Suppression de distorsion projective par transformation directe linéaire

Victor Devilliers
Master Mathématiques et Applications
Université de Lorraine

2021

Table des matières

I. Introduction **3**

II. Contexte **4**

III. Espace projectifs **6**
1. Coordonnées homogènes et objets usuels 6
2. Transformations usuelles 9
3. Homographies 11
4. Degrés de liberté d'objets 13
5. Degrés de liberté des transformations 14

IV. Cadre de la modélisation **15**

V. Transformation directe linéaire ou DLT **17**
1. Homographie à partir de correspondances entre points 18
2. Normalisation 23
3. Détermination de la matrice de caméra 26

VI. Image de la conique absolue et paramètres internes de caméra **28**
1. Contraintes sur ω 28
2. Extraction des paramètres internes 30
3. Paramètres internes de caméra et homographie 31

VII. Annexe **32**
1. SVD et Factorisation de Cholesky 32
2. Matlab 33

VIII. Bibliographie **35**

I. Introduction

La vision par ordinateur est apparue dans les années 60. Le but initial était d'arriver à imiter les systèmes de vision humaine de sorte qu'on puisse "interroger" directement un ordinateur afin d'automatiser le processsus d'analyse d'image.

Les avancées technologiques des soixante dernières années ont permis un développement continu de riches modèles et théories mathématiques qui engendrent de nombreuses applications : en physique, neurobiologie, traitement de signal, médecine, véhicules autonomes, armement...

Autant de domaines où la vision par ordinateur a su s'imposer en seulement un demi-siècle. Plus récemment, la disponibilité de masses de données conséquentes a entraîné l'essor du deep learning et de la vision par ordinateur à travers, notamment, la reconnaissance faciale.

Nous verrons comment de nombreuses branches des mathématiques s'assemblent pour former de redoutables outils de traitement d'image, et comment des théories poussées permettrent d'extraire des informations importantes de détails a priori anodins.

FIGURE 1 – Deux cartes de même taille dans la chambre d'Ames, illusion d'optique célèbre

II. Contexte

Les pixels d'une image sont identifiées par leurs coordoonées en abscisse et en ordonnée, ainsi que par une valeur d'intensité, définie par la couleur.
Les images sont enregistrées dans matlab grâce à la commande *imread()*.
Dans un premier temps, les données d'une image sont au format RGB : chaque point de l'image, est associé à 3 valeurs, chacune correspondant à l'intensité de rouge, vert ou bleu respectivement.

La commande *rgb2gray()* modifie l'intensité de la manière suivante :
on exprime l'intensité I en *grayscale* comme somme pondérée des valeurs RGB
$I = 0.2989R + 0.587G + 0.114B$
Par la suite nous utiliserons des images en *grayscale* ; les données de l'image sont stockées dans une matrice $m \times n$ dont les éléments indiquent une valeur qui correspond à l'intensité.
Suivant le type de donnée, l'intensité est à valeurs dans des intervalles différents. Lorsqu'il s'agit d'un tableau d'éléments de type *single* ou *double*(single/double precision respectivement), l'intensité est à valeurs dans $[0, 1]$; les éléments de type single offrent une moins grande précision que les double, qui sont à valeurs dans (environ) :
$[-1.8e + 308, -2.2e - 308] \cup [2.2e - 308, 1.8e + 308]$
Par défaut, matlab considère les données comme des type double. Les autres types comme int8 sont encore plus précis et dans ce cas l'intensité est à valeurs dans un intervalle bien plus grand, par exemple $[0, 255]$ pour int8.

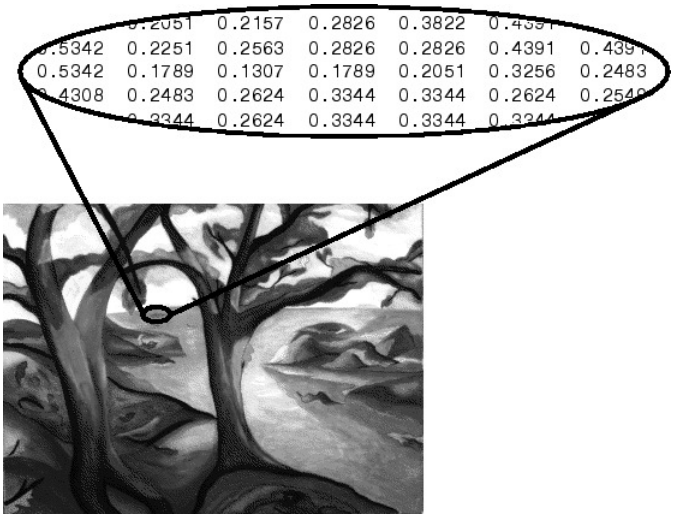


FIGURE 2 – Position des pixels et intensité associée en grayscale

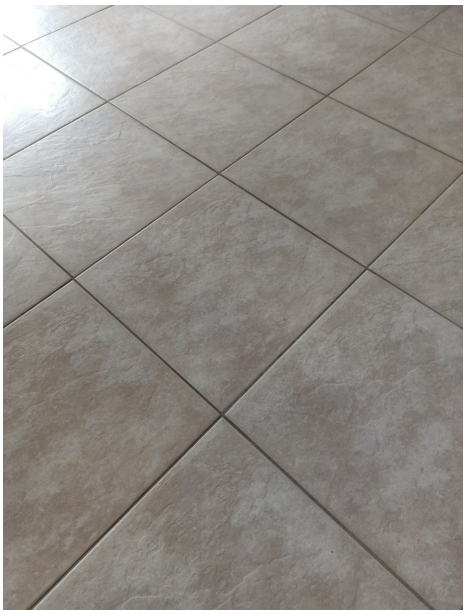


Figure 3 – Exemple de distortion

Sur cette image, l'orthogonalité des lignes n'est pas respectée, et la distortion fausse les proportions : les carreaux ne sont pas carrés.
On cherche à rectifier à la fois les proportions et les angles, en ne disposant que de cette image et en connaissant certaines propriétés "réelles" de ses éléments, dans cet exemple les dimensions du carré voulu et l'orthogonalité des lignes.

Nous verrons dans un premier temps quel cadre mathématique peut répondre à cette problématique et quels outils en tirer avant d'en déduire des applications directes via Matlab.

Les *toolbox* présentes dans Matlab ne seront pas utilisées ; notamment "Computer Vision" et "Image Processing".

Une fois les transformations d'image déterminées, la fonction *griddata* permettra, par interpolation, de créer une nouvelle image à partir de l'image de départ et de l'homographie.
Son fonctionnement est détaillé en annexe.

III. Espace projectifs

1. Coordonnées homogènes et objets usuels

Les points d'un espace euclidien de dimension 2 sont représentés par une paire de coordonnées $\begin{pmatrix} x \\ y \end{pmatrix}$, auxquelles on ajoute une coordonnée d'échelle : $\begin{pmatrix} x \\ y \\ s \end{pmatrix}$

On notera $\mathbb{P}^2$ le plan usuel augmenté d'une coordonnée d'échelle, aussi appelée paramètre projectif.

Le point (x, y) est représenté par $\begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$ et on passe des coordonnées du plan projectif aux coordonnées classiques en divisant par le facteur d'échelle lorsqu'il est non nul :

$$\begin{pmatrix} x \\ y \\ s \end{pmatrix} \longrightarrow \begin{pmatrix} \frac{x}{s} \\ \frac{y}{s} \end{pmatrix}$$

Définition : Pour tout $\lambda \in \mathbb{Z}^*$, $x, y \in \mathbb{P}^2$ on note $x \sim y \iff x = \lambda y$.

Proposition : La relation définie ci-dessus est une relation d'équivalence sur $\mathbb{P}^2$.

<u>Preuve</u> :
(i) Réflexivité : $x = 1.x$
(ii) Symétrie : $x = \lambda y \implies y = \frac{1}{\lambda} x$ donc $x \sim y \implies y \sim x$
(iii) Transivité : si $x \sim y$ et $y \sim z$ alors $y = \lambda x \implies \lambda x \sim z \implies x \sim z$

Par la suite, un point en coordonnées homogènes sera considéré comme le représentant de la classe d'équivalence associée.

Lorsque la 3ème coordonnée est nulle, il n'y a pas de correspondance entre le point représenté en coordonnées homogènes et un point du plan ; la division par le facteur d'échelle n'étant alors pas définie.
Ces points sont appelés "points à l'infini" : $\mathrm{x}_\infty = \{(x, y, 0) | x, y \in \mathbb{R}^2\}$

L'ensemble des classes d'équivalence des vecteurs de $\mathbb{R}^3 \backslash (0, 0, 0)$ forme l'espace projectif $\mathbb{P}^2$.

