Vision 3D

Notes écrites par
Yoann Coudert–Osmont
ycoudert@ens-paris-saclay.fr

D'après un cours de
Pascal Monasse, Mathieu Aubry et Renaud Marlet
ENPC

11 novembre 2019

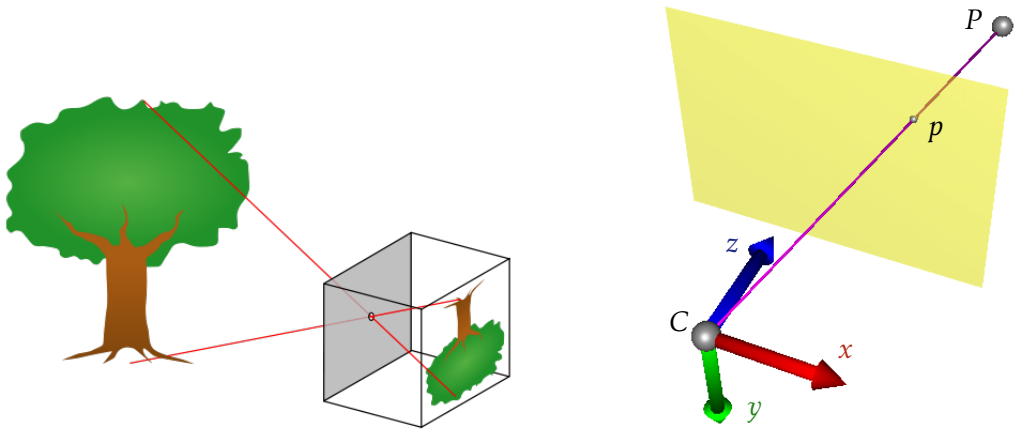
Table des matières

Chapitre A

Caméra et Projection

Table des matières

I Caméra sténopé (*pinhole*)



On se place dans le modèle idéal où l'ouverture de la caméra est réduite à un point. On ignore toute distorsion géométrique et tout flou qui pourrait par exemple être causé par la diffraction. Afin de ne pas avoir une image retournée comme dans les caméras physiques on place le plan de projection devant le point C de la caméra. La direction $\vec{z}$ est la direction dans laquelle regarde la caméra. Les deux autres axes sont parallèles aux bordures de notre de projection.

Le point de projection p se trouve sur la droite passant par C et P. On écrit :

$$\vec{Cp} = \lambda \vec{CP}$$

Le plan de projection est placé à la profondeur f correspondant à la focale. On a donc :

$$\begin{pmatrix} x \\ y \\ f \end{pmatrix} = \lambda \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$

On en déduit alors l'égalité $\lambda = f/Z$. Par la suite on travaille avec une grille de pixels située sur le plan de projection. On note alors $c = (c_x, c_y)$ les coordonnées de la projection de la droite $C\vec{z}$ sur cette grille. On introduit aussi un facteur d'échelle α pour passer de la distance en mètres à la distance en pixels, et on obtient :

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \alpha x + c_x \\ \alpha y + c_y \end{pmatrix} = \begin{pmatrix} (\alpha f)X/Z + c_x \\ (\alpha f)Y/Z + c_y \end{pmatrix}$$

II Géométrie projective

Comme deux points alignés avec C ont la même projection on défini naturellement la relation d'équivalence $\mathcal{R}$ suivante :

$$x \mathcal{R} y \Leftrightarrow \exists \lambda \neq 0, x = \lambda y$$

Cette relation permet alors de définir le plan projectif :

$$\mathbb{P}^2 = \left(\mathbb{R}^3 \setminus \{0\}\right)/\mathcal{R}$$

On représentera souvent les vecteurs de $\mathbb{P}^2$ par des vecteurs à trois dimensions dont la troisième composante vaut 1. Les points de notre projections qui ne peuvent pas s'exprimer de la sorte sont des points situés à l'infini. Ils ont une troisième composante nulle.

$$\begin{pmatrix} x & y & \epsilon \end{pmatrix} = \begin{pmatrix} x/\epsilon & y/\epsilon & 1 \end{pmatrix} \xrightarrow[\epsilon \to 0]{} \begin{pmatrix} x & y & 0 \end{pmatrix}$$

1/ Lignes et intersections

Un plan de $\mathbb{R}^3$ passant par 0 à un projection qui est une ligne dans $\mathbb{P}^2$. Un plan dans $\mathbb{R}^3$ et donc une ligne dans $\mathbb{P}^2$ est représenté par son vecteur orthogonal (a, b, c) qui vérifie l'équation :

$$\begin{pmatrix} a & b & c \end{pmatrix}^{\mathsf{T}} \begin{pmatrix} X & Y & Z \end{pmatrix} = 0$$

La ligne passant par les deux points x_1 et x_2 est définie par :

$$l = x_1 \times x_2 \qquad \text{en effet } (x_1 \times x_2)^{\mathsf{T}} x_i = \begin{vmatrix} x_1 & x_2 & x_i \end{vmatrix} = 0$$

L'intersection des deux lignes l_1 et l_2 est définie par :

$$x = l_1 \times l_2 \qquad \text{en effet } l_i^{\mathsf{T}} (l_1 \times l_2) = \begin{vmatrix} l_i & l_1 & l_2 \end{vmatrix} = 0$$

Les points à l'infini forment une ligne à l'infini représentée par :

$$l_\infty = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \qquad \text{en effet } l_\infty^\mathsf{T} \begin{pmatrix} x \\ y \\ 0 \end{pmatrix} = 0$$

Autre point intéressant : les lignes "parallèles" s'intersectent à l'infini :

$$\begin{pmatrix} a \\ b \\ c_1 \end{pmatrix} \times \begin{pmatrix} a \\ b \\ c_2 \end{pmatrix} = (c2 - c_1) \begin{pmatrix} b \\ -a \\ 0 \end{pmatrix} \in l_\infty$$

2/ Matrice de calibration

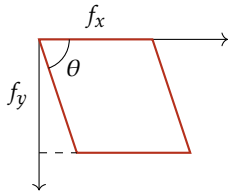
Pour simplifier la suite, on remplace αf par f. On a alors :

$$\begin{pmatrix} u \\ v \end{pmatrix} = \frac{1}{Z} \begin{pmatrix} fX + c_x Z \\ fY + c_y Z \end{pmatrix} \quad \Leftrightarrow \quad p = Z \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} f & & c_x \\ & f & c_y \\ & & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$

La **matrice de calibration** est alors la matrice K qui apparaît ci-dessous.

$$K = \begin{pmatrix} f & & c_x \\ & f & c_y \\ & & 1 \end{pmatrix}$$

Si les pixels ne sont pas des rectangles mais des parallélogrammes alors on obtient une matrice de calibration différente :



$$K = \begin{pmatrix} f_x & s & c_x \\ & f_y & c_y \\ & & 1 \end{pmatrix} \quad \text{avec } s = -f_x \operatorname{cotan} \theta$$

La caméra pouvant être translaté on introduit les matrices de translation qui nécessite de passer à la 4ème dimension :

$$\begin{pmatrix} X + t_x \\ Y + t_y \\ Z + t_z \end{pmatrix} = \begin{pmatrix} 1 & & & t_x \\ & 1 & & t_y \\ & & 1 & t_z \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = TP$$

La caméra peut aussi subir une rotation. Les rotations sont représentées par les matrices orthogonales de déterminant 1. On notera R les matrices de rotations. La **matrice de projection** est alors le produit d'une translation suivie d'une rotation suivie de la matrice de calibration :

$$P = K \begin{pmatrix} R & T \end{pmatrix}$$

Proposition 1

SI P est une matrice 3×4 dont la sous-matrice 3×3 de gauche est inversible alors il existe une unique décomposition :

$$P = K \begin{pmatrix} R & T \end{pmatrix}$$

Démonstration On pose $P = \begin{pmatrix} A & P_4 \end{pmatrix}$. Appliquer l'orthonormalisation de Gram-Schmidt sur les lignes de A conduit à ce qu'on appelle la décomposition RQ où R signifie une matrice triangulaire supérieure et Q une matrice orthogonale. Cela nous donne donc $A = KR$. On se retrouve alors avec $T = K^{-1}P_4$. ■

3/ Perspective

On s'intéresse à toutes les droites parallèles de direction d dans l'espace. Considérons la droite de direction d et passant par P. Alors :

$$K(P + \lambda d) = KP + \lambda Kd$$

Ainsi la projection de la droite est représentée par $l_P = (KP) \times (Kd)$. On pose alors le point $v = Kd$. Ainsi pour tout $Q \in R^3$ on a $l_Q^\mathsf{T} v = 0$. Cela veut dire que toutes les droites de directions d s'intersecte en un seul point v. Ce point est appelé un **point de fuite**.

