PNP_Backend - Bases théoriques

Introduction

Le problème "Perspective-n-Points" (PnP) consiste, à partir d'une photo, à estimer les caractéristiques et la position de l'appareil de prise de vue.

Cette estimation se fait à partir de n *points* dont on connaît à la fois les coordonnées réelles (en 3 dimensions) et les coordonnées dans la photo (en 2 dimensions). L'estimation repose sur une modélisation du comportement des appareils photo appelée "pinhole-camera" (on emploiera plutôt le terme "caméra" dans ce document). Cette modélisation repose sur la *projection en perspective*, d'où le nom du problème.

Dans ce document, on commence par présenter le modèle "pinhole-camera". Puis on présentera le problème PnP, basé sur cette modélisation.

Modèle pinhole-camera

Dans ce modèle, on identifie l'ouverture de la caméra à un point unique ("pinhole") et on étudie comment se forme l'image dans le plan focal de la caméra grâce aux lois de l'optique géométrique. Il est à noter que l'image est "inversée" par rapport à l'objet réel.

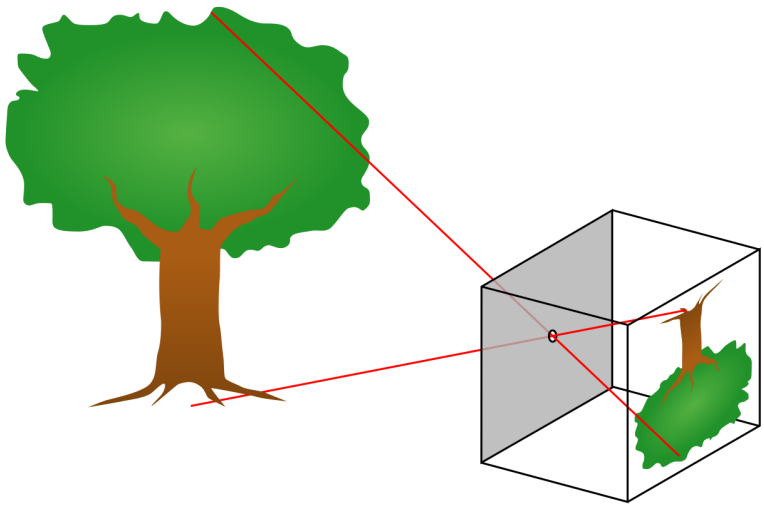


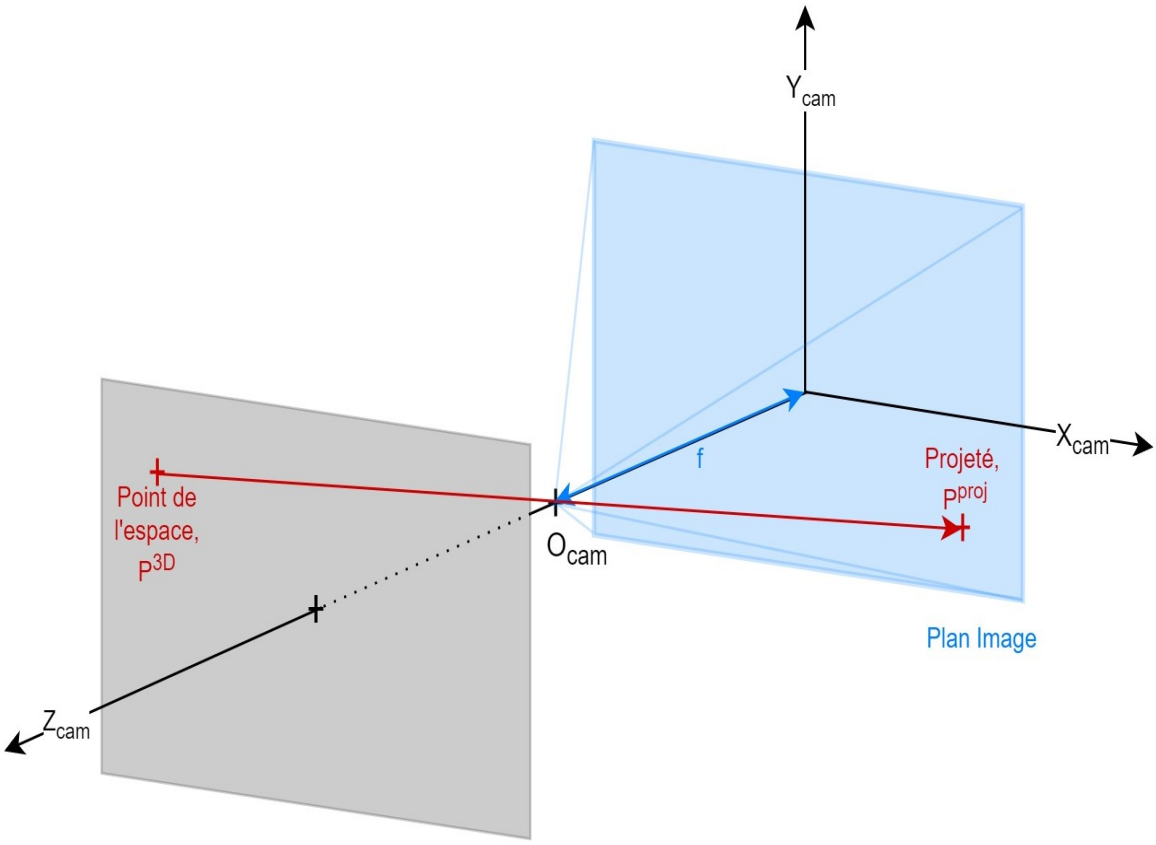
Schéma d'une caméra "pinhole"