Une équation de droite d dans $\mathbb{P}^2$ est donnée par :
$ax + by + cw = 0$
où $d = (a, b, c)^T$ et les points appartenant à d vérifient $\mathrm{x} = (x, y, w)^T$.
Autrement dit :
$\langle d, \mathrm{x} \rangle = ax + by + cw = 0$, qu'on peut écrire matriciellement :
$d^T \mathrm{x} = 0$ ie $\mathrm{x} \in d$ ou encore $\mathrm{x}^T d = 0$: d passe par x

Dans le plan projectif, deux droites s'intersectent en un point donné par $d_1 \wedge d_2$ et deux points distincts appartiennent à une unique droite qui vérifie $x \wedge x' = d$

Le birapport :
Le birapport ou *cross ratio* va jouer un rôle important dans les transformations projectives, il est défini de la manière suivante :
Sur la droite projective $\mathbb{P}^1$, les points sont représentés en coordonées homogènes par $p = (p_1, p_2)^T$, d'échelle (ou paramètre projectif) $\frac{p_1}{p_2}$

Pour deux paires de points p_a, p_b et p_c, p_d, de paramètres projectifs respectifs a, b, c, d le *cross ratio* s'écrit :

$$\mathrm{C}_{\{p_a,p_b,p_c,p_d\}} = \frac{a-c}{a-d} : \frac{b-c}{b-d} = \frac{a-c.b-d}{b-c.a-d}$$

Coniques dans $\mathbb{P}^2$:

Une conique est donnée par une équation du second degré dans le plan. En géométrie euclidienne on distingue, suivant l'angle formé par le plan et les génératrices du cône, trois types de conique : l'ellipse, la parabole et l'hyperbole. Par la suite, on ne fera pas de distinctions entre les types de coniques : dans le plan projectif toutes les coniques non-dégénérées sont projectivement équivalentes.

En coordonnées usuelles, l'équation d'une conique est un polynôme de degré 2 :
$ax^2 + bxy + cy^2 + dx + ey + f = 0$
En passant aux coordonnées homogènes : $x \longrightarrow \frac{x_1}{x_3}, y \longrightarrow \frac{x_2}{x_3}$, on obtient :

$a_1^x + bx_1x_2 + cx_2^2 + dx_1x_3 + ex_2x_3 + fx_3^2 = 0$

Soit $\mathrm{x}^T C \mathrm{x} = 0$

avec $C = \begin{pmatrix} a & \frac{b}{2} & \frac{d}{2} \\ \frac{b}{2} & c & \frac{e}{2} \\ \frac{d}{2} & \frac{e}{2} & f \end{pmatrix}$ matrice (symétrique) des coefficients

De la même manière que pour $\mathbb{P}^2$, l'espace euclidien $\mathbb{R}^n$ peut être ramené à l'espace projectif $\mathbb{P}^n$ en représentant les points comme des vecteurs homogènes.

x $\in \mathbb{P}^3$ s'écrit x $= (x_1, x_2, x_3, x_4)^T$ où x_4 est le paramètre projectif.
L'équation d'un plan est : $\pi_1 X + \pi_2 Y + \pi_3 Z + \pi_4 = 0$

En coordonnées usuelles :
$X \longrightarrow \frac{x_1}{x_4}, Y \longrightarrow \frac{x_2}{x_4}, Z \longrightarrow \frac{x_3}{x_4}$, soit :
$\pi_1 x_1 + \pi_2 x_2 + \pi_3 x_3 + \pi_4 = 0$
Autrement dit, $\pi^T X = 0$ lorsque $X \in \pi$.

Le plan à l'infini π_∞ permet d'identifier des propriétés affines comme le parallélisme et a les propriétés suivantes :
- deux plans sont parallèles si et seulement si ils s'intersectent en une droite appartenant à π_∞
- une droite est parallèle à une autre droite ou un plan si et seulement si leur point d'intersection est sur π_∞
Ainsi, dans $\mathbb{P}^3$, tous les plans s'intersectent en une droite et deux plans parallèles s'intersectent en une droite appartenant au plan à l'infini.

Les points à l'infini dans $\mathbb{P}^2$ forment une droite qu'on appelle droite à l'infini, habituellement notée l_∞, tandis que dans $\mathbb{P}^3$ ils forment π_∞.
En choisissant une droite du plan projectif comme l_∞, il est possible de définir le parallélisme de deux droites.
La géométrie affine est alors un cas particulier de géométrie projective où l'on a fixé l_∞.
En géométrie affine on peut arbitrairement déformer le plan de sorte qu'aucune distinction n'est faite entre un cercle et une ellipse, de la même manière, en coordonnées homogènes (X, Y, Z, S), toutes les sphères intersectent le plan à l'infini en une courbe d'équation :

$X^2 + Y^2 + Z^2 = 0; S = 0$
C'est l'équation d'une conique appartenant au plan à l'infini, composée uniquement de points complexes. On l'appelle la conique absolue, notée Ω_∞.

De la même manière qu'on a défini la géométrie affine en 2D comme un cas particulier de géométrie projective en fixant l_∞, on définit la géométrie euclidienne en 3D comme cas particulier de géométrie projective en fixant π_∞ et Ω_∞.

La conique absolue et son image, notée ω, permettront d'obtenir des informations supplémentaires à partir d'éléments "réels" d'une image : orthogonalité naturelle (par exemple sur des fenêtres), points de fuite, lignes de fuite..

2. Transformations usuelles

Le groupe formé par les transformations projectives est le groupe projectif linéaire $PL(n)$: c'est le groupe quotient de GL(n) par la relation d'équivalence $M \sim N \iff M = \lambda N$.

On peut faire agir PL(3) sur $\mathbb{P}^2$; PL(3) contient le groupe affine : il s'agit des transformations projectives ayant [0,0,1] pour dernière colonne.
Il contient aussi le groupe Euclidien, lui même inclus dans le groupe affine : celui-ci correspond aux matrices ayant comme bloc supérieur gauche une matrice orthogonale 2×2.

Chacun de ces types de transformation possède des invariants différents qui détermineront l'outil à choisir suivant le type de rectification recherchée.

Isométries :
Sous forme matricielle :
Dans toute la suite $t = (t_x, t_y)^T$ est une translation du plan.

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} \epsilon cos\theta & -sin\theta & t_x \\ \epsilon sin\theta & cos\theta & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

où $\epsilon = \pm 1$
Si $\epsilon = 1$, l'isométrie est la composition d'une translation et d'une rotation et préserve l'orientation. Lorsque $\epsilon = -1$, l'orientation est inversée.
La matrice par bloc dans $\mathbb{P}^2$ est donnée par :

$$x' = H_e x = \begin{bmatrix} R & t \\ (0,0) & 1 \end{bmatrix} x$$

avec R matrice de rotation 2×2, t une translation des deux premières coordonnées. Lorsqu'il s'agit d'une rotation resp. translation pure, $t = 0$ resp. $R = 0$.

Les isométries conservent les distances entre deux points, l'aire ou encore l'angle entre deux droites (donc le parallélisme).

Similitudes :

Il s'agit d'une isométrie composée avec une homothétie, la matrice correspondante est :

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} \rho cos\theta & -\rho sin\theta & t_x \\ \rho sin\theta & \rho cos\theta & t_y \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

$$x' = H_s x = \begin{bmatrix} sR & t \\ (0,0) & 1 \end{bmatrix} x$$

Les similitudes conservent les formes, les angles (et à nouveau le parallélisme) mais pas la distance entre deux points : seul le ratio de deux longueurs est conservé (on multiplie et divise par le facteur associé à l'homotétie).

Transformations affines :

Elles s'obtiennent par transformation non-singulière suivie d'une translation. La matrice correspondante est :

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

soit :

$$x' = H_a x = \begin{bmatrix} A & t \\ (0,0) & 1 \end{bmatrix} x$$

La décomposition de A par SVD donne :

$$A = UDV^T = (UV^T)(VDV^T) = R(\theta)R(-\phi)DR(\phi)$$

car U et V sont orthogonales, où $R(\theta), R(\phi)$ sont des rotations et $D = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}$

Il s'agit donc de plusieurs rotations, et d'une multiplication par deux scalaires différents selon la coordonnée. La différence principale avec une similitude est donc la présence de la multiplication par λ_1, λ_2, là où l'homotétie multiplie les deux premières coordonnées par le même scalaire.