Si on considère deux direction d_1 et d_2. On obtient deux points de fuites $v_1 = Kd_1$ et $v_2 = Kd_2$. Alors le point de fuite des lignes de directions $\alpha d_1 + \beta d_2$ est $\alpha v_1 + \beta v_2$ qui appartient à la ligne $v_1 \times v_2$. Si d_1 et d_2 sont des directions horizontales alors la ligne $v_1 \times v_2$ est appelée **ligne de fuite**. Cette ligne correspond à l'horizon.

III Homographies

Rotation de caméra Une scène peut être prise en photos sous plusieurs angles/positions. Un point de la scène peut alors avoir les coordonnées $\mathbf{x}$ dans une photos et les coordonnées $\mathbf{x}'$ dans une seconde. Par exemple si on fait simplement une rotation de la caméra on obtient la relation suivante :

$$\mathbf{x}' = K'RK^{-1}\mathbf{x} = H\mathbf{x}$$

La matrice K^{-1} permet de retourner dans les coordonnées de l'espace. La matrice R fait une rotation dans l'espace puis on reprojette notre point qui a subit une rotation avec la matrice K' qui peut être différente de K si entre temps les paramètre de l'appareil d'acquisition ont changé.

Photo d'un plan Une homographie n'est pas nécessairement une transformation d'une projection à une autre, mais plus généralement d'un plan à un autre. Par exemple si on observe le plan $Z = 0$ alors P est de la forme $\begin{pmatrix} X & Y & 0 & 1 \end{pmatrix}^\mathsf{T}$ (la quatrième coordonnée est ajoutée pour les translation). P peut alors être contracté en $\mathbf{x} = \begin{pmatrix} X & Y & 1 \end{pmatrix}^\mathsf{T}$. On peut alors obtenir les coordonnées du point P sur une photo prise de n'importe qu'elle position dans l'espace avec n'importe quelle rotation :

$$\mathbf{x}' = K\begin{pmatrix} R_1 & R_2 & R_3 & T \end{pmatrix}P = K\begin{pmatrix} R_1 & R_2 & T \end{pmatrix}\mathbf{x} = H\mathbf{x}$$

Les deux cas précédent font apparaître une matrice H qui réalise une transformation linéaire des coordonnées dans un plan vers les coordonnées dans un autre plan. Une telle transformation est appelée une **homographie**. Avec :

$$\mathbf{x} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \qquad \mathbf{x}' = \lambda \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} \qquad H = \begin{pmatrix} h_1^\mathsf{T} \\ h_2^\mathsf{T} \\ h_3^\mathsf{T} \end{pmatrix}$$

On obtient le système :

$$\begin{cases} \lambda x' &= h_1^\top \mathbf{x} \\ \lambda y' &= h_2^\top \mathbf{x} \\ \lambda &= h_3^\top \mathbf{x} \end{cases} \Rightarrow \begin{cases} x' &= \dfrac{h_1^\top \mathbf{x}}{h_3^\top \mathbf{x}} = \dfrac{h_{1,1}x + h_{1,2}y + h_{1,3}}{h_{3,1}x + h_{3,2}y + h_{3,3}} \\ y' &= \dfrac{h_2^\top \mathbf{x}}{h_3^\top \mathbf{x}} = \dfrac{h_{2,1}x + h_{2,2}y + h_{2,3}}{h_{3,1}x + h_{3,2}y + h_{3,3}} \end{cases}$$

On remarque finalement que γH résulte en la même transformation que H si $\gamma \in \mathbb{R}^*$. On peut alors poser $h_{3,3} = 1$. L'espace des homographies est alors de dimension 8 et la données de 4 paires de points $(\mathbf{x}, \mathbf{x}')$ avec $\mathbf{x}' = H\mathbf{x}$ permet de retrouver l'homographie grâce aux équations :

$$\begin{cases} x' &= \dfrac{h_{1,1}x + h_{1,2}y + h_{1,3}}{h_{3,1}x + h_{3,2}y + 1} \\ y' &= \dfrac{h_{2,1}x + h_{2,2}y + h_{2,3}}{h_{3,1}x + h_{3,2}y + 1} \end{cases} \tag{A.1}$$

Le théorème suivant est une généralisation de l'unicité d'une homographie étant donné un certain nombre de points.

Proposition 2 (Unicité d'homographie)

Étant donné les vecteurs $e_1, ..., e_{d+1}$ et $f_1, ..., f_{d+1}$ dans $\mathbb{R}^d$ tels que pour tout $i \in \{1, ..., d+1\}$, les familles $\{e_j\}_{j \neq i}$ et $\{f_j\}_{j \neq i}$ sont des bases de $\mathbb{R}^d$. Alors il existe un unique isomorphisme H (à un facteur multiplicatif près) et des scalaires uniques λ_i tel que pour tout i, $He_i = \lambda_i f_i$. Ainsi étant donné $(n+2)$ paires $(x_i, x_i') \in \mathbb{P}^n$ il existe une unique homographie associant les x_i' aux x_i.

Démonstration $(e_1, ..., e_d)$ et $(f_1, ..., f_d)$ sont des base de $\mathbb{R}^d$. On peut donc écrire de manière unique les vecteurs e_{d+1} et f_{d+1} dans ces bases :

$$e_{d+1} = \sum_{i=1}^{d} \mu_i e_i \qquad f_{d+1} = \sum_{i=1}^{d} \nu_i f_i$$

On obtient donc :

$$\sum_{i=1}^{d} \nu_i \lambda_{d+1} f_i = \lambda_{d+1} f_{d+1} = H e_{d+1} = \sum_{i=1}^{d} \mu_i H e_i = \sum_{i=1}^{d} \mu_i \lambda_i f_i$$

D'où les égalités :

$$\forall i = 1, ..., d \ : \quad \mu_i \lambda_i = \nu_i \lambda_{d+1}$$

Or si $\mu_i \neq 0$ alors la famille $\{e_j\}_{j \neq i}$ est linéairement dépendante. C'est une contradiction avec nos hypothèses donc tous les μ_i et ν_i sont non nuls. Cela nous donne donc l'égalité :

$$\forall i = 1, ..., d \ : \quad \lambda_i = \frac{\nu_i}{\mu_i} \lambda_{d+1}$$

On peut alors fixer $\lambda_{d+1} = 1$ comme l'unicité est à un facteur multiplicatif près. Et alors il existe une unique matrice H qui envoie la base $\{e_i\}_{1 \leqslant i \leqslant d}$ sur la base $\{\lambda_i f_i\}_{1 \leqslant i \leqslant d}$. ■

Panorama On se donne un ensemble de n paires de points $(\mathbf{x}_i, \mathbf{x}'_i)_{1 \leqslant i \leqslant n}$ sur deux photos d'une même scène. On suppose que la transformation des x_i vers les x'_i est une homographie. Par exemple la caméra a subi une rotation ou bien on observe un plan. On reprend alors l'équation A.1.

$$\begin{cases} x'_i &= h_{1,1}x + h_{1,2}y + h_{1,3} - h_{3,1}x_i x'_i - h_{3,2}y_i x'_i \\ y'_i &= h_{2,1}x + h_{2,2}y + h_{2,3} - h_{3,1}x_i y'_i + h_{3,2}y_i y'_i \end{cases}$$

On a donc un système linéaire : $A_i h = \mathbf{x}'_i$. Où :

$$h = \begin{pmatrix} h_{1,1} & h_{1,2} & \dots & h_{3,2} \end{pmatrix} \qquad A_i = \begin{pmatrix} x_i & y_i & 1 & 0 & 0 & 0 & -x_i x'_i & -y_i x'_i \\ 0 & 0 & 0 & x_i & y_i & 1 & -x_i y'_i & -y_i y'_i \end{pmatrix}$$

Cela donne finalement le système : $Ah = \mathbf{x}'$. Avec :

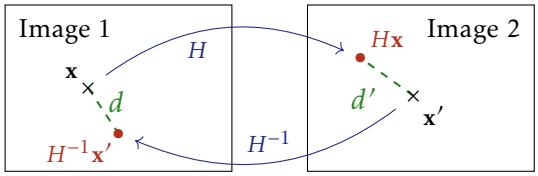
$$A = \begin{pmatrix} A_1 \\ \vdots \\ A_n \end{pmatrix} \qquad \mathbf{x}' = \begin{pmatrix} \mathbf{x}'_1 \\ \vdots \\ \mathbf{x}'_n \end{pmatrix}$$

Lorsque n est strictement plus grand que 4, on se retrouve avec un problème d'optimisation car il est possible qu'aucun vecteur h ne vérifie le système. Voici alors plusieurs fonctions possibles à optimiser :

— L'erreur quadratique algébrique (qui n'a pas de sens géométrique) :

$$\min_h \|Ah - \mathbf{x}'\|^2$$

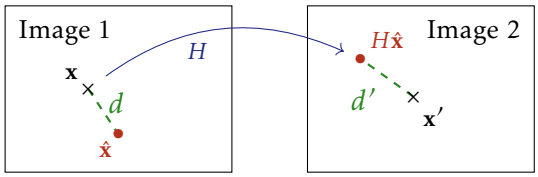
— Une autre erreur qui a plus de sens géométrique est l'erreur de transfert :



L'erreur de transfert est alors : $\min_H d'^2 = d(\mathbf{x}', H\mathbf{x})^2$.
Cette erreur peut-être symétrisé par :

$$\min_H \; d^2 + d'^2 = d(\mathbf{x}, H^{-1}\mathbf{x}')^2 + d(\mathbf{x}', H\mathbf{x})^2$$

— On peut aussi supposer que nos points $\mathbf{x}$ et $\mathbf{x}'$ sont des observations bruités. On peut alors rajouter de nouveaux paramètres à optimiser qui seraient la vrai position $\hat{\mathbf{x}}$.



