecture du code

On conserve les notations introduites au paragraphe précédent.

Dans le paragraphe précédent la tentative de repérage du code s'est faite strictement sur les lignes 96 et 384 de la webcam. Toutefois cette lecture est hasardeuse compte tenu de l'inclinaison du code sur le champ de l'image.



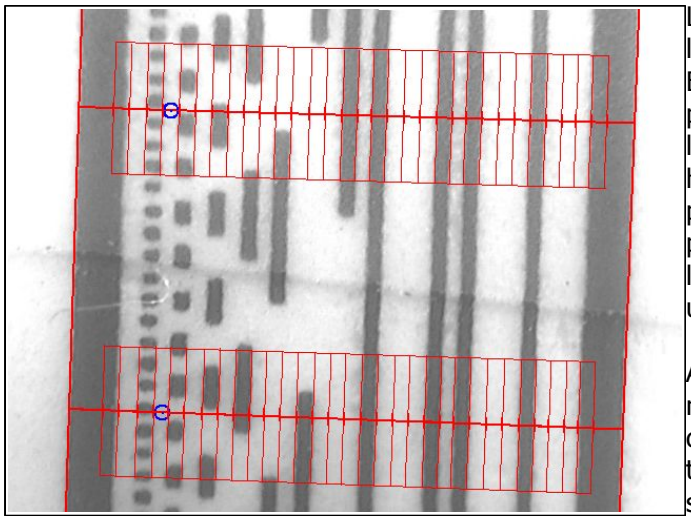
Les lignes de lecture sont donc inclinées de l'angle **i** mesuré sur les lignes guides.

Les points de pivotement des lignes guides sont choisis comme rattachés au code Gray et est localisé sur la bande blanche séparant les deux bits de poids faible (pour la suite on introduit n_e, $n_e = 2$). Les points sont repérés sur l'image ci-contre par les cercles.

La localisation des points de pivotement est dans la zone des bits de poids faibles. En effet, la méthode de localisation des bits de poids faibles est particulièrement robuste et imposera la valeur finale de la graduation lu par la webcam (voir §4.6).

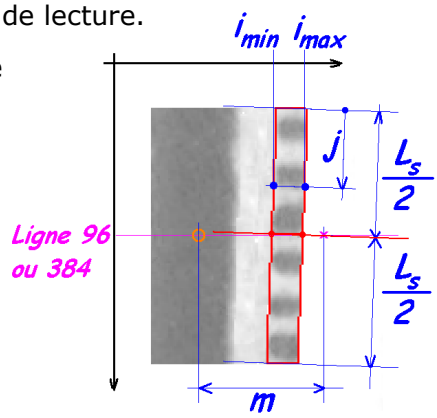
Ainsi, les points de pivotement conservent les ordonnées valant respectivement 96 et 384. La distance entre ces lignes de lecture effectives est toujours 288 pixels sur le champ de la webcam.

On définit à présent des zones de lecture du zone. On illustre ici le cas idéal (image ci-dessous), mais dans la pratique on considère les zones de lecture tenant compte des paramètres E_b *et* E_b introduit au paragraphe précédent.



L'ensemble des pixels de chaque zone de lecture sera utilisé par la lecture du code. En effet une simple détection du niveau des pixels situés exactement sur la ligne de lecture conduirait à une interprétation hasardeuse de la valeur du code. En particulier, la méthode serait particulièrement sensible au contraste entre les zones noires et blanches et conduirait à un code peu robuste.

Ainsi, de la façon que l'œil humain, nous nous servons de toute l'information disponible dans les zones de lectures pour trouver la valeur du niveau (bas ou haut) sur la ligne de lecture.