Invariants : d'après la décomposition qui précède, les aires ne sont pas modifiées car $detA = \lambda_1\lambda_2$ donc A modifie toutes les aires uniformément.
Le parallélisme est à nouveau préservé : comme deux droites parallèles s'intersectent en un point à l'infini et que la 3ème ligne de la matrice d'une transformation affine préserve le paramètre projectif, ce point à l'infini reste à l'infini et l'intersection des droites n'a pas lieu "dans" le plan.
Ces deux propriétés font que les ratios de longueurs entre droites parallèles sont eux aussi préservés.

Les transformations affines généralisent les similitudes au sens où cette fois les angles ne sont pas préservés et donc les formes ne sont pas préservées.
Notons que ces trois premiers types de transformation ont en commun $[0, 0, 1]$ comme 3ème ligne, elles n'agissent donc pas sur l'échelle (paramètre projectif).

3. Homographies

:

Transformation non-singulière de coordonnées homogènes : c'est la généralisation d'une transformation affine qui n'agit que sur les coordonnées non-homogènes.

$$\begin{pmatrix} x'_1 \\ x'_2 \\ x'_3 \end{pmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

Bloc :
$x' = H_p x = \begin{bmatrix} A & t \\ (v_1, v_2) & \lambda \end{bmatrix} x$

Impossible de distinguer les homographies qui préservent l'orientation.

Lorsque $\lambda \neq 0$ toute homographie H admet la décomposition suivante :
$H = H_S H_A H_P = \begin{bmatrix} sR & t \\ (0,0) & 1 \end{bmatrix} \begin{bmatrix} K & 0 \\ (0,0) & 1 \end{bmatrix} \begin{bmatrix} I & 0 \\ (v_1, v_2) & \lambda \end{bmatrix}$
avec K une matrice triangulaire supérieure de déterminant 1, H_S une similitude, H_A une transformation affine et H_p une transformation projective : chacune d'elle a ses propres invariants, ce qui rend possible de rectifier une image étape par étape.

Les ratios de longueurs ne sont pas invariants par homographie mais le bi-rapport défini précédemment l'est, de plus, les homographies préservent les lignes.
Ainsi, les angles, les aires et les longueurs ne sont pas préservés, ce qui ouvre la voie à de nouveaux types de correction d'image.

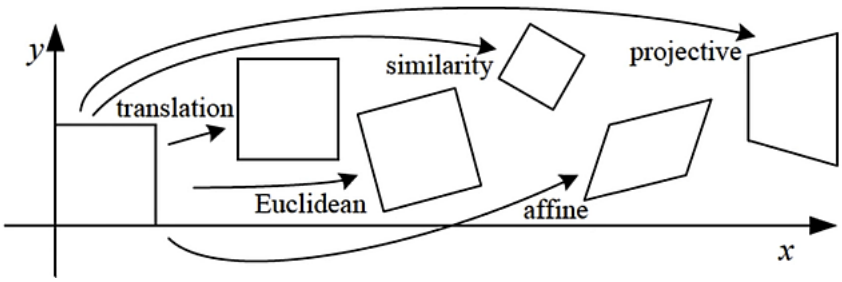


FIGURE 4 – Image d'un carré par type de transformation

Propriétés des homographies :

Une transformation projective : $\mathbb{P}^n \longrightarrow \mathbb{P}^n$ multiplie un point par une matrice non singulière ; ainsi les points à l'infini ne sont **pas** préservés par les transformations projectives :
Les transformations classiques vues précédemment ont toutes $[0, 0, 1]$ comme 3ème ligne, elles envoient la 3ème coordoonnée sur elle-même et préservent donc les points à l'infini.
Tandis que la 3ème ligne d'une homographie est $(h_1, h_2, h_3) \neq (0, 0, 1)$;
soit x_∞ un point à l'infini, alors :
$Hx_\infty = H \begin{pmatrix} x \\ y \\ 0 \end{pmatrix} = \begin{pmatrix} x' \\ y' \\ h_7x + h_8y \end{pmatrix}$ donc si $h_7x + h_8y \neq 0$, on peut donc repasser en coordonnées usuelles, de sorte que Hx_∞ correspond à $\begin{pmatrix} \frac{x'}{h_7x+h_8y} \\ \frac{x'}{h_7x+h_8y} \end{pmatrix}$.

Définition : Une homographie (ou transformation projective) est une application inversible $h : \mathbb{P}^2 \longrightarrow \mathbb{P}^2$ telle que trois points x_1, x_2, x_3 sont sur la même droite si et seulement si $h(x_1), h(x_2), h(x_3)$ le sont aussi.

Théorème : $H : \mathbb{P}^2 \longrightarrow \mathbb{P}^2$ est une homographie si et seulement si il existe une matrice non-singulière H telle que pour tout représentant x d'une classe de points de $\mathbb{P}^2$, h vérifie $h(x) = H(x)$.

De plus, on peut multiplier H par un scalaire non nul sans changer la transformation projective : on dit que H est une matrice homogène. C'est dû au fait que, de la même manière que pour la représentation homogène d'un point, seul le ratio des éléments de la matrice importe donc la multiplication par un scalaire non nul n'a aucun effet sur le ratio.

Le birapport de 4 points caractérise leur configuration projective.
Nous verrons plus loin que la connaissance de 4 points et de leurs images permet à elle seule de déterminer l'homographie associée.

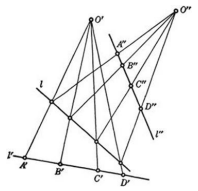


FIGURE 5 – Invariance du birapport par transformation projective

4. Degrés de liberté d'objets

:

Un point est défini grâce à deux coordonnées ce qui correspond à deux degrés de liberté, tandis qu'une droite (en coordonnées homogènes) dépend de deux paramètres : deux ratios indépendants ; en coordonnées usuelles les deux paramètres d'une droite sont, par exemple, l'ordonnée à l'origine et le coefficient directeur de la droite.

On a déterminé précédemment la matrice d'une conique :

$C = \begin{pmatrix} a & \frac{b}{2} & \frac{d}{2} \\ \frac{b}{2} & c & \frac{e}{2} \\ \frac{d}{2} & \frac{e}{2} & f \end{pmatrix}$

Il y a 6 coefficients différents ; choisissons indifféremment l'un d'entre eux.

On peut alors obtenir 5 ratios indépendants les uns des autres : par exemple $\frac{a}{b}, \frac{a}{c}, \frac{a}{d}, \frac{a}{e}, \frac{a}{f}$ en choisissant a ; le ratio $\frac{a}{a}$ n'apporte aucune information supplémentaire.

On a donc 5 degrés de liberté pour une conique ; on aurait aussi pu dire qu'il y a six coefficients à choisir moins un coefficient d'échelle.

En reprenant l'équation d'une conique pour un point (x_i, y_i) quelconque, on obtient :

$a_i^{x2} + bx_iy_i + cy_i^2 + dx_i + ey_i + f = 0$

ie $(x_i^2, x_iy_i, y_i^2, x_i, y_i, 1)c = 0$ pour $c = (a, b, c, d, e, f)^T$

Supposant qu'on connaisse les coordonnées de 5 points distincts, on peut rassembler ces équations dans un même système :

$$\begin{bmatrix} x_1^2 & x_1y_1 & y_1^2 & x_1 & y_1 & 1 \\ x_2^2 & x_2y_2 & y_2^2 & x_2 & y_2 & 1 \\ x_3^2 & x_3y_3 & y_3^2 & x_3 & y_3 & 1 \\ x_3^2 & x_3y_3 & y_3^2 & x_3 & y_3 & 1 \\ x_3^2 & x_3y_3 & y_3^2 & x_3 & y_3 & 1 \end{bmatrix} c = 0$$

On obtient ainsi une matrice 5×6 et la solution de ce système s'obtient en déterminant son noyau.

Ce raisonnement, à savoir la construction d'une matrice de contraintes à partir de données connues, et sa résolution grâce au noyau (en particulier par SVD) sera omniprésent par la suite.

L'exemple de la conique est principalement illustratif puisqu'il ne s'agit pas ici d'un système linéaire.

5. Degrés de liberté des transformations

Dans le cas d'une isométrie, si elle préserve l'orientation (cas $\epsilon = 1$), il y a un degré de liberté pour la rotation (qui correspond au choix de l'angle θ) et deux degrés de liberté pour la translation, 1 pour la translation en x, un pour la translation en y.
Deux points suffisent donc à déterminer une transformation euclidienne du plan.