Dans la suite, ce qui nous intéresse est la manière de passer des coordonnées 3D aux coordonnées 2D, à partir des caractéristiques de l'appareil. Ces équations sont à la base de la résolution du problème PnP.

Notations

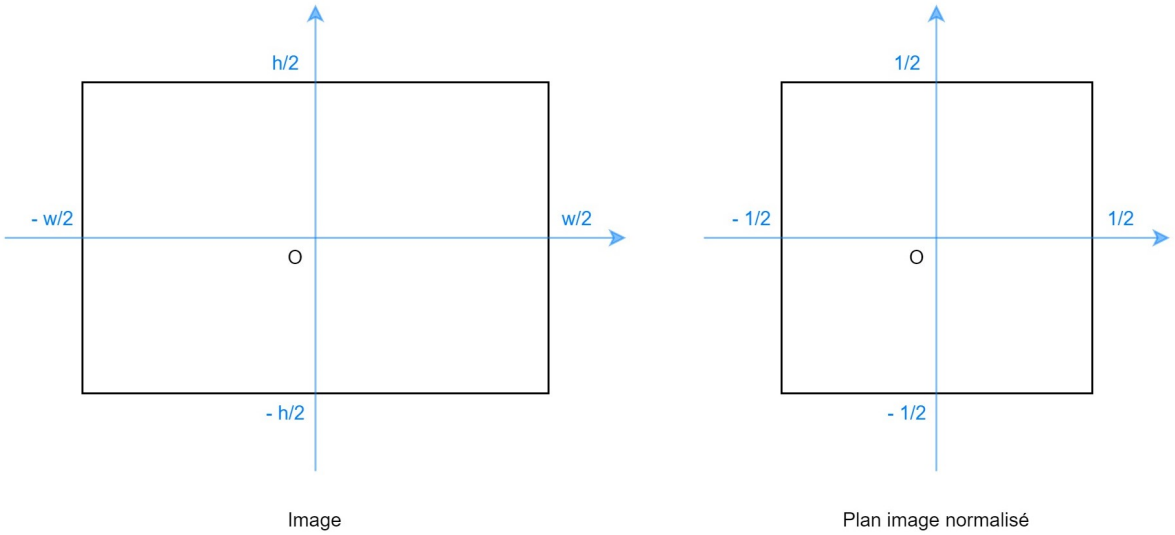
Dans ce qui suit, on utilise les notations suivantes:

- $R \in M_{3,3}$ représente la rotation de la caméra dans le repère-monde.
- $t \in \mathbb{R}_3$ représente la position de la caméra dans le repère monde.
- f représente la distance focale de la caméra.
- h et w représentent les dimensions de la photo en pixels (h : hauteur ; w : largeur).
- $P_w^{3D} = \begin{pmatrix} x_w^{3D} \\ y_w^{3D} \\ z_w^{3D} \end{pmatrix}$ désigne un point de l'espace, dans le repère monde (w=world)
- $P_c^{3D} = \begin{pmatrix} x_c^{3D} \\ y_c^{3D} \\ z_c^{3D} \end{pmatrix}$ désigne le même point, dans le repère de la caméra.
- $P_c^{proj} = \begin{pmatrix} x_c^{proj} \\ y_c^{proj} \\ z_c^{proj} \end{pmatrix}$ désigne la projection de P^{3D} dans le plan de la caméra.
- $P^{2D} = \begin{pmatrix} x^{2D} \\ y^{2D} \end{pmatrix}$ désigne les coordonnées du points correspondant sur l'image, en pixels.