Ce qui nous donne :

$$\min_{H,\hat{\mathbf{x}}} \; d(\mathbf{x}, \hat{\mathbf{x}})^2 + d(\mathbf{x}', H\hat{\mathbf{x}})^2$$

Chapitre B

Géométrie épipôlaire

Table des matières

Rappel sur le produit vectoriel Le produit vectoriel de deux vecteurs est défini par :

$$a \times b = [a]_\times b = \begin{pmatrix} yz' - zy' \\ zx' - xz' \\ xy' - yx' \end{pmatrix} \qquad \text{Avec } [a]_x = \begin{pmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{pmatrix}$$

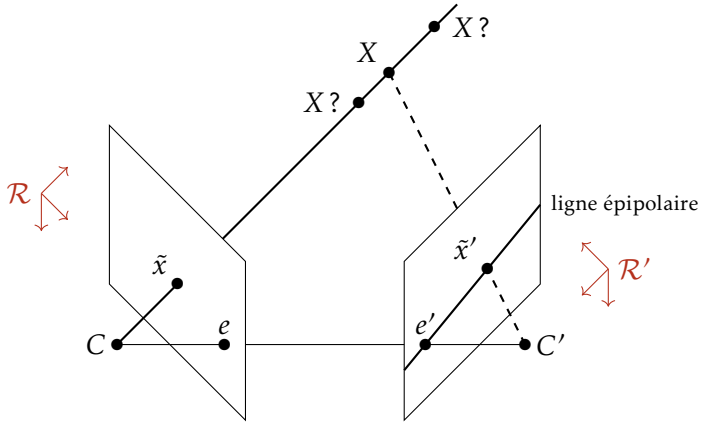
Ce produit est bilinéaire et antisymétrique. De plus il a un lien avec le déterminant puisque :

$$a^\mathsf{T}(b \times c) = \begin{vmatrix} a & b & c \end{vmatrix}$$

On a aussi la propriété de composition suivante :

$$(a \times b) \times c = (a^\mathsf{T} c)b - (b^\mathsf{T} c)a$$

I Matrices essentielle et fondamentale



On observe un point X à partir de deux caméras de centres C et C'. Les trois points X, C, et C' définissent un plan qui contient les trois vecteurs $\vec{Cx}$, $\vec{C'x'}$, et $\vec{T} = \vec{CC'}$. Pour simplifier on va se placer le repère de l'espace $\mathcal{R}$. On note x les coordonnées de $\vec{Cx}$, et T les coordonnées de $\vec{T}$ dans ce repère. Finalement on note x' les coordonnées de $\vec{C'x'}$ dans le repère $\mathcal{R}'$. En appelant R, la matrice de rotation pour passer du repère $\mathcal{R}'$ au repère $\mathcal{R}$, on a que les trois points x, T et Rx' forment une famille liée de $\mathbb{R}^3$. C'est à dire :

$$\begin{vmatrix} x & T & Rx' \end{vmatrix} = x^\top (T \times Rx') = x^\top [T]_\times Rx' = 0$$

On défini alors la **matrice essentielle** par :

$$E = [T]_\times R \qquad \text{tel que } x^\top E x' = 0$$

Cette matrice est intéressante mais elle traite des points de l'espace. Or on manipule des images et on a pas accès directement à x et x' mais plutôt aux coordonnées projectives $y = Kx$ et $y' = K'x'$ de ces deux points dans leurs images respectives. On défini alors la **matrice fondamentale** suivante :

$$F = \left(K^{-1}\right)^\top E K'^{-1} = \left(K^{-1}\right)^\top [T]_\times R K'^{-1} \qquad \text{tel que } y^\top F y'$$

Observations

— Le point e est l'**épipôle** de la première caméra. Ses coordonnées projectives pour la première caméra sont définies par :

$$e = KT \qquad \text{tel que } e^\top F = 0$$

— Le point e' est l'épipôle de la seconde caméra. Ses coordonnées projectives pour la seconde caméra sont définies par :

$$e' = K'R^{-1}T \qquad \text{tel que } Fe' = 0$$

— La ligne Fx' définie la **ligne épipôlaire** de la première image. En effet si x est un point de la première image résultant de la projection d'un point X qui se projette aux coordonnées x' sur la seconde image alors $x^\top F x' = 0$.

— La ligne $F^\top x$ définie la ligne épipôlaire de la seconde image.

— Finalement on peut remarquer que si $T = 0$ alors $F = 0$. Sans déplacement on ne peut alors tirer aucune information de F.

Proposition 3

Une matrice 3×3 définie une matrice fondamentale si et seulement si son rang vaut 2.

II Calculs des matrices

Proposition 4 (SVD)

Soit A une matrice de taille $m \times n$. Alors il existe deux matrices orthogonales U et V de tailles respectives $m \times m$ et $n \times n$, et une matrice diagonale Σ de taille $m \times n$ tel que :

$$A = U \Sigma V^\top$$

Démonstration Dans l'idée voici les étapes de la démonstration :
On sait que $A^\mathsf{T}A$ est symétrique semi définie positive. On peut donc l'orthogonalisé et ses valeurs propres sont positives. On écrit : $A^\mathsf{T}A = VS^2V^\mathsf{T}$. Où S est diagonale et ses valeurs propres nulles sont placées à la fin.
Pour tous les i tel que $s_{i,i} \neq 0$ on pose $U_i = \frac{1}{s_{i,i}} V_i S$. Calculons le produit de deux colonnes U_i et U_j :

$$U_i^\mathsf{T} U_j = \frac{1}{s_{i,i}s_{j,j}} V_i^\mathsf{T} A^\mathsf{T} A V_j = \frac{1}{s_{i,i}s_{j,j}} S_{i,j}^2 = \delta_{i,j}$$

Ainsi en complétant U par des vecteurs orthonormaux aux précédent on obtient une matrice U orthogonale. On supprime ou ajoute des lignes à S pour former une matrice Σ aux bonne dimensions.
On peut alors vérifier que $A = U\Sigma V^\mathsf{T}$. Mais je ne le ferai pas ici, car il faut un peu de temps et de rigueur. ■

Implémentation La preuve précédente décrit un algorithme pour obtenir cette décomposition mais le passage à la matrice $A^\mathsf{T}A$ cause des instabilités numériques car on élève au carré les valeurs de la matrice. Il existe des solutions pour palier à ces problèmes numériques. Mais ces solutions sont d'une grande complexité. En pratique on utilisera des librairies comme des boîtes noirs.

Méthode des 8 points Comme F est définie à un facteur multiplicatif près, 8 points suffisent à retrouver la matrice. Cela donne un problème linéaire simple. On peut bien évidement prendre plus de 8 points (et c'est ce qui est fait en pratique) et ensuite faire de l'optimisation. Pour chaque paire de point on a $\mathbf{x}_i^\mathsf{T} F \mathbf{x}_i' = 0$. On transforme alors F en un vecteur f pour obtenir un système linéaire :

$$\mathbf{x}_i^\mathsf{T} F \mathbf{x}_i' = 0 \Leftrightarrow A_i^\mathsf{T} f = 0 \quad \text{Avec } f = \begin{pmatrix} f_{11} & f_{12} & \cdots & f_{33} \end{pmatrix}^\mathsf{T}$$

$$A_i^\mathsf{T} = \begin{pmatrix} x_i x_i' & x_i y_i' & x_i & y_i x_i' & y_i y_i' & y_i & x_i' & y_i' & 1 \end{pmatrix}$$

On impose finalement la contrainte $\|f\| = 1$ en dernière contrainte. Voici alors la tête de notre problème d'optimisation pour n points :

$$\min_f \|Af\|^2 \quad \text{t.q. } \|f\| = 1 \quad \text{Avec } A = \begin{pmatrix} A_1 & \cdots & A_n \end{pmatrix}^\mathsf{T}$$

La solution de ce problème est obtenue pour f un vecteur propre de $A^\mathsf{T}A$ associée à la plus petite valeur propre (c'est à dire $f = V_9$ dans la SVD de A). Finalement on obtient généralement une matrice F inversible alors que le rang doit être 2. Pour palier à ça une solution consiste à calculer la SVD de F et mettre σ_3 à zéro puis de recomposer F.