Une similitude diffère d'une transformation euclidienne seulement à multiplication par un scalaire (de la rotation) près ; soit 3+1 degrés de liberté en comptant le choix du scalaire, à nouveau, 2 points suffisent.

On a vu précédemment $\begin{bmatrix} A & t \\ (0,0) & 1 \end{bmatrix} x$ soit 6 éléments à choisir, la dernière ligne étant déjà fixée.
Des correspondances entre 3 points (d'une image à l'autre) suffisent donc.
Nous verrons par la suite qu'une homographie a 8 degrés de liberté et que 4 correspondances entre points suffisent : on en déduira un premier algorithme.

	Degrés de liberté	**Nombre d'images de points nécessaires**
Transformation euclidienne	3	2
Similitude	**4**	2
Transformation Affine	6	3
Homographie	8	4

De manière plus générale, H peut se décomposer en $H = MN$ avec M une similitude ,la partie métrique de la transformation , et N la partie non-métrique de l'homographie.

N s'écrit à son tour $N = H_a H_p$ avec H_a une transformation affine et H_p une transformation projective.

IV. Cadre de la modélisation

On utilise le modèle ci-dessous, dit de caméra à sténopé :

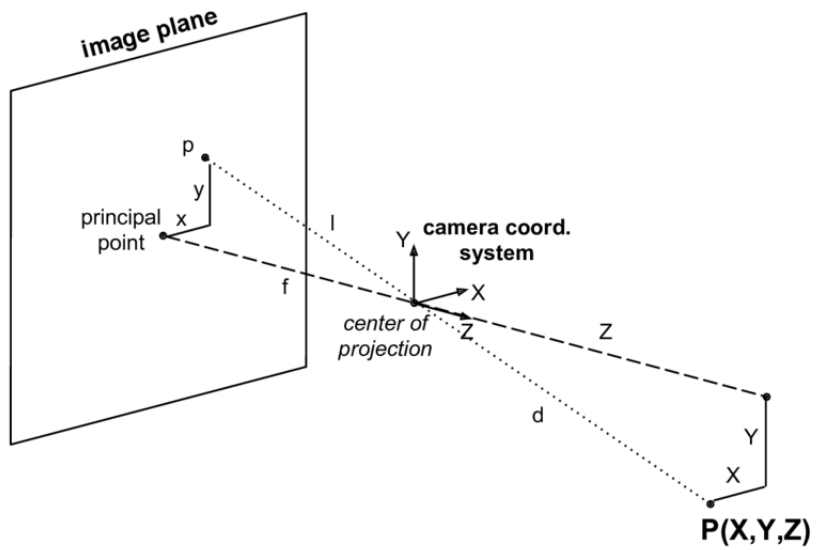


FIGURE 6 – Modèle de caméra à sténopé

On le représente par une matrice de projection P 3×4 tel que :
$PX =$ x pour X un point du monde réel et x un point de l'image.

P se décompose en $P = K(R|t)$
avec K la matrice des paramètres internes de la caméra et R,t une rotation et une translation respectivement, qui donnent l'orientation et la position de la caméra par rapport aux systèmes de coordonnées du monde réel.

$$K = \begin{bmatrix} f & s & u_0 \\ 0 & rf & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

Où f correspond à la distance focale, s l'angle entre les capteurs horizontaux et verticaux de la caméra (le *skew*), en général considéré comme nul , (u_0, v_0) les coordonnées du centre optique et r le format d'image (ou plutôt *aspect ratio*) c'est le rapport largeur :hauteur de l'image.
Le **point principal** correspond à l'intersection de l'axe optique et du plan de l'image.

Lorsque qu'une caméra n'est pas placée sur le plan à l'infini, l'image du plan par la caméra apparaît sur l'image car chaque point de l'image correspond à un rayon de l'espace (provenant de la caméra) qui intersecte le plan à l'infini en un unique point.
Ainsi, la connaissance du plan à l'infini ne donne pas d'information supplémentaire sur l'image.
Ce n'est pas le cas de la conique absolue : c'est une conique appartenant au plan à l'infini, donc sa projection est aussi une conique sur le plan de l'image : ω.
Lorsqu'on connaît la position de ω sur une image, on dit que la caméra est "calibrée".

Par définition, l'image de la projection centrale d'un plan est liée au plan "original" par une transformation projective, il s'en suit que l'image correspond à une distortion projective de l'original.

FIGURE 7 – Autre exemple de distortion

Il existe donc une homographie qui corrige cette distortion en transformant l'image de départ en une image aux propriétés voulues.
Comment déterminer cette homographie ?
Bien qu'on connaisse la "composition" générale d'une homographie on est très loin de deviner ses coefficients, d'autant plus que, des propriétés comme l'horizontalité seront faussées à la moindre variation de coefficients (tant qu'il ne s'agit pas du coefficient d'échelle).
Nous allons montrer qu'il est possible de déterminer directement l'homographie, sans connaître sa décomposition.

V. Transformation directe linéaire ou DLT

L'homographie qui envoie un point appartenant au plan de l'image sur un point du plan du monde réel est une transformation projective en 2D ; elle est représentée par une matrice homogène H $\in \mathbb{M}_{3,3}(\mathbb{R})$ non-singulière vérifiant :

$\mathrm{x}' = H\mathrm{x}$

Supposons dans un premier temps qu'on connaisse explicitement la position, sur un plan du monde réel, d'un certain nombres de points de l'image qu'on cherche à rectifier.

$\mathrm{x}' = H\mathrm{x}$ s'écrit explicitement :
avec x, x' des points du plan projectif représentés en coordonnées homogènes, c'est à dire :

$$\begin{pmatrix} x'_1 \\ x'_2 \\ x'_3 \end{pmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

H étant une matrice homogène : la multiplication par un scalaire non nul n'altère pas la transformation projective, seuls comptent les ratios des éléments de la matrice.

Seuls les ratios des éléments comptent : on en fixe un au hasard et on obtient 8 ratios indépendants en le divisant tour à tour par chacun des autres.

Il y a donc 8 degrés de liberté pour une homographie, le dernier élément restant correspond à un facteur d'échelle qui n'altère pas H puisqu'elle est homogène.

1. Homographie à partir de correspondances entre points

Connaissant les coordonnées d'un nombre suffisants de points dans l'image de départ, et celles des points recherchés dans l'image d'arrivée, il est possible de déterminer l'homographie correspondante H à partir de l'algorithme de DLT.

Commençons par déterminer le nombre de points nécessaires ; d'après ce qui précède on a 8 degrés de liberté pour l'homographie recherchée.

Par la suite, notons :

$\mathrm{x_i} = \begin{pmatrix} x_i \\ y_i \\ w_i \end{pmatrix}$ un point en coordonnées homogènes de l'image de départ

$\mathrm{x_i}' = \begin{pmatrix} x_i' \\ y_i' \\ w_i' \end{pmatrix}$ un point en coordonnées homogènes de l'image d'arrivée

Pour chaque correspondance $\mathrm{x_i} \longleftrightarrow \mathrm{x_i}'$:
on a deux degrés de liberté : un pour chacune des deux premières coordonnées de $\mathrm{x_i}'$, la troisième correspondant au facteur d'échelle :
H doit envoyer $\mathrm{x_i}$ sur $\mathrm{x_i}'$, et à nouveau H est homogène.

Chaque correspondance entre deux points compte pour 2 degrés de liberté, il suffit donc de 4 correspondances pour atteindre les 8 degrés de liberté de l'homographie.

Revenons à la définition de l'homographie ; on cherche à résoudre, pour chaque correspondance :
$\mathrm{x_i}' = H\mathrm{x_i}$ d'inconnue H.

Or H est homogène, cela revient alors à résoudre :

$\mathrm{x_i}' = \lambda H\mathrm{x_i}$
avec $\lambda \in \mathbb{R}$ un scalaire

Ou encore :
$\mathrm{x_i}' \wedge H\mathrm{x_i} = 0$
Puisque le produit vectoriel de deux vecteurs est nul $\iff$ ils sont colinéaires, et $H\mathrm{x_i}$ est bien un vecteur à 3 lignes.