Projection d'un point dans le plan image de la caméra

On pose également que l'image correspond au domaine $y_c \in [-\frac{1}{2}, \frac{1}{2}], x_c \in [-\frac{1}{2}, \frac{1}{2}]$ du plan de la caméra, $z_c = -f$, comme sur le schéma ci-dessous :



Plan image normalisé

Passage 3D -> 2D : projection perspective

Dans cette section, on suppose qu'il n'y a pas de distorsion. On cherche à exprimer les coordonnées P^{2D} d'un point sur l'image en fonction de ses coordonnées dans l'espace, P_w^{3D}.

On commence par exprimer les coordonnées du point dans le repère de la caméra, P_c^{3D}, en fonction de P_w^{3D}.

$$P_c^{3D} = R^{-1} \times (P_w^{3D} - t)$$

Maintenant que l'on connaît les coordonnées du point 3D dans le repère caméra, on exprime le projeté P_c^{proj} en fonction de P_c^{3D}.

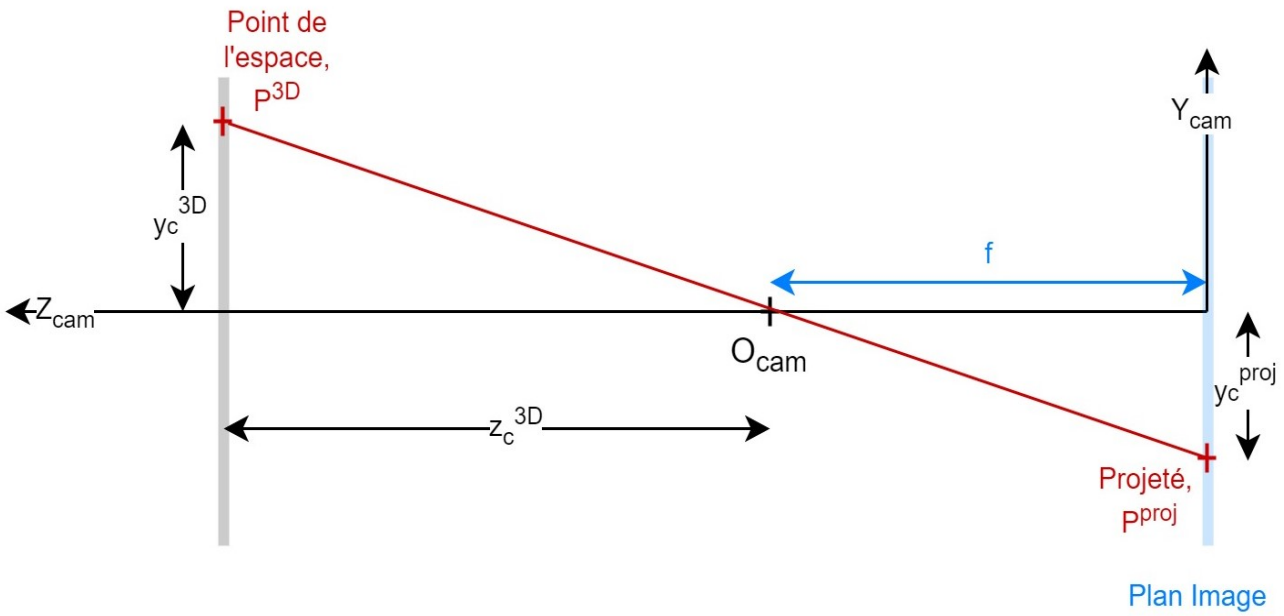
Par définition du plan image de la caméra :

$$z_c^{proj} = -f$$

Pour x_c^{proj} et y_c^{proj}, on applique le théorème de Thalès :

$$x_c^{proj} = -f \frac{x_c^{3D}}{z_c^{3D}}$$

$$y_c^{proj} = -f \frac{y_c^{3D}}{z_c^{3D}}$$



Projection d'un point dans le plan image - vue en coupe

On normalise les coordonnées de P^{proj} dans le plan image $y_c \in [-\frac{1}{2}, \frac{1}{2}], x_c \in [-\frac{1}{2}, \frac{1}{2}]$; ce qui nous donne :

$$P_c^{proj} = \begin{pmatrix} -f \times \frac{x_c^{3D}}{z_c^{3D}} \\ \\ -\frac{fw}{h} \times \frac{y_c^{3D}}{z_c^{3D}} \end{pmatrix}$$