La méthode des 7 points Forcer la contrainte $\det F = 0$ après minimisation n'est pas optimal. On peut alors imposer cette condition au cours de la minimisation ce qui conduit à la méthode des 7 points puisqu'une contrainte est ajoutée. Le système obtenue est alors un système linéaire $Af = 0$ avec A de taille 7×9. On décompose alors A en SVD afin d'obtenir deux vecteurs f_1 et f_2 qui forment la base du noyau de A. On recherche ensuite une solution de la forme $f = f_1 + \lambda f_2$ avec $\det F = 0$. Cela revient alors à chercher les zéros du polynôme suivant :

$$P(X) = \det(F_1 + XF_2)$$

C'est un polynôme de degré 3 qui admet alors un ou trois zéros.
Cette méthode nécessite une paire de points en moins ce qui diminue les chances d'avoir une mauvaise correspondance de points. Mais vient quand même la question de savoir comment peut-on être sûr de ne pas incorporer de mauvaises correspondances dans nos équations.

Normalisation L'échelle des coefficients de A peut poser pas mal de problèmes numériques. En effet si les x_i et y_i sont de l'ordre de 10^3 alors la matrice A contient des coefficients de l'ordre de 10^6 et de l'ordre de 1 (pour $A_{i,9}$). C'est pourquoi il est intéressant de normaliser les coordonnées avec une matrice de la forme :

$$N = \begin{pmatrix} 10^{-3} & & \\ & 10^{-3} & \\ & & 1 \end{pmatrix} \qquad \mathbf{x}_i \leftarrow N\mathbf{x}_i,\ \mathbf{x}_i \leftarrow N\mathbf{x}_i$$

Calcul de E E dépend de 5 paramètres. En effet il y en a 3 pour la rotation R, puis 3 pour la translation T et finalement une de moins car la matrice est définie à un facteur de multiplication près. Le calcul de cette matrice est bien plus complexe que F. Il peut être montrer que E est une matrice essentielle si et seulement si E possède une valeur singulière nulle et deux autre positives. Cela se traduit par :

$$2EE^{\mathsf{T}}E - \mathbf{tr}(EE^{\mathsf{T}})E = 0 \quad \text{et} \quad \det E = 0$$

La méthode des 5 points a été développé en 2004 par Nister. Le noyau de A est alors de dimension 4. Et les contraintes donnent 10 équations de degré 3 pour trouver quel est la bonne matrice dans le noyau de A. Bref c'est compliqué ...

III RANSAC

Comme évoqué précédemment on peut avoir des paires de points qui ne correspondent pas. Une estimation robuste de la matrice doit faire abstraction des valeurs aberrantes. Une idée consiste à tester plusieurs modèles construits à partir de différents sous-ensembles de paires de points. C'est algorithme est l'algorithme **RANSAC pour RANdom SAmple Consensus**.

Algorithm 1: RANSAC

Input: Un ensemble de n échantillons et une précision σ à atteindre

repeat

1. Sélectionner k échantillons parmi les n, où k est le nombre minimal pour construire un modèle.
2. Calculer un modèle avec ces échantillons et compter le nombre d'échantillons parmi les n qui sont correctement estimés à une précision σ
3. Si ce nombre est plus grand que pour tous les modèles précédemment rencontrés alors le garder

until *Un modèle satisfaisant à été trouvé*;

Dans notre cas la précision est donné par les distances $d(\mathbf{x}'_i, F^\top \mathbf{x}_i)$ des points de la seconde image à leur ligne épipôlaire prédite par le modèle grâce aux points correspondant dans la première image.

$$d(\mathbf{x}'_i, F^\top \mathbf{x}_i) = \frac{\left|\left(F^\top \mathbf{x}_i\right)_1 x'_i + \left(F^\top \mathbf{x}_i\right)_2 y'_i + \left(F^\top \mathbf{x}_i\right)_3\right|}{\sqrt{(F^\top \mathbf{x}_i)_1^2 + (F^\top \mathbf{x}_i)_2^2}}$$

Supposons que nous avons m paires de points corrects. Alors la probabilité de prendre k paires de points corrects est $(m/n)^k$. On veut que la probabilité de N_{iter} itérations de RANSAC donnent que des sous-ensembles contenant une mauvaise paire de points soit inférieur à $\beta = 1\%$:

$$\left(1 - (m/n)^k\right)^{N_{iter}} \leqslant \beta \qquad \Leftrightarrow \qquad N_{iter} \geqslant \frac{\log \beta}{\log\left(1 - (m/n)^k\right)}$$

m est inconnu mais on connaît une borne inférieur qui est le plus grand nombre de paires de points expliqué par un modèle obtenu jusque là. A chaque fois qu'on trouve un meilleur modèle on peut alors recalculer m.

IV Rectification épipôlaire

Il est commode d'avoir les lignes épipôlaires parallèles et à la même ordonné dans les deux images. La conséquence est que les épipôles sont les points à l'infini horizontalement.

$$e = e' = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

On peut toujours se retrouver dans cette situation en appliquant une rotation virtuelle à la caméra qui résulte en une homographie.
De plus deux point $\mathbf{x}$ et $\mathbf{x}'$ peuvent être des projection du même point X de l'espace sur la première et la seconde caméra si et seulement si ces deux point ont la même ordonnée. En effet la ligne épipôlaire d'un point est la ligne horizontale de même ordonnée. Cela donne :

$$\mathbf{x}^\top F \mathbf{x}' = 0 \Leftrightarrow y - y' = 0 \Leftrightarrow \begin{pmatrix} x & y & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ -y' \end{pmatrix} = 0 \Leftrightarrow \mathbf{x}^\top \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{pmatrix} \mathbf{x}' = 0$$

Finalement il existe une constante multiplicative λ tel que :

$$F = \lambda \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}_\times$$

Enfin les épipôles sont à l'infini si et seulement si les plans images des caméras sont confondus. Ainsi il n'y a pas de rotation entre les repères des deux caméras ($R = I_3$) :

$$P = K\begin{pmatrix} I_3 & 0 \end{pmatrix} \qquad P' = K'\begin{pmatrix} I_3 & T \end{pmatrix} \qquad \text{Avec } T = \begin{pmatrix} x & y & z \end{pmatrix}^\top$$

$$K = \begin{pmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \qquad K = \begin{pmatrix} f'_x & s' & c'_x \\ 0 & f'_y & c'_y \\ 0 & 0 & 1 \end{pmatrix}$$

Pour le calcul de $F = \left(K^{-1}\right)^{\mathsf{T}} [T]_\times K'^{-1}$ on a besoin des trois matrices suivante :

$$\left(K^{-1}\right)^{\mathsf{T}} = \frac{1}{f_x f_y} \begin{pmatrix} f_y & 0 & 0 \\ -s & f_x & 0 \\ sc_y - f_y c_x & -f_x c_y & f_x f_y \end{pmatrix} \qquad K'^{-1} = \frac{1}{f'_x f'_y} \begin{pmatrix} f'_y & -s' & s'c'_y - f'_y c'_x \\ 0 & f'_x & -f'_x c'_y \\ 0 & 0 & f'_x f'_y \end{pmatrix}$$

$$[T]_\times = \begin{pmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{pmatrix}$$

D'après la forme de F trouvée précédemment on sait que la première ligne de F est nulle. Comme les distances focales sont nulles on obtient alors que la première ligne de $[T]_\times K'^{-1}$ est nulle :

$$\begin{cases} -zf'_x & = & 0 \\ f'_x\left(zc'_y + yf'_y\right) & = & 0 \end{cases} \Rightarrow z = y = 0$$

Ainsi il existe B tel que $T = Be_1$. Cela simplifie grandement les calculs qui aboutissent à :

$$F = \frac{B}{f_y f'_y} \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -f'_y \\ 0 & f_y & c_y f'_y - c'_y f_y \end{pmatrix}$$

Ainsi on doit avoir :

$$f'_y = f_y \quad c'_y = c_y$$

On cherche alors des homographies H et H' tel que $F = H^{\mathsf{T}}[e_1]_\times H'$. Appliquer les inverses de ces homographies aux deux images permettrait de rectifier les lignes épipôlaires. Beaucoup de techniques pour faire ça, mais on ne les verra pas ici.