On cherche H sous la forme $\mathrm{H} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$

En notant $h^{(j)T}$ la j-ème ligne de H, $H\mathrm{x_i}$ s'écrit :

$$H\mathrm{x_i} = \begin{pmatrix} h^{(1)T}\mathrm{x_i} \\ h^{(2)T}\mathrm{x_i} \\ h^{(3)T}\mathrm{x_i} \end{pmatrix}$$

Puis :

$$\mathrm{x_i}' \wedge H\mathrm{x_i} = \begin{pmatrix} y_i' h^{(3)T}\mathrm{x_i} - w_i' h^{(2)T}\mathrm{x_i} \\ w_i' h^{(1)T}\mathrm{x_i} - x_i' h^{(3)T}\mathrm{x_i} \\ x_i' h^{(2)T}\mathrm{x_i} - y_i' h^{(1)T}\mathrm{x_i} \end{pmatrix}$$

Comme $h^{(k)T}\mathrm{x_i} = \mathrm{x_i}^T h^{(k)}$:

$$\mathrm{x_i}^{'} \wedge H\mathrm{x_i} = \begin{pmatrix} y_i' \mathrm{x_i}^T h^{(3)} - w_i' \mathrm{x_i}^T h^{(2)} \\ y_i' \mathrm{x_i}^T h^{(1)} - y_i' \mathrm{x_i}^T h^{(3)} \\ y_i' \mathrm{x_i}^T h^{(2)} - y_i' \mathrm{x_i}^T h^{(1)} \end{pmatrix}$$

Ce qui équivaut au système :

$$\begin{bmatrix} \begin{pmatrix} 0 & 0 & 0 \end{pmatrix} & -w_i' \mathrm{x_i}^T & y_i' \mathrm{x_i}^T \\ w_i^{'} \mathrm{x_i}^T & \begin{pmatrix} 0 & 0 & 0 \end{pmatrix} & -x_i' \mathrm{x_i}^T \\ x_i' \mathrm{x_i}^T & -y_i' \mathrm{x_i}^T & \begin{pmatrix} 0 & 0 & 0 \end{pmatrix} \end{bmatrix} \begin{pmatrix} h^{(1)} \\ h^{(2)} \\ h^{(3)} \end{pmatrix} = 0$$

Notons que les $w_i' \mathrm{x_i}^T, x_i' \mathrm{x_i}^T y_i' \mathrm{x_i}^T$ sont des vecteurs à 3 colonnes, et qu'il s'agit donc d'une matrice 3×9.

En notant A_i cette matrice, associée à la correspondance $\mathrm{x_i} \longleftrightarrow \mathrm{x_i}'$,

et h le vecteur $\begin{pmatrix} h^{(1)} \\ h^{(2)} \\ h^{(3)} \end{pmatrix}$ on obtient l'équation :

$A_i h = 0$ (I)

Notons L_1, L_2, L_3 les lignes de A_i, alors :
$x'_i L_1 + y'_i L_2 = \mathrm{x_i}^T \begin{pmatrix} y'_i w'_i & -w'_i x'_i & y'_i x'_i - x'_i y'_i \end{pmatrix}$

$= \mathrm{x_i}^T \begin{pmatrix} y'_i w'_i & -w'_i x'_i & 0 \end{pmatrix} = w'_i L_3$

La dernière ligne de A_i n'est donc pas nécessaire et on peut réécrire $A_i h = 0$ sous la forme :

$$\begin{bmatrix} \begin{pmatrix} 0 & 0 & 0 \end{pmatrix} & -w'_i \mathrm{x_i}^T & y'_i \mathrm{x_i}^T \\ w'_i \mathrm{x_i}^T & \begin{pmatrix} 0 & 0 & 0 \end{pmatrix} & -x'_i \mathrm{x_i}^T \end{bmatrix} \begin{pmatrix} h^{(1)} \\ h^{(2)} \\ h^{(3)} \end{pmatrix} = 0$$

où A_i est cette fois une matrice 2×9

Soit 2 équations linéairement indépendantes pour chaque $\mathrm{x_i} \longleftrightarrow \mathrm{x_i}'$.

En répétant l'opération 4 fois, on en obtient 8 qu'on rassemble dans une seule matrice notée A :

$$\mathrm{A} = \begin{bmatrix} A_1 \\ A_2 \\ A_3 \\ A_4 \end{bmatrix}$$

On réécrit alors le problème sous la forme :

$Ah = 0$ **(II)**

or A est une matrice 8×9, avec 8 lignes linéairement indépendantes ie A est de rang 8 et par le théorème du rang, son noyau est de dimension 1.

Il existe donc une solution h non triviale de (II), de laquelle on va déduire H.

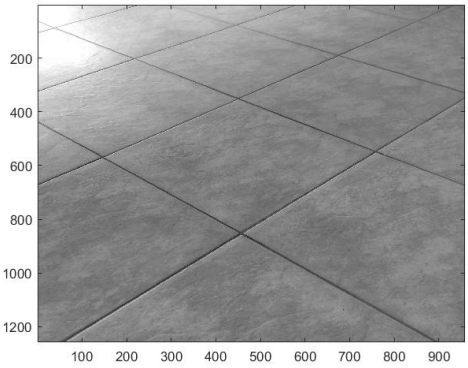
On détermine ensuite la SVD de A , c'est à dire $A = UDV^T$
Par définition de la SVD, la dernière colonne de V donne le noyau de A qui correspond ici à la solution (non triviale) recherchée.

La matrice H de l'homographie recherchée étant de taille 3×3, il est nécessaire de redimmensioner la solution h obtenue (vecteur à 9 lignes) : les 3 premières lignes de h correspondent à la première ligne de H et ainsi de suite.
Voici le code matlab correspondant :

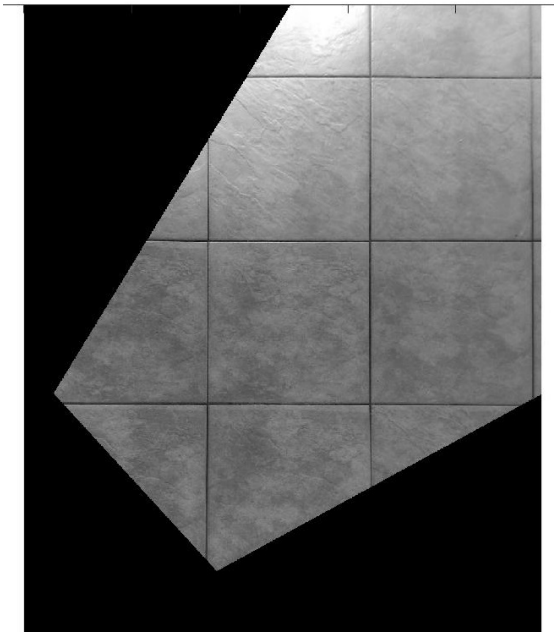
```
% En entrée : X1, X2
% on part de n correspondances de deux points en coord. homogènes
% les points connus sont stockés séparément dans deux matrices


function H = Dlt_s(X1,X2)

% A va contenir le système à résoudre
%n=4
k=0;
A=zeros(8,9);
for i=1:2:7 %on incrémente 2 par 2
    k = k+1;
    A(i,:)   = [zeros(1,3),  -X2(3,k)*X1(:,k)' , X2(2,k)*X1(:,k)'];
    A(i+1,:) = [X2(3,k)*X1(:,k)', zeros(1,3), -X2(1,k)*X1(:,k)']
    %on complète A 2 lignes par 2 lignes, 2lignes = 1 correspondance
end
[U,S,V] = svd(A);
h=V(:,9); %h != 0 solution de Ah = 0 ; est dans le noyau de A
H=reshape(h,3,3)' %on redimensionne h afin d'obtenir l'homographie
end
```

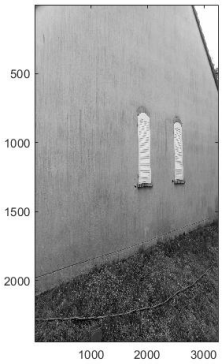


(a)

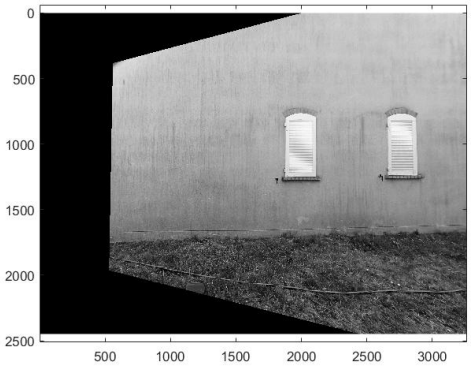


(b) Image rectifiée, on perçoit des défauts de pose du carrelage, non dûs à l'image (prouvé sur le terrain)

FIGURE 8 – Correction de distortion projective



(a)



(b)

FIGURE 9 – Autre exemple de correction. Notons qu'elle n'est pas parfaite à cause de l'erreur "manuelle" commise lors de la sélection, avec la souris, des points dans l'image de départ.