Pour finir, on en déduit les coordonnées-image, P^{2D} :

$$P^{2D} = \begin{pmatrix} -x_c^{proj} \times w \\ \\ -y_c^{proj} \times h \end{pmatrix} = \begin{pmatrix} f \times w \times \frac{x_c^{3D}}{z_c^{3D}} \\ \\ \frac{fw}{h} \times h \times \frac{y_c^{3D}}{z_c^{3D}} \end{pmatrix}$$

(le $-$ provient du fait que l'image est inversée par rapport au projeté).

Prise en compte de la distorsion

Nous avons adopté un modèle simplifié dans lequel la distorsion est dûe uniquement à un décalage du centre optique de la caméra. Autrement dit, le centre optique de la caméra a pour coordonnées $(c_x, c_y) \neq (0, 0)$, ce qui a pour conséquence de décaler les points de la photo :

$$P_c^{proj} = \begin{pmatrix} -f \times \frac{x_c^{3D}}{z_c^{3D}} - c_x \\ \\ -\frac{fw}{h} \times \frac{y_c^{3D}}{z_c^{3D}} - c_y \end{pmatrix}$$

puis :

$$P^{2D} = \begin{pmatrix} fw \times \frac{x_c^{3D}}{z_c^{3D}} + c_x w \\ \\ \frac{fw}{h} \times h \times \frac{y_c^{3D}}{z_c^{3D}} + c_y h \end{pmatrix}$$

Généralisation : Equation de projection

En généralisant le raisonnement précédent, on peut résumer le lien entre un point P_w^{3D} de l'espace et le point projeté P_c^{proj} :

$$P_c^{proj} = K \times [I_3|0] \times [R|t] \times P_w^{3D}$$

où $K = \begin{pmatrix} f & 0 & c_x \\ \\ 0 & \frac{fw}{h} & c_y \\ \\ 0 & 0 & 1 \end{pmatrix}$ est appelée *matrice intrinsèque* de la caméra.

Problème PnP

Position du problème

Le problème PnP consiste à estimer les paramètres intrinsèques (matrice K) et extrinsèques (position t et rotation R) de la caméra, à partir de n correspondances ($P_i^{3D} \leftrightarrow P_i^{proj}$), telles que :

$$\forall i, P_i^{proj} = K \times [I_3|0] \times [R|t] \times P_i^{3D}$$

Paramètres à évaluer

Dans notre cas, on cherche à déterminer **9 paramètres** :

- la focale f => 1 paramètre
- le décentrage (c_x, c_y) => 2 paramètres
- l'orientation de la caméra dans l'espace => 3 paramètres
- la position de la caméra dans l'espace => 3 paramètres
  (les dimensions w et h de la photo sont connues)

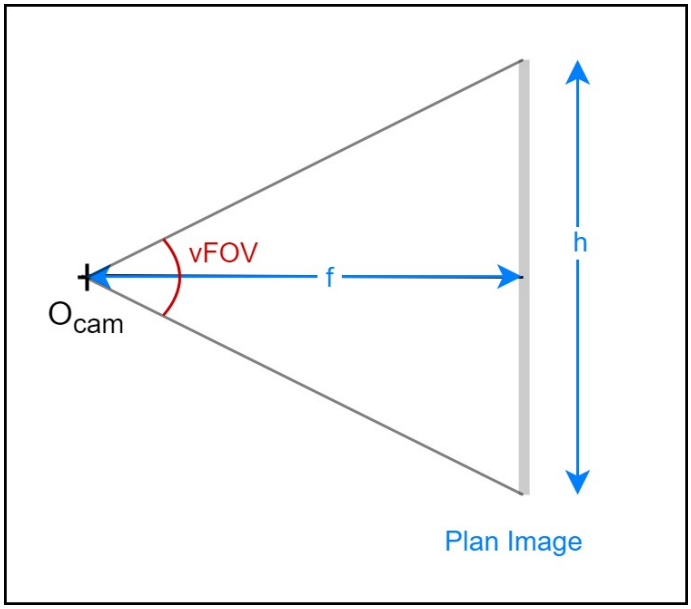
Remarque 1 : Dans la littérature, on parle le plus souvent de 11 paramètres. En effet, dans le cas général, la matrice intrinsèque de la caméra est de la forme :