Chapitre C

Carte de disparités

Table des matières

I Triangulation

Commençons par reprendre les formules binoculaire pour observer un point de l'espace X avec des caméras de matrices de projections P et P'. Les coordonnées homogènes de l'images de X par les deux caméras sont notées x et x' :

$$x = PX \Leftrightarrow [x]_\times PX = 0 \qquad x' = P'X \Leftrightarrow [x']_\times P'X = 0$$

L'objectif de la triangulation est à partir des deux images x et x', de retrouver le point dans l'espace X. Cela est faisable en calculant une SVD pour trouver le noyau d'une matrice :

$$X \in \ker \begin{pmatrix} [x]_\times P \\ [x']_\times P' \end{pmatrix}$$

En se plaçant dans le repère de la seconde caméra et en notant R la matrice de rotation du repère de la seconde caméra au repère de la première caméra et T la translation entre les deux caméras. On peut réécrire les équations précédentes avec :

$$\lambda x = K(RX + T) \qquad \lambda' x' = K'X$$

On pose alors le vecteur $Y = \begin{pmatrix} X^\top & 1 & \lambda & \lambda' \end{pmatrix}^\top$ qui vérifie le système suivant :

$$\begin{pmatrix} KR & KT & -x & 0 \\ K' & 0 & 0 & -x' \end{pmatrix} Y = 0$$

On retrouve alors X à l'aide de la SVD de la matrice de droite qui nous donne le noyau.

Récupération de R et T Supposons que nous connaissons les matrices de calibration K et K' en plus de la matrice essentielle E (obtenable à partir de la matrice fondamentale F grâce à K et K'). On essaye alors de retrouver R et T. On sait que $E = [T]_\times R$.

$$E^\top E y = R^\top [T]_\times^\top [T]_\times R y = -R^\top [T]_\times ([T]_\times (Ry)) = -R^\top \Big((T^\top (Ry))T - (T^\top T)(Ry) \Big)$$

Ce qui donne :

$$E^\top E = R^\top \left(T T^\top - \|T\|_2^2 I_3 \right) R = -xx^\top + \|T\|_2^2 I_3 \qquad \text{Avec } x = R^\top T$$

On observe alors que $E^\top E x = 0$ et que pour y orthogonal à x, $E^\top E y = \|T\|_2^2 y$. Ainsi $E^\top E$ est une matrice ortho-diagonalisable de valeur propres $\|T\|_2^2$ et 0 et de rang 2. Ainsi les valeurs propres de E sont $\sigma_1 = \sigma_2 = \|T\|_2 = \sigma$ et $\sigma_3 = 0$. La SVD de E donne :

$$E = U \begin{pmatrix} \sigma & & \\ & \sigma & \\ & & 0 \end{pmatrix} V^\top = \sigma U \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} U^\top U \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} V^\top = [T]_\times R$$

On identifie alors :

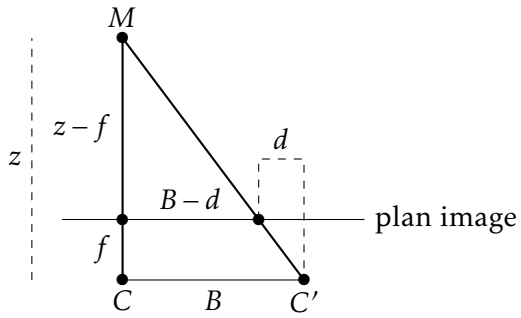
$$[T]_\times = \pm \sigma U \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} U^\top \qquad R = \pm U \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} V^\top$$

Ce qui donne 4 solutions possibles :

$$\begin{cases} T &= \pm \sigma U [e_3]_\times U^\top \\ R &= \pm U R_z \left(\pm \frac{\pi}{2} \right) V^\top \end{cases}$$

II Carte de disparités

On cherche à déterminer la profondeur du point observé en chaque pixel d'une image. Pour cela on se sert d'une autre image de la même scène prise à une position légèrement différente.



Avec un simple théorème de Thalès on a que la profondeur z d'un point M est inversement proportionnel à la disparité d de ce point, lorsque l'on bouge la caméra d'une distance B :

$$\frac{B-d}{B} = \frac{z-f}{z} \Leftrightarrow z = \frac{fB}{d}$$

A partir de quelques points caractéristiques de l'image on est capable de déterminer des bornes sur les valeurs que peut prendre d. Cela facilitera énormément le calcul de d pour tous les pixels de l'image car on pourra borné la zone de recherche pour trouver la nouvelle position de chaque pixel. Le principal défi se trouve au niveau du choix de l'invariant que l'on considère pour identifier la nouvelle position d'un pixel (exemple : la couleur, le gradient selon l'axe x, ...). Enfin on distingue deux méthodes de résolution du problème. Les méthodes globales et celles locales.

Méthode globale On cherche une image de disparité d^* solution du problème :

$$d^* = \arg\min_d \sum_p E_{data}(p, p + d(p); I_L, I_R) + \sum_{p \sim p} E_{reg}(d(p), d(p'); p, p', I_L, I_R)$$

Où $p \sim p'$ désigne l'ensembles des paires de pixels voisins. E_reg permet d'imposer une certaine régularité dans la carte de disparité. On peut par exemple prendre :

$$E_{reg} = \left| d(p) - d(p') \right|^2$$

En revanche à la frontière entre deux objets de profondeur différente, on ne souhaite pas une telle régularité. Les frontière sont généralement caractérisé par un changement de couleur entre deux pixels voisins. Voici donc une autre possibilité le coût de régularisation :

$$E_{reg} = \exp\left(-\frac{1}{\sigma^2} \| I_L(p) - I_L(p') \|^2 \right) \left| d(p) - d(p') \right|$$

Malheureusement ce problème est NP-difficile pour presque tous les termes de régularité (sauf $E_{reg} = \lambda_{p,p'} |d(p) - d(p')|$). Il faut donc utiliser des approximations.

Méthode locale Ces méthodes consiste à comparer des patchs autour des pixels. Voici par exemple des distances sur des patchs P :

$$\text{SAD (Sum of Absolute Differences) :} \quad D(p,q) = \sum_{r \in P} |I_L(p + r) - I_R(q + r)|$$

$$\text{SAD (Sum of Squared Differences) :} \quad D(p,q) = \sum_{r \in P} |I_L(p + r) - I_R(q + r)|^2$$

Augmenter la taille de P permet une plus grande régularité car la proportion de pixel en commun entre $p + P$ et $p' + P$ pour p et p' voisins, est plus grande. En revanche augmenter la taille du pixel risque d'effacer de petits objets.
Un problème qui peut intervenir est un changement d'intensité lumineuse entre l'image de gauche et celle de droite. Pour remédier à ce problème il est possible de soustraire la moyenne d'intensité sur le patch à l'intensité de chaque pixel. On peut même normalisé par la moyenne et la variance pour être encore plus robuste. Cela conduit aux de nouvelle carte de disparité suivante :

$$\text{CSSD (Centered Sum of Squared Distances) :}$$
$$d(p) = \arg\min_d \sum_{r \in P} \left(I_L(p + r) - \overline{I_{L|p+P}} - I_R(p + de_1 + r) + \overline{I_{R|p+de_1+P}} \right)^2$$

$$\text{NCC (Normalized Cross-Correlation) :}$$
$$d(p) = \arg\max_d \frac{\sum_{r \in P} \left(I_L(p + r) - \overline{I_{L|p+P}} \right) \left(I_R(p + de_1 + r) - \overline{I_{R|p+de_1+P}} \right)}{\sqrt{\sum \left(I_L(p + r) - \overline{I_{L|p+P}} \right)^2} \sqrt{\sum \left(I_R(p + de_1 + r) - \overline{I_{R|p+de_1+P}} \right)^2}}$$

Expansions de graines On peut obtenir des erreurs et ainsi du bruit dans notre carte de disparité. Pour remédier à cela une technique consiste à ne garder la disparité calculé que pour pour les pixels qui obtiennent une bonne corrélation ou une faible distance avec le patchs auquel ils sont associés. Tous ces pixels sont mis dans une file de priorité. Puis on déplie la file et pour chaque pixel p de la file on recalcule la disparité de ses voisin $p' \sim p$ qui n'ont encore jamais été dans la file. Cette disparité est calculé en retenant la meilleur parmi $d(p) - 1$, $d(p)$ et $d(p) + 1$. Cela permet d'obtenir une certaine régularité et évite le bruit. De plus comme généralement les pixels dont on n'est pas trop sûr de la disparité se trouve dans des zones de faible gradient on impose généralement pas cette régularité aux frontières entre les objets.

Supposition On a supposé dans les calculs précédent qu'il n'y avait pas de rotation de la caméra et que la caméra se déplaçait uniquement suivant l'axe des x. Ces suppositions ne sont pas problématique car, comme vu dans le chapitre précédent, il est possible de réaliser une rectification épipôlaire pour se placer dans les conditions voulues.