2. Normalisation

Dans le problème précèdent les x' étant choisis arbitrairement, il est clair que ce choix est dépendant du système de coordonées et donc que le résultat n'est pas invariant par similitude.
Autrement dit, certains systèmes de coordonnées sont meilleurs que d'autres pour la rectification d'image.
Avant d'appliquer le DLT à des données, il est donc important de normaliser les données.
Nous allons procéder d'abord par translation puis par scaling, après avoir obtenu un premier résultat par DLT on pourra ensuite exprimer le résultat dans le système de coordonnées de départ, ce qui permet de choisir un cadre "canonique".

Ainsi, en plus d'améliorer la précision du résultat, cette étape supplémentaire rend le DLT invariant par similitude.
1ère étape : translation

On envoie, pour chaque image, le centre de gravité des points (connus) sur l'origine, ce qui nous donne directement la translation correspondante et nous permet de l'appliquer à chacun des points.

2ème étape : *scaling*
Comme l'origine est invariante par multiplication, on peut diviser chacun des points par sa distance au centre de gravité, de sorte qu'ils sont tous à une distance de 1 de la nouvelle origine, puis multiplier par un scalaire arbitrairement choisi, ici $\sqrt(2)$.

La matrice correspondant à l'entièreté de la transformation s'obtient directement par composition des deux opérations.

Sans normalisation, pour des points en coordonnées homogènes, x et y peuvent prendre des valeurs très élevées, tandis que l'échelle vaut typiquement 1.

Si toutes les coordonnées changent par une transformation autre qu'une homotétie, par exemple dès qu'une estimation par les moindres carrés entre en jeu ou qu'on utilise un produit scalaire, la coordonnée d'échelle peut très rapidement atteindre des valeurs aberrantes et le passage subséquent en coordonnées classiques est alors entièrement faussé.

La normalisation est donc très importante, d'autant plus depuis que les résolutions d'image ont fortement augmenté.

Une fois obtenues les coordonnées des points sur chacune des images :
On applique la translation qui envoie O, le centre de gravité des n points, sur (0,0) (resp (0,0,0) en 3D).
Après translation, les 4 points sont toujours à la même distance d = $d(x_1, O)$, il suffit de multiplier par $\frac{\sqrt{(2)}}{\mathrm{d}}$ (resp$\frac{\sqrt{(3)}}{\mathrm{d}}$)

La matrice de translation s'écrit :

$$T_1 = \begin{pmatrix} 1 & 0 & -O_{11} \\ 0 & 1 & -O_1(2) \\ 0 & 0 & 1 \end{pmatrix}$$

De sorte que $T_1 x = \begin{pmatrix} x_1 - O_1 \\ x_2 - O_{12} \\ x_3 \end{pmatrix}$

Après translation, les 4 points sont toujours à la même distance d = $d(x_1, O)$, il suffit de multiplier par $\ell = \frac{\sqrt{(2)}}{\mathrm{d}}$ pour que les points soient à distance $\sqrt{2}$ de O.
La matrice associée est :

$$\begin{pmatrix} \ell & 0 & 0 \\ 0 & \ell & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Par composition du scaling et de la translation, on obtient finalement :

$$T = \begin{pmatrix} \ell & 0 & \ell(-O_{11}) \\ 0 & \ell & \ell(-O_1(2)) \\ 0 & 0 & 1 \end{pmatrix}$$

Notons que $det(T) = \ell^2 \neq 0$ donc T est bien inversible.
On applique d'abord le DLT sur les données normalisées pour déterminer $\widetilde{H}$ puis on réécrit :

$H = T_2^{-1} \widetilde{H} T_1$
Pour repasser au système de coordonnées initial.

Normalisation dans $\mathbb{P}^3$, on commence d'abord par translater le centre de gravité sur l'origine, puis on les multiplie afin qu'ils soient tous à une distance $\sqrt{3}$ de l'origine (de sorte que le point moyen a des coordonnées de magnitude $\begin{pmatrix}1\\1\\1\\1\end{pmatrix}$.

La normalisation pour les points de l'image est celle définie précédemment.
Pour les points de $\mathbb{P}^3$ on procède similairement, la seule différence étant qu'on veut que la distance à l'origine soit $\sqrt{3}$ plutôt que $\sqrt{2}$.
La matrice correspondante est :

$$T = \begin{pmatrix} \ell_{\sqrt{3}} & 0 & 0 & -\ell_{\sqrt{3}}O_1) \\ 0 & \ell_{\sqrt{3}} & 0 & -\ell_{\sqrt{3}}O_2 \\ 0 & 0 & \ell_{\sqrt{3}} & -\ell_{\sqrt{3}}O_3 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Code matlab de normalisation de points de l'espace, presque identique pour la normalisation de points du plan :

```
% ii) on normalise les points de l'espace (R^3)
%centroid
O_2 = mean(X_world(1:3,:))';
M = zeros(4,6);
for i=1:length(X_world)
        M(1,i) = X_world(1,i)-O_2(1);
        M(2,i) = X_world(2,i)-O_2(2);
        M(3,i) = X_world(3,i)-O_2(3);
        M(4,i) = 1

end
Origin_2 = [0 0 0]';
% on détermine la distance des points à l'origine après translation
d_init_world = norm(M(1,1));
% donc le scalaire associé à l'homotétie
% qui ramène les points à une distance de sqrt(3) de l'origine, est :
L_world =  sqrt(3)/d_init_world ;
% la composition de la translation et de l'homotétie nous donne l'application recherchée
T_world =  [ L_world 0 0 -L_world*O_2(1) ;
        0 L_world 0 -L_world*O_2(2);
        0 0 L_world -L_world*O_2(3);
        0 0 0 1];
%donc les données normalisées associées à l'image sont
Xworld_n = T_world*X_world;
```

3. Détermination de la matrice de caméra

On cherche à déterminer la matrice de caméra P.
Partons à nouveau d'un certain nombre de correspondances $x_i \longleftrightarrow X_i$, avec x_i des points de l'image et X_i des points du monde réel.
On cherche une matrice de caméra P de taille 3x4 telle que $x_i = PX_i$ pour tout i.

On cherche P sous la forme P = $\begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix}$

En notant $P^{(i)}$ les lignes de P et en procédant comme en (V.1), on obtient

$$\begin{bmatrix} \begin{pmatrix} 0 & 0 & 0 & 0 \end{pmatrix} & -w_i^{'}\mathrm{X_i}^T & y_i^{'}\mathrm{X_i}^T \\ w_i^{'}\mathrm{X_i}^T & \begin{pmatrix} 0 & 0 & 0 & 0 \end{pmatrix} & -x_i^{'}\mathrm{X_i}^T \end{bmatrix} \begin{pmatrix} P^{(1)} \\ P^{(2)} \\ P^{(3)} \end{pmatrix} = 0$$

pour chacune des correspondances, les X_i ayant 4 coordonnées on obtient une matrice 2x12 pour chaque itération, soit une matrice 2nx12 au total.

A nouveau, chaque correspondance $\mathrm{x_i} \longleftrightarrow \mathrm{X_i}$ compte pour 2 degrés de liberté.
P a 12 coefficients, donc 11 degrés de liberté en comptant l'échelle.
On a donc besoin de 11 equations ie $5+1/2$ correspondances suffisent puisqu'on peut se contenter d'utiliser une des deux équations fournies par la 6ème correspondance en utilisant la 1ère ou 2ème coordonnée de x_i.

Pour $n = 5\frac{1}{2}$ on a exactement 11 équations et la solution est exacte.
On l'obtient en résolvant $Ap = 0$ grâce à la SVD de A de la même manière qu'en II, le vecteur p étant le noyau non-trivial de A, puis en réorganisant les termes contenus dans p en une matrice 3x4.

Supposons qu'on ait $n \geq 6$, et en supposant qu'il y ait des erreurs de mesure, on cherche à obtenir P en minimisant l'erreur géométrique.
On suppose que les coordonnées du points du monde réel sont obtenus beaucoup plus précisément que les coordonnées des points de l'image. C'est notamment le cas si des mesures optimales ont été faites dans le monde réel, tandis que les points de l'image ont été récupérés en cliquant, comme c'est le cas lorsqu'on utilise la fonction *ginput* en Matlab.