En utilisant les notations de la figure de droite, les pixels d'une zone de lecture sont tels que:

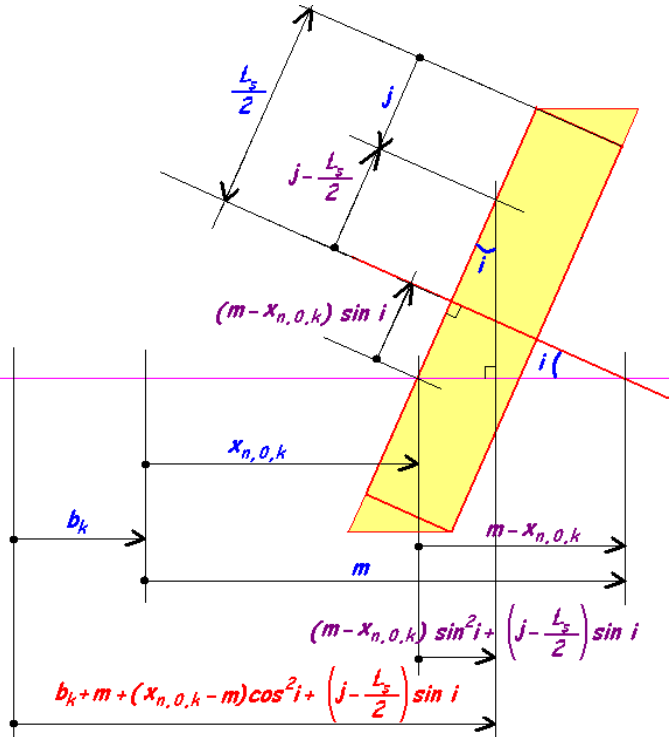
$$i_{min} = b_k + m + (x_{n,0,k} - m)\cos^2 i + (j - \frac{L_s}{2})\sin i$$

$$i_{max} = b_k + m + (x_{n,1,k} - m)\cos^2 i + (j - \frac{L_s}{2})\sin i$$

avec k=0 pour la ligne de lecture du haut
k=1 pour la ligne de lecture du bas

et $m = (n_e+1.5)\frac{L}{\kappa}$ On rappelle que $n_e=2$

Le figure ci-contre présente la géométrie d'une zone de lecture et les différentes cotations permettant d'obtenir i_{min} et i_{max}.



Les niveaux des pixels d'une zone de lecture sont stockés dans un tableau aussi bien pour les lignes guides, les lignes de bits, et les inter-lignes blanches.

4.5) Bits de poids faible

On conserve les notations introduites au paragraphe précédent.

Les bits de poids faibles que l'on va considérer dans cette partie sont tels que la zone de lecteur est suffisamment longue pour voir au moins une zone noire et un zone blanche entièrement. La sélection des bits considérée ici dépendant donc de l'échelle spatiale $\mathbf{e_x}$ que l'on notera pour la suite $\boldsymbol{e}$.

Par ailleurs, L_S est choisi comme étant une puissance entière de 2. Ainsi, on note $\boldsymbol{p_s}$ la valeur :

$$p_s = \frac{\ln L_s}{\ln 2}$$ avec ln la fonction logarithme népérien.

Ainsi, la somme des longueurs d'une zone noire et d'une zone blanche d'un bit de poids n doit vérifier:

$$(2^n + 2^n)\, e < L_s$$

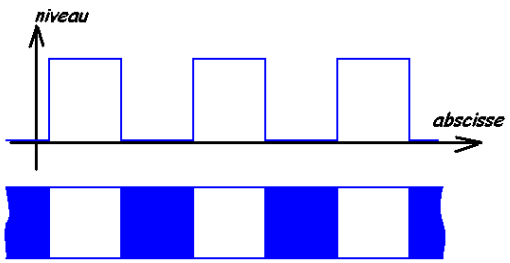
d'où $$n \ln(2) + \ln(2e) < p_s \ln(2)$$

Ainsi $$n < p_s - \frac{\ln(2e)}{\ln(2)}$$

Le nombre de bits considéré n_{max} est donc $n_{max} = \boldsymbol{E}\left(p_s - \frac{\ln(2e)}{\ln(2)}\right)$, $\boldsymbol{E}$ étant la fonction partie entière

- Une ligne de bit est une succession de rectangles noires et de rectangles blancs de même largeur

Les longueurs des intervalles sont connues et ne dépendent que de la valeur de e. On peut donc considérer le modèle idéal et le comparer à l'observation sur l'ensemble de la zone de lecture d'un bit.



Le principe est de tester le modèle en le décalant pixel par pixel par rapport à l'observation (voir figure ci-après).