Chapitre D

Coupes de graphes

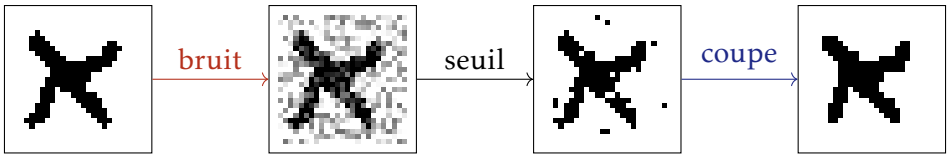
Table des matières

Rappel Le cours a commencé avec quelques rappels sur MAX-FLOW et MIN-CUT. Je ne ferai pas les rappels dans ces notes de cours. En revanche une slide contenait une généralisation du problème MIN-CUT appelé MIN-k-CUT, que je décrit ici. On se donne k terminaux $s_1, ..., s_k$. Le but est alors de partitionner le graphe en k partitions contant chacune exactement un terminal. L'objectif est d'avoir une coupe minimal. En notant $V_1, ..., V_k$ notre partition des sommets le problème s'écrit :

$$\min \sum_{i=1}^{k} \sum_{j=i+1}^{k} \sum_{p \in V_i} \sum_{q \in V_j} c(p,q)$$

Pour $k \geqslant 3$ ce problème est NP-difficile et même APX-difficile c'est où APX est la classe de complexité des problèmes approximables en temps polynomial. En revanche la restriction aux graphes planaires admet des solutions exactes.

I Restauration d'images noir et blanc



Pour chaque pixel $p \in \mathcal{P}$ on va assigner un label $l \in \{0,1\}$ pour dire si sa couleur est blanche ou noir. On note $D_p(l)$ la pénalité obtenu en assignant le label l à p. On associe le label 0 à la source s de notre graphe et le label 1 au puits t. Ainsi on défini les poids des arrêtes terminales par :

$$w(s,p) = D_p(1) \qquad w(p,t) = D_p(0)$$

La nouveau label d'un pixel sera alors 0 s'il est dans la composante de la source s et 1 s'il est dans la composante du puits t. On note I_p le label associé à p dans notre image de départ (celle obtenue après seuillage et avant la coupe sur le schéma ci-dessus). Une façon classique de définir les pénalité est la suivante :

$$D_p(l) = \begin{cases} 0 & \text{Si } I_p = l \\ 1 & \text{Si } I_p \neq l \end{cases}$$

Finalement on veut minimiser la longueur des contours. On pose alors une valeurs $\lambda > 0$ tel que :

$$\forall p, q \in \mathcal{P},\ w(p,q) = \lambda$$

Ainsi en cherchant la coupe minimal $C = \{\mathcal{S}, \mathcal{T}\}$ de notre graphe on cherche à minimiser :

$$\min |C| = \sum_{p \in \mathcal{P}} D_p(\mathbb{1}_{\mathcal{T}}(p)) + \lambda \sum_{p,q \in \mathcal{N}} \mathbb{1}_{\mathcal{S}\times\mathcal{T}}(p,q)$$

Où $\mathcal{N}$ est l'ensemble des couples de voisins. Si λ est très petit alors la coupe ne modifie pas les labels. En revanche si λ est trop grand on obtient soit une image toute blanche soit une image toute noire. Il faut donc trouver un bon équilibre.

De manière plus générale on note f_p le label de p :

$$f_p = \begin{cases} 1 & \text{Si } p \in \mathcal{T} \\ 0 & \text{Si } p \in \mathcal{S} \end{cases}$$

L'énergie de f que l'on veut minimiser s'exprime alors :

$$E(f) = E_{data}(f) + E_{regul}(f) = \sum_{p \in \mathcal{P}} D_p(f_p) + \sum_{p,q \in \mathcal{N}} V_{p,q}(f_p, f_q)$$

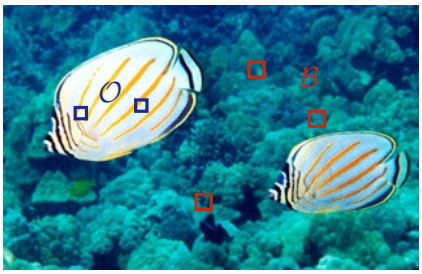
Proposition 5 (Kolmogorov & Zabih 2004)

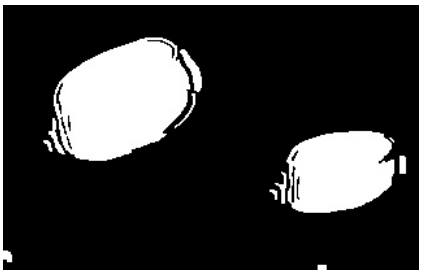
Si les potentiels V vérifient la condition de régularité suivante :

$$V_{p,q}(0,0) + V_{p,q}(1,1) \leqslant V_{p,q}(0,1) + V_{p,q}(1,0)$$

Alors il existe un graphe dont la coupe minimale définie une labellisation f qui atteint le minimum d'énergie.

II Segmentation d'images





Étant donné une image et des exemples de pixels d'un objet et du fond (*background*) $(\mathcal{O}, \mathcal{B})$ le but est de séparé tous les pixels de l'image dans ces deux classes. Il faut donc réfléchir à des critères qui vont nous permettre de faire cette séparation. On peut donc penser aux caractéristiques suivantes :

— Les pixels de l'objet et du fond ressemblent à ceux donnés en exemple dans le couple $(\mathcal{O}, \mathcal{B})$.
— Les contours de la segmentation se trouvent sur les zones de fort gradient. De plus afin d'éviter les fractales on veut que ces contours soient courts.

On va maintenant construire une énergie E à minimiser. On pose :

$$E(f) = D(f) + \lambda R(f)$$

Où D est le terme de données qui pénalise les pixels qui ne sont pas assignés à un label qui correspond à l'image de départ et $R(f)$ est le terme de régularisation qui pénalise les discontinuité dans les voisinages de pixels. Une façon de voire cette décomposition est avec la formulation bayésienne suivante. Minimiser E revient à maximiser la probabilité à posteriori $\mathbb{P}(f \mid I)$. Avec la formule de Bayes on obtient :

$$\underbrace{-\log \mathbb{P}(f \mid I) - \log \mathbb{P}(I)}_{E(f)+c^{te}} = \underbrace{-\log \mathbb{P}(I \mid f)}_{D(f)} \underbrace{-\log \mathbb{P}(f)}_{R(f)}$$

Terme des données / Vraisemblance En supposant les pixels indépendants, on a :

$$\mathbb{P}(I \mid f) = \prod_{p \in \mathcal{P}} \mathbb{P}(I_p \mid f_p)$$

D'où :

$$D(f) = \sum_{p \in \mathcal{P}} D_p(f_p) \qquad \text{Avec } D_p(f_p) = -\log \mathbb{P}(I_p \mid f_p)$$

Voici alors différente approche pour définir la vraisemblance :

— Utiliser les histogrammes de couleurs des pixels de l'objets et des pixels du fond afin d'en déduire des distributions empiriques $\mathbb{P}_{emp}$.
— Utiliser des mixtures de gaussiennes.
— Faire des comparaisons de patchs de texture.

Terme de régularisation / A priori On suppose que f est un champ Markovien par rapport au voisinage $\mathcal{N}$. C'est à dire :

$$\mathbb{P}(f_p = x \mid f_{\mathcal{P}\setminus\{p\}}) = \mathbb{P}(f_p = x \mid f_{\mathcal{N}_p})$$

On dit que $X = \left(X_p\right)_{p \in \mathcal{P}}$ est un champs de Gibbs par rapport au graphe $\mathcal{G}$ si :

$$\mathbb{P}(X = x) \propto \exp\left(- \sum_{C \text{ clique de } \mathcal{G}} V_C(x) \right)$$

Proposition 6 (Hammersley-Clifford 1971)

Si la distribution de probabilité de X est strictement positive. C'est à dire que pour tout x, $\mathbb{P}(X > 0) > 0$. Alors X est un champ Markovien par rapport à $\mathcal{N}$ si et seulement si X est un champ de Gibbs par rapport à $\mathcal{N}$

Dans notre cas f est une distribution strictement positive. Ainsi f est un champ Markovien et les seuls cliques de $\mathcal{N}$ sont ses arrêtes. D'où :

$$R(f) = -\log \mathbb{P}(f) = -\log \exp\left(-\sum_{p,q\in\mathcal{N}} V_{p,q}(f)\right) = \sum_{p,q\in\mathcal{N}} V_{p,q}(f_p, f_q)$$

De plus on suppose la symétrie des potentiels. Ainsi il existes des réels $B_{p,q}$ tel que :

$$V_{p,q}(f_p, f_q) = B_{p,q}\left(1 - \delta_{f_p,f_q}\right)$$

Où δ est ici le symbole de Kronecker. On prendra généralement l'expression suivante de la constante pour pénaliser la continuité de l'intensité :

$$B_{p,q} = \exp\left(-\frac{(I_p - I_q)^2}{2\sigma^2}\right) / \operatorname{dist}(p,q)$$

Cela revient à peu près à pénaliser les faibles gradients :

$$B_{p,q} = g\left(\left\|\nabla I_p\right\|\right)$$

Où g est une fonction strictement positive et décroissante. Par exemple $g(x) = 1/(1+cx^2)$.