Minimiser l'erreur géométrique revient alors à minimiser :
$\sum_i d(x_i, PX_i)^2$

On procède ensuite par itération en utilisant la fonction Matlab *lsqcurvefit* (cf Annexe 2) avec l'algorithme de Levenberg-Marquardt en choisissant comme point de départ l'estimation obtenue dans un premier temps par DLT.
Le code est majoritairement le même que pour les exemples précédents, seule la dernière partie introduit de nouveaux éléments :

```
P_linear_estimate = DLT_2_3d(Xworld_n,Xpic_n);
P_double = double(P_linear_estimate);

% III on minimise l'erreur géométrique, méthode choisie : Levemberg-marquardt
f_arg = @(x,xdata) x*xdata;
% LSQCURVEFIT
options = optimoptions('lsqcurvefit','Algorithm','levenberg-marquardt');
%le point de départ de l'estimation est le résultat obtenu via DLT
P_estimate = lsqcurvefit(f_arg, P_double, Xworld_n , Xpic_n);
%on repasse en coordonnées initiales
Camera_matrix = inv(T_pic)*P_estimate*T_world;
disp(Camera_matrix);
%résultats d'éxécution
Local minimum found.

Optimization completed because the size of the gradient is less than
the value of the optimality tolerance.

<stopping criteria details>
   1.0e+03 *

    0.0035    0.0000   -0.0002    1.8713
   -0.0002   -0.0006   -0.0038    1.7591
    0.0000    0.0000   -0.0000    1.0000
```

Où P est la matrice 3×4 qu'on obtient une fois l'optimisation terminée.

VI. Image de la conique absolue et paramètres internes de caméra

Deux images obtenues avec le même centre de caméra sont projectivement équivalentes. De plus, les images d'éléments sur le plan à l'infini sont indépendantes de la position de la caméra et dépendent seulement de la rotation de la caméra et des paramètres internes de la caméra donnés par K.

L'image du conique absolu ω est invariante par rotation de la caméra et dépend uniquement de K.
Il est donc possible de déterminer K lorsqu'on connaît ω, qu'on peut obtenir en utilisant des caractéristiques externes de l'image comme la présence de points de fuite correspondant à des directions orthogonales, ou internes : *aspect ratio* des pixels qui vont fournir des contraintes sur ω.
Une fois ω, matrice symétrique définie positive, connue, on peut la décomposer comme $\omega = (KK^t)^{-1} = K^{-t}K^{-1}$ en utilisant sa décomposition de Cholesky puis inverser le résultat.

Notons qu'il est possible dans certains cas de déterminer ω sans qu'elle soit définie positive (à cause de données approximatives) et il est alors impossible d'en déduire K.
On a montré précédemment qu'il suffisait de 5 contraintes pour déterminer une conique. On s'intéresse à un cas particulier nous permettant de déterminer directement ω.
Considérons une image quelconque, comportant des lignes de fuite orthogonales :

1. Contraintes sur ω

Deux points de fuite de l'image correspondant à des directions orthogonales vérifient :
$x_1^T \omega x_2 = 0$
on en déduit une contrainte sur ω.
Dans le cas où le point principal $p = (u_0, v_0)^T$ est situé au centre de l'image, p vérifie :
$p = K(0,0,1)^T = (u_0, v_0, 1)^T$
car le centre de ω est translaté de l'origine à p, ainsi ω envoie p sur l_∞, qui a pour position canonique $l_\infty = (0,0,1)^T$. On en déduit la contrainte :
$\omega p = (0,0,1)^T$ qu'on réécrit :
$u_0\omega_1 + v_0\omega_2 + \omega_4 = 0$
$u_0\omega_2 + v_0\omega_3 + \omega_5 = 0$
ce qui correspond à deux contraintes sur ω.
Les deux dernières contraintes s'obtiennent sous les hypothèses suivantes :
- $s = K_{12} = 0$ ie la distortion dûe à la caméra est nulle.

- les pixels sont carrés : le rapport largeur/hauteur est de 1.
On déduit de la première contrainte $\omega_{12} = \omega_{21}$ et de la deuxième $K_{11} = K_{12}$ soit $\omega_{11} = \omega_{22}$.

On écrit les contraintes sous la forme $a^T w = 0$ avec $w = (w_1, w_2, w_3, w_4, w_5, w_6)$:

- la contrainte pour deux points de fuite orthogonaux s'écrit $[v(1)u(1), v(1)u(2)+ v(2)u(1), v(2)u(2), v(1)u(3) + v(3)u(1), v(2)u(3) + v(3)u(2), v(3)u(3)]w = 0$

- point principal au centre de l'image :
$$\begin{cases} [u_0, v_0, 0, 1, 0, 0]w &= 0 \\ [0, u_0, v_0, 0, 1, 0]w &= 0 \end{cases}$$
- $s = 0$ et les pixels sont carrés :
$$\begin{cases} [0, 1, 0, 0, 0, 0]w &= 0 \\ [1, 0, -1, 0, 0, 0]w &= 0 \end{cases}$$

FIGURE 10 – Les lignes de fuite correspondant à une même direction s'intersectent en un point de fuite

2. Extraction des paramètres internes

ω s'écrit $\omega = \begin{pmatrix} w_1 & w_2 & w_4 \\ w_2 & w_3 & w_5 \\ w_4 & w_5 & w_6 \end{pmatrix}$

De la même manière que précédemment, chaque contrainte donne lieu à une ou deux équations, qu'on rassemble dans A 5 by 6, la matrice des contraintes A telle que $Aw = 0$ est la suivante :

$$\begin{pmatrix} v(1)u(1) & v(1)u(2)+v(2)u(1) & v(2)u(2) & v(1)u(3)+v(3)u(1) & v(2)u(3)+v(3)u(2) & v(3)u(3) \\ p_1 & p_2 & 0 & 1 & 0 & 0 \\ 0 & p_1 & p_2 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 & 0 \end{pmatrix}$$

Qu'on résout à nouveau par SVD.
On extrait ensuite de w les coefficients de la matrice ω, qu'on réécrit sous la forme d'une matrice 3×3.

Une fois ω déterminée, rappelons que sa factorisation de Cholesky donne :
$\omega = (KK^t)^{-1} = K^{-t}K^{-1}$ donc $\text{chol}(\omega) = K^{-1}$ en matlab et $inv(chol(\omega)) = K$.
Code matlab associé et résultats d'éxécution :

```
function K = Principal_point(u,v,p)
% - u,v vanishing points corresponding to orthogonal lines
% 1 contrainte associée
%  - known principal point p
% 2 contraintes associées
% square pixels + zero skew ; 2 contraintes
M = zeros(5,6);
M(1,:) = [ v(1)*u(1), v(1)*u(2) + v(2)*u(1) , v(2)*u(2) , v(1)*u(3)+v(3)*u(1) , v(2)*u(3)+v(
M(2,:) =  [ p(1), p(2) , 0 , 1 ,0 ,0];
M(3,:)  =  [0, p(1), p(2), 0 ,1 ,0 ];
M(4,:) = [0 , 1 ,0 ,0, 0 ,0];
M(5,:)  = [ 1, 0, -1, 0, 0 ,0];

[U,S,V] = svd(M);
w=V(:,6); %o != 0 ; est dans le noyau de M
omega= [ w(1) w(2) w(4) ; w(2) w(3) w(5) ; w(4) w(5) w(6) ];
disp(omega);
K_inv = chol(omega);
K = inv(K_inv);

end
```

Lorsque ω est définie positive on peut obtenir K :

```
0.0000    0.0000   -0.0002
    0.0000    0.0000   -0.0001
   -0.0002   -0.0001    1.0000


K =

   1.0e+03 *

    1.6500   -0.0000    0.5270
         0    1.6500    0.3834
         0         0    0.0011
```

Notons que la forme générale de K est respectée, bien que $K(1,2)$ ne soit pas parfaitement nul et que $K(3,3)$ pas exactement égal à 1 : c'est dû aux erreurs faites lors du calcul des points de fuite.

3. Paramètres internes de caméra et homographie

La connaissance de K permet de déterminer certaines homographies, notamment l'homographie dite *infinie*, obtenue directement par :
$H_\infty = KRK^{-1}$ où R est une rotation.
H_∞ ne dépend pas de translations d'une vue à l'autre mais seulement de la rotation et des paramètres internes de la caméra.

Cette homographie correspond à une rotation dite "synthétique" de la caméra par rapport à son centre et joue aussi un rôle majeur dans la calibration et l'autocalibration de caméra. (Pollefeys and Van Gool, 1999), (Köser, Beder and Koch, 2008),(Jun-Sik and Soo-Ok, 2006)

VII. Annexe

1. SVD et Factorisation de Cholesky

Décomposition en valeurs singulières ou SVD :

Soit M une matrice réelle $m \times n$. Les valeurs singulières de M sont les racines carrées des valeurs propres de la matrice $M^T M$.