$K = \begin{pmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}$ au lieu de $K = \begin{pmatrix} f & 0 & c_x \\ 0 & \frac{fw}{h} & c_y \\ 0 & 0 & 1 \end{pmatrix}$ dans notre cas.

Nous avons choisi de négliger les déformations des axes car elles sont difficiles à simuler. Ces déformations sont modélisées par le paramètre s ("skew", ou "inclinaison") et par les deux "focales" f_x et f_y (tout se passe comme si l'axe x était vu par un appareil de focale f_x tandis que l'axe y était vu par un appareil de focale f_y. Il s'agit d'une vue de l'esprit destinée essentiellement à gérer les photos rectangulaires, un appareil n'ayant bien sûr qu'une seule distance focale).

Remarque 2 : Pour des raisons de facilité d'implémentation, notre solveur calcule la longueur focale f mais retourne l'angle d'ouverture verticale $vFOV$ au lieu de la longueur focale f. Ces deux paramètres sont liés par la relation suivante :

$$tan(\frac{vFOV}{2}) = \frac{h}{2f} \Leftrightarrow vFOV = 2arctan(\frac{h}{2f})$$



Ouverture vertical, vFOV

Nombre de points nécessaires

La donnée d'une correspondance ($P_w^{3D} \leftrightarrow P^{2D}$) diminue de 2 les degrés de liberté du problème. Puisque nous avons 9 paramètres à évaluer, nous avons besoin d'un **minimum de** $\lceil \frac{9}{2} \rceil = 5$ **correspondances** (problème P5P).

Remarque : Dans la littérature, on peut lire qu'il ne faut que 3 points (problème P3P). Il s'agit du cas particulier où l'on connaît déjà les paramètres intrinsèques de la caméra. Dans ce cas en effet, on n'a que 6 paramètres à trouver (3 pour la position, 3 pour l'orientation), donc $\frac{6}{2} = 3$ point suffisent.
Dans notre cas en revanche, on veut pouvoir appliquer notre solveur à un dessin ou une photo d'archive, dont on ne peut pas connaître les paramètres intrinsèques (contrairement aux appareils récents qui encodent leurs paramètres intrinsèques dans les photos).

Critère d'optimisation

Comme on l'a vu, le problème PnP consiste à évaluer les matrices K, R et t à partir de n correspondances $P_i^{3D} \leftrightarrow P_i^{proj}$ telles que :

$$\forall i, P_i^{proj} = K \times [I_3|0] \times [R|t] \times P_i^{3D}$$

On cherche une solution approchée de ce problème, il s'agit donc de minimiser un certain critère d'erreur. Comme critère d'erreur, on choisit l'erreur de reprojection, que l'on explicite ici.

Supposons que l'on ait une estimation des matrices K (paramètres intrinsèques), R (orientation de la caméra dans l'espace) et t (position de la caméra dans l'espace). On note K^{*}, R^{*} et t^{*} ces estimations.

Pour estimer l'erreur, on re-projette les points 3D initiaux dans le plan de la caméra :
P^{proj*}_c = K^* \times [I_3 | 0] \times [R^{*}| t^{*}] \times P^{3D}_w

On obtient ainsi des points correspondant sur la photo :

$$P^{2D*} = \begin{pmatrix} P_c^{proj*}.x \times w \\ P^{proj*}.y \times h \end{pmatrix}$$

Puis, on compare avec les points 2D initiaux. On calcule ainsi une erreur par point :

$$e_i = \frac{1}{2} \times \left(\frac{|x^{2D*} - x^{2D}|}{w} + \frac{|y^{2D*} - y^{2D}|}{h} \right)$$

Enfin, on donne une estimation de l'erreur globale :

$$E = \frac{1}{n} \sum_{i=1}^{n} e_i$$

Implémentation du solveur

Notre solveur est basé sur les méthodes *calibrateCamera* et *solvePnP* de la librairie opencv (version 3.1).