Conclusion On cherche donc à minimiser l'énergie suivante (ou toute autre fonctions similaire) :

$$E(f) = \sum_{p\in\mathcal{P}} \mathbb{P}_{emp}(I_p \mid f_p) \; + \lambda \sum_{p,q\in\mathcal{N}} \left(1 - \delta_{f_p,f_q}\right) \exp\left(-\frac{(I_p - I_q)^2}{2\sigma^2}\right) / \operatorname{dist}(p,q)$$

Où :

$$\mathbb{P}_{emp}(c \mid l) = \begin{cases} \dfrac{\#\left\{p \in \mathcal{O} \mid I_p = c\right\}}{\#\mathcal{O}} & \text{Si } l = 1 \\ \dfrac{\#\left\{p \in \mathcal{B} \mid I_p = c\right\}}{\#\mathcal{B}} & \text{Si } l = 0 \end{cases}$$

On remarque alors qu'un pixel de $\mathcal{O}$ n'est pas nécessairement assigné au label 1 alors qu'il devrait l'être. On modifie alors légèrement D_p de la manière suivante :

$$D_p(l) = \begin{cases} (+\infty).(1-l) & \text{Si } p \in \mathcal{O} \\ (+\infty).l & \text{Si } p \in \mathcal{B} \\ \mathbb{P}_{emp}(I_p \mid l) & \text{Sinon} \end{cases}$$

III Coupes de graphes multi-labels

Cette fois-ci notre ensemble de labels n'est plus $\{0,1\}$ mais un ensemble $\mathcal{L}$ fini. L'énergie a toujours la même expression, à savoir :

$$E(f) = \sum_{p\in\mathcal{P}} D_p(f_p) \; + \sum_{p,q\in\mathcal{N}} V_{p,q}(f_p, f_q)$$

Proposition 7 (Ishikawa 2003)

Si les $V_{p,q}$ sont convexes et si $\mathcal{L}$ est linéairement ordonné (c'est à dire que $\mathcal{L}$ se trouve dans un espace à une seule dimension) alors une labellisation optimale f^* peut être obtenue en utilisant la coupe minimale d'un graphe.

La convexité impose le fait qu'il est plus rentable de faire des petits sauts de labels de proche en proche plutôt que des gros saut de labels :

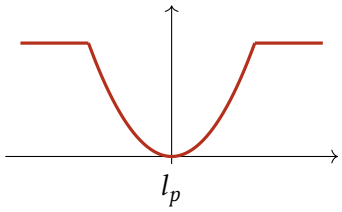
1	2	3	4	5	6	7	8

faible coût

1	1	1	1	8	8	8	8

fort coût

Les changement de labels sont alors continues et lisses. Il peut arriver que l'on ai besoin de fortes discontinuités dans les labels. Dans ce cas on est obligé de prendre $V_{p,q}$ non convexe. Le théorème de Ishikawa ne tient alors plus mais en général on connaît de bon algorithmes d'approximations de la solution optimale. Il faut savoir que dans la plupart des cas non convexes, le problème est NP-difficile. On peur par exemple prendre la fonction suivante :

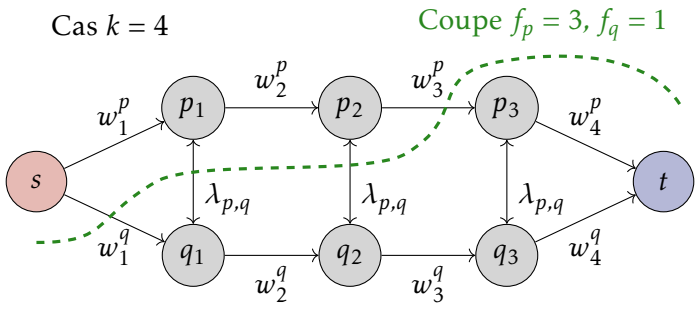


$$V_{p,q}(l_p, l_q) = \min\left(K, \|l_p - l_q\|^2\right)$$

Résolution On suppose $\mathcal{L} = \{1, ..., k\}$ et on suppose que le terme de régularisation est de la forme :

$$V_{p,q}(f_p, f_q) = \lambda_{p,q}|f_p - f_q|$$

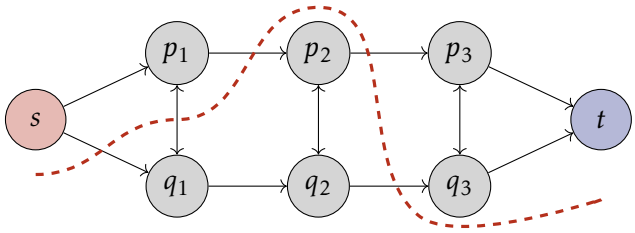
L'idée est de se ramener à une coupe binaire en créant une couche de nœuds par label :



Le label associé à la coupe est alors défini par la position de la coupe entre les nœuds $s, p_1, p_2, ..., p_{k-1}, t$. Reste à choisir les poids w_l^p. On peut dans un premier temps penser à :

$$w_l^p = D_p(l)$$

Hélas avec ce poids on peut obtenir une coupe minimale qui coupe plusieurs fois une même ligne :



Une solution consiste à ajouter une grande constante à tous les poids w_l^p afin que la coupe ne passe que par une seule de ces arrêtes pour chaque ligne. On pose alors :

$$w_l^p = D_p(l) + K_p \qquad \text{Avec } K_p = 1 + (k-1)\sum_{q \in \mathcal{N}_p} \lambda_{p,q}$$

IV Carte de disparités

On se donne deux images rectifiées I et I'. On cherche à trouver la disparité d_p de chaque pixel de I. On a donc $d_p \in \mathcal{L} = \{d_{min}, ..., d_{max}\}$. Ensuite on veut $D_p(d_p)$ petit lorsque I_p est proche de $I'_{p+d_p e_1}$ et aussi $V_{p,q}(d_p, d_q)$ petit lorsque d_p et d_q sont proches. Ainsi pour $D_p(d_p)$ on prend n'importe quel distance entre les patchs de l'images I à la position p et de l'image I' à la position $p + d_p e_1$ vu au chapitre précédent C.II. Pour la régularisation on prend de même l'expression utilisé dans la résolution de la section précédente. Cela donne par exemple :

$$E(f) = \sum_{p \in \mathcal{P}} D_{ZCSSD}\left(I_{P+p}, I'_{P+p+d_p e_1}\right) + \sum_{p,q \in \mathcal{N}} \lambda_{p,q} |d_p - d_q|$$

Où NCSSD signifie *Normalized Centered Sum of Square Differences* :

$$D_{ZCSSD}(I, I') = \sum_p \left(\left(I_p - \bar{I}\right)/\sigma_I - \left(I'_p - \bar{I'}\right)/\sigma_{I'}\right)^2$$

Approximation En pratique optimiser sur toute l'image nécessite beaucoup de nœuds. C'est pourquoi une approximation consiste à séparer l'image en sous-problème et optimiser de manière exacte localement tous ces nouveaux sous-problèmes.

Déplacement de labellisations Soit une labellisation $f : \mathcal{P} \to \mathcal{L}$ et des labels α et β.

— On dit que f' est un **déplacement standard** depuis f si f et f' diffère en au plus un pixel.
— On dit que f' est une α-**expansion** depuis f si pour tout pixel p la valeur de f'_p est soit f_p soit α.
— On dit que f' est une α-β-**permutation** depuis f si pour chaque pixel p alors soit $f_p = f'_p$ soit $(f_p, f'_p) \in \{\alpha, \beta\}^2$.

Une technique d'optimisation consiste à itérer ces déplacement pour converger vers un minimum local. Les deux derniers déplacement présenté ci-dessous peuvent être optimisés avec une coupe de graphe binaire ce que l'on sait très bien faire.

Occlusion On appelle pixel d'occlusion les pixels qui n'apparaissent que dans une seule des images. Pour trouver les pixels d'occlusion on peut calculer les cartes de disparité sur les deux images I puis I' et ensuite vérifier que si un pixel p de I est envoyé vers un pixel q de I' alors q est envoyé vers p. Si ce n'est pas le cas alors p est un pixel d'occlusion.