Soit $A \in \mathbb{M}_{m\times n}(\mathbb{R})$. Alors A se décompose en :
$A = U \sum V^T$
avec $U \in \mathbb{M}_{m\times m}, V \in \mathbb{M}_{n\times n}$ deux matrices unitaires, et $\sum \in \mathbb{M}_{m\times n}$ une matrice diagonale correspondant aux valeurs singulières de A.
Cette décomposition est appelée SVD ou décomposition en valeurs singulières.

On l'obtient en matlab par la fonction svd(), notamment :
$[U, \sum, V^T] = svd(A)$
donne directement $U, \sum, V^T$ tels que définis ci-dessus.
Propriétés :
Notons r le rang de A. Alors :
- les r premières colonnes de V correspondent à l'espace engendré par les lignes de A
- les n-r dernières colonnes de V correspondent au noyau de A
- les r premières colonnes de U correspondent à l'espace engendré par les colonnes de A
- les m-r dernières colonnes de V correspondent au noyau de A^T

La deuxième propriété s'avère particulièrement utile puisqu'elle nous donne directement les solutions non triviales des systèmes linéaires de la forme $AX = B$ lorsque A n'est pas carrée.

Factorisation de Cholesky :
Soit A une matrice symétrique définie positive.
Alors il existe une matrice triangulaire inférieure telle que :
$A = RR^T$
En Matlab, $R = chol(A)$ nous donne R tel que $A = R^T R$.

2. Matlab

Grid Data :
vq = griddata(x,y,v,xq,yq)

griddata prend en entrée un échantillon de points x,cette fonction interpole le plan (en 2D) en les points donnés par (xq,yq) et renvoie les valeurs interpolées vq.
Ce plan passe nécessairement par les données indiquées par x et y.
Les points sont définis en coordonnées (x, y) lorsqu'on est en 2D.
Grid data, interpole les données en des points pré-définis sur une grille.
Notons que l'enveloppe convexe des points est d'abord calculée et que l'interpolation est réalisée seulement lorsque les données à interpoler appartiennent à cette enveloppe, sans quoi l'interpolation en ce point donne NaN : not a number.

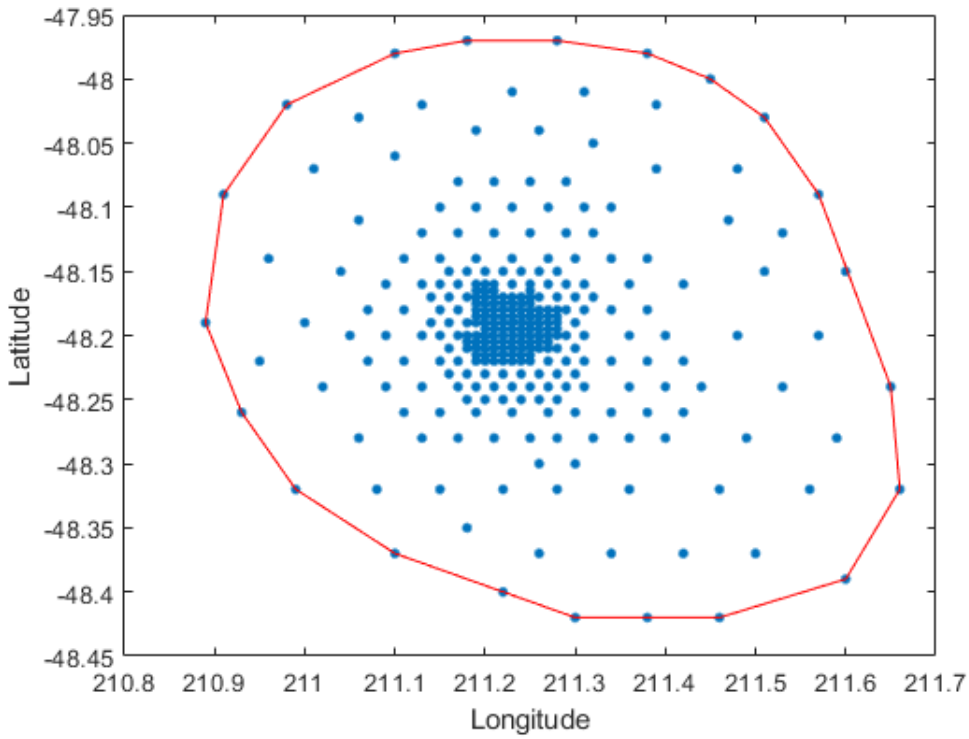


FIGURE 11 – Enveloppe convexe d'un nuage de points

LeastSquareCurveFit :

x = lsqcurvefit(fun,x0,xdata,ydata)

Détermine la solution x, au sens des moindres carrés non linéaires, du problème suivant :
$\min_x ||F(x, xdata) - ydata||_2^2 = \min_x \sum_i ((F(x, xdata_i) - ydata_i)^2$
Étant donnés des données $x$data, et des observations ydata, avec $x$data et ydata des vecteurs ou matrices, où F (x, $x$data) est de même taille que ydata.
Dans l'usage qu'on en fait ici, la fonction correspondant à 'fun' est d(x_i, PX_i).

On peut choisir 'lb' et 'ub' comme borne inférieure/supérieure respectivement pour les éléments de x, ces arguments peuvent être des vecteurs ou matrices.
L'algorithme est itératif. On part d'un paramètre initial, que l'on suppose « assez proche » d'un minimum prédifini, et qui constituera le vecteur de départ.
La spécification 'levenberg-marquardt' indique la méthode d'itération utilisée.
L'algorithme de Levenberg-Marquardt peut être vue comme une variation de la méthode de Gauss-Newton.
Ses itérations successives suivent les étapes indiquées ci-dessous :

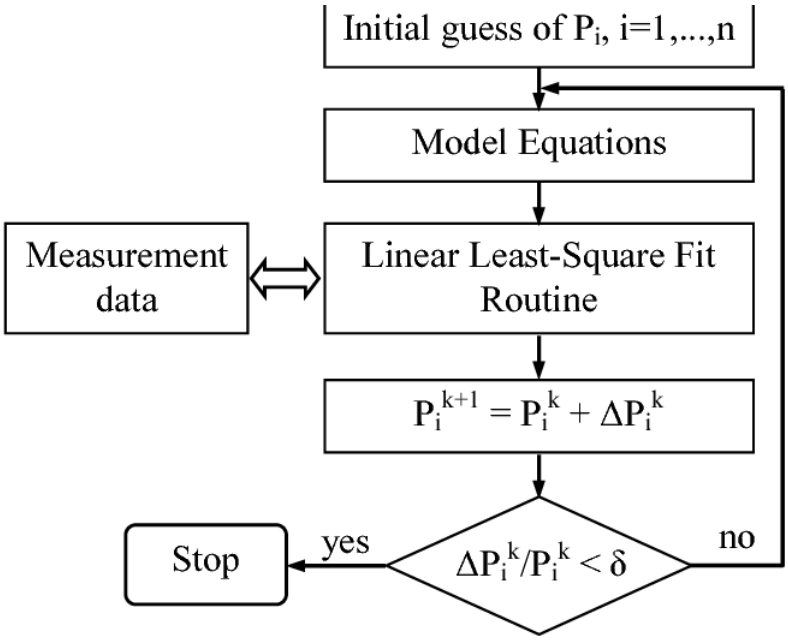


Fig. 2. Flow of optimization with Levenberg-Marquardt algorithm

VIII. Bibliographie

Hartley, R. and Zisserman, A., 2003. Multiple view geometry in computer vision. Cambridge : Cambridge University Press.

Leibowitz, D., 2001. Camera Calibration and Reconstruction of Geometry from Images. Robotics Research Group Department of Engineering Science University of Oxford.

https ://mathworks.com

Pollefeys, M. and Van Gool, L., 1999. Stratified Self-Calibration with the Modulus Constraint. 21st ed. IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE,.

Köser, K., Beder, C. and Koch, R., 2008. Conjugate Rotation : Parameterization and Estimation from an Affine Feature Correspondence. [online] Available at : https ://www.researchgate.net/publication/221364079

Jun-Sik, K. and Soo-Ok, K., 2006. Infinite Homography Estimation Using Two Arbitrary Planar Rectangles. [online] Available at : https ://www.researchgate.net/publication/22074502