Chapitre E

Détection et description de zones d'intérêt

Table des matières

Motivations Pour calculer les panoramas et les matrices fondamentales, nous avons eu besoin de points d'intérêts à matcher entre des paires d'images. Pointer les points à la main prend beaucoup de temps et n'est pas non plus une méthode très précise. C'est pourquoi on aimerait bien avoir des techniques pour trouver automatiquement des points saillant dans une image. Cela va amener d'autres thématiques comme la description de ces points saillants pour ensuite chercher des points similaires dans d'autres images. Ou encore la vérification de la consistance géométrique pour des objets rigides ou déformables.

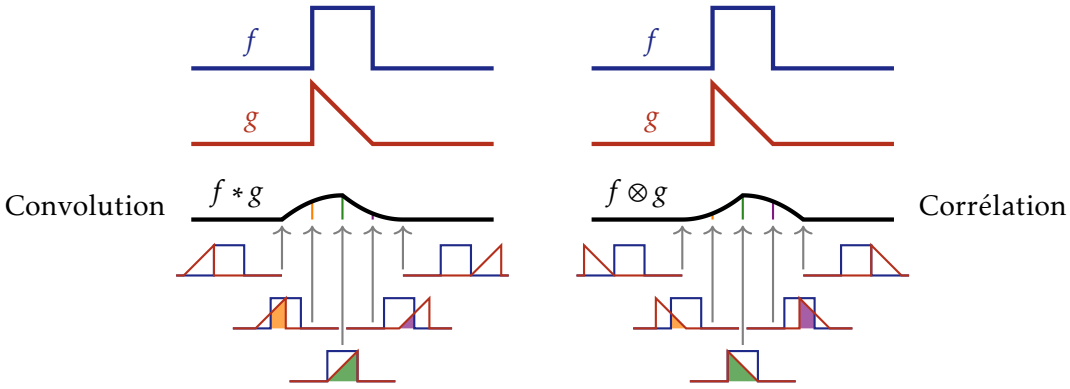
Une des applications est de pouvoir retrouver du contenu (comme un monument) dans une base de données d'images. Les difficultés résident dans les changement de points de vu, de luminosité, de réflexion sur des surfaces réfléchissantes, de paramètres de caméra, de saison ou encore d'occlusion ...

I Filtres d'images

Rappels Les slides contiennent des rappels sur les produits de convolutions et la FFT. En particulier le fait que tout opérateur linéaire et stationnaire peut être exprimé comme un produit de convolution. Puis petite revu des filtres classiques. Je vais juste donner ici ce qu'est la corrélation :

$$(f \otimes g)(x) = \int_{-\infty}^{+\infty} f(u)g(x+u)du = \int_{-\infty}^{+\infty} f(u-x)g(u)du$$

Il faut faire attention au fait que l'on perd la commutativité avec la corrélation bien que la linéarité, l'associativité et l'invariance par translation soit conservé.



Gradient Lorsqu'une image est bruité, le gradient est élevé à peu près partout, il est donc difficile de trouver les arrêtes. Pour remédier à cela, il suffit d'appliquer une filtre de lissage comme un filtre gaussien afin de faire disparaître le bruit pour se retrouver avec un signal lisse dont le gradient est élevé uniquement sur les arrêtes de l'image. On peut factoriser ces deux opération de lissage puis de gradient en une seule grâce à la formule :

$$\frac{d}{dx}(f * g) = f * \frac{d}{dx} g$$

Par exemple dans le cas d'un filtrage gaussien, on peut calculer explicitement la dérivée de la gaussienne pour n'avoir qu'une seule opération à faire.

II Détecteur de Harris *(Harris corner)*

Les coins d'une image sont les jonctions entre plusieurs arrêtes. C'est à dire les points au niveau desquels le gradient est fort dans toutes les directions. Les coins forment des points facilement repérables en plus d'être invariant aux translations, rotations, ... Le détecteur de Harris propose alors de trouver ces points là. On considère une image I en niveaux de gris. On s'intéresse dans un premier temps à la distance "L_2" muni d'une matrice de poids w entre un patch centré en $(0,0)$ et un autre centré en (x,y). On note S cette distance :

$$S_w(x,y) = \sum_u \sum_v w(u,v)\left(I(x+u,y+v) - I(u,v)\right)^2$$

Pour simplifier cette expression, on développe I au premier ordre :

$$I(u+x,v+y) \approx I(u,v) + I_x(u,v)x + I_y(u,v)y$$

Ce qui donne :

$$S_w(x,y) \approx \sum_u \sum_v w(u,v)\Big(I_x(u,v)x + I_y(u,v)y\Big)^2 = \begin{pmatrix} x & y \end{pmatrix} \sum_u \sum_v w(u,v) \begin{pmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{pmatrix} (u,v) \begin{pmatrix} x \\ y \end{pmatrix}$$

Le fait que le point en aux coordonnées $(0,0)$ soit un coin se traduit par le fait que les patchs voisins dans toutes les directions sont différents. C'est à dire que S_w soit grand dans toutes les directions. Or S_w est décrite par la matrice suivante :

$$M_w = \sum_u \sum_v w(u,v) \begin{pmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{pmatrix} (u,v)$$

Cette matrice étant symétrique, on peut l'orthodiagonalisé pour obtenir les valeurs propres λ_1 et λ_2. Finalement notre nouvelle caractérisation pour que $(0,0)$ soit un coin est que ces deux valeurs propres soient grandes. Pour quantifier le fait qu'elles soient toutes deux grandes sans que l'une soit vraiment plus grande que l'autre, Harris et Stephens ont introduit la valeur :

$$R_w = \lambda_1\lambda_2 - \kappa(\lambda_1+\lambda_2)^2 = \det(M_w) - \kappa\,\mathrm{trace}(M_w)^2$$

Où $\kappa < 1/4$ et généralement on choisi $\kappa \in [0.04, 0.06]$. Ici on a fait le travail pour le point $(0, 0)$ mais de manière générale pour le point (i, j) il suffit de prendre :

$$M_w(i,j) = \left[w * \begin{pmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{pmatrix} \right](i,j)$$

On prend souvent une gaussienne pour w. Enfin une fois qu'on a le score $R_w(i,j)$ pour tous les pixels (i,j), on ne garde plus que ceux qui ont un score supérieur à un certain seuil et qui sont des maximas locaux dans leur voisinage de 8 pixels.

1/ Maximalité

En réalité, garder les maxima locaux sur des voisinages de 9 pixels n'est pas très robuste et donne des distributions de points assez inégales.

NMS Une solution consiste à vérifier la maximalité dans un plus grand voisinage. Par exemple, dans une boule de rayon r. Pour ne pas supprimer trop de points lorsque r est grand on peut rajouter une constante c proche de 1 et inférieur à 1 (par exemple $c = 0.9$) et ne garder que les points x_p qui vérifient :

$$\forall q,\ \|x_p - x_q\| \leqslant r \Rightarrow cR_w(x_q) \leqslant R_w(x_p)$$

On a toujours le problème de la distribution inégale, mais on obtient un algo plus robuste bien qu'il faille choisir une valeur r.

ANMS Dans cette solution on calcule un rayon r pour tous les points et on trie ensuite les points par leur rayon.

Algorithm 2: ANMS

Input: Réponse de Harris R_w sur un ensemble $DetectedPoints$
$ProcessedPoints \leftarrow \emptyset$
foreach *point* $p \in DetectedPoints$ *par réponse* R_w *décroissante* **do**
  $r_p \leftarrow \inf_{\substack{q \in ProcessedPoints \\ R_w(x_p) < cR_w(x_q)}} \|x_p - x_q\|$
  Ajouter p à $ProcessedPoints$
Retourner les n points avec le plus grand rayon r

2/ Comparaison

Les matrices $M_w(x_p)$ définisse des régions elliptiques autour des points par :

$$\left\{x_p + x \mid x^\mathsf{T} M_w(x_p) x \leqslant 2\right\} = x_p + \mu_{M_w(x_p)}$$

Pour normaliser ces régions d'intérêts afin de pouvoir faire des comparaisons entre des images qui ont différent point de vues, on peut alors appliquer la transformation :

$$x' = M_w^{1/2} x$$

Sinon si on sait qu'entre nos deux images il y a une homographie H (i.e. $I = H(I')$), alors on peut faire une approximation affine de H que l'on note $\hat{H}$. Et la distance entre un point d'intérêt de matrice M dans la première image et un point d'intérêt de matrice M' dans la seconde peut être la distance de Jaccard suivante :

$$1 - \frac{\left|\mu_M \cap \left(\hat{H}^\mathsf{T} \mu_{M'} \hat{H}\right)\right|}{\left|\mu_M \cup \left(\hat{H}^\mathsf{T} \mu_{M'} \hat{H}\right)\right|}$$