Vision tridimensionelle

Calibrage 1

Vincent Chapdelaine-Couture
Nicolas Martin
Sébastien Roy

Laboratoire Vision3D
Département d'informatique et de recherche opérationnelle
Université de Montréal

Automne 2023

Au programme

Introduction

Modèle complet

Un rappel du modèle de caméra perspective:

$$\boldsymbol{M} = \underbrace{\begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix}}_{\text{internes}} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\text{projection}} \underbrace{\begin{bmatrix} r_{11} & r_{12} & r_{13} & 0 \\ r_{21} & r_{22} & r_{23} & 0 \\ r_{31} & r_{32} & r_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\text{externes}}$$

Calibrage et pose

Peut-on retrouver les paramètres de ce modèle à partir d'images?

Définition

Calibrage

Méthode permettant de retrouver les paramètres internes de la caméra :

- matrice $\boldsymbol{K}$
- coefficients de distortion radiale

$\Rightarrow$ Une caméra calibrée est une caméra dont on connaît les **paramètres internes**

Estimation de pose

Méthode permettant de déterminer les **paramètres externes** :

- orientation : $\boldsymbol{R}$
- position : $\boldsymbol{t}$

Rappel: modèle de projection perspective : $\boldsymbol{M} = \boldsymbol{K}\boldsymbol{R}[\mathtt{I}|\boldsymbol{t}]$

Définition

Calibrage

Méthode permettant de retrouver les paramètres internes de la caméra :

- matrice $\boldsymbol{K}$
- coefficients de distortion radiale

$\Rightarrow$ Une caméra calibrée est une caméra dont on connaît les **paramètres internes**

Estimation de pose

Méthode permettant de déterminer les **paramètres externes** :

- orientation : $\boldsymbol{R}$
- position : $\boldsymbol{t}$

Rappel: modèle de projection perspective : $\boldsymbol{M} = \boldsymbol{K}\boldsymbol{R}[\mathtt{I}|\boldsymbol{t}]$

De qu'elle information on dispose pour la calibration?

- Avec images d'un objet 3D connu
- Avec images d'un objets planaires (2D) connu
- Avec entités de l'image (points de fuite, angle entre les lignes)
- À partir de mouvements connus de la caméra

Grandes étapes

- Identifier les éléments (points, lignes, etc) dans les images
- Minimiser une fonction des paramètres et des entités projetées

Calibrage

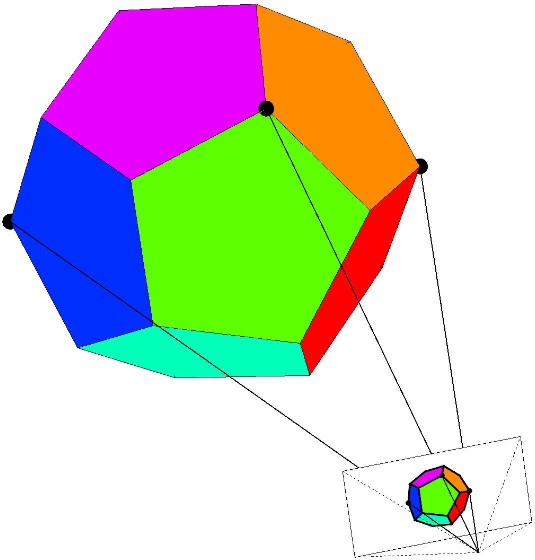
De qu'elle information on dispose pour la calibration?

- Avec images d'un objet 3D connu
- Avec images d'un objets planaires (2D) connu
- Avec entités de l'image (points de fuite, angle entre les lignes)
- À partir de mouvements connus de la caméra

Grandes étapes

- Identifier les éléments (points, lignes, etc) dans les images
- Minimiser une fonction des paramètres et des entités projetées

Calibrage avec objet 3D



- Points 3D connus :
  - les coins d'un dodécaèdre
  - deux plans avec des damiers
- Points images détectés
  - pas toujours facile...
  - un détection à la main est souvent la solution la plus simple
- Est-ce que les points 3D peuvent être quelconques ?

Direct Linear Transform

On cherche $\boldsymbol{M}$ avec une méthode linéaire

$\boldsymbol{p}^i$ est l'image du point $\boldsymbol{p}_w^i$ telle que :

$$\tilde{\boldsymbol{p}}^i \propto \boldsymbol{M}\tilde{\boldsymbol{p}}_w^i$$

- Donc : $\tilde{\boldsymbol{p}}^i \times (\boldsymbol{M}\tilde{\boldsymbol{p}}_w^i) = \boldsymbol{0}$
- Chaque correspondance donne trois équations :
  - mais deux linéairement indépendantes
  - donc deux contraintes sur $\boldsymbol{M}$
- $\boldsymbol{M}$ a 11 degrés de liberté : 12 entrées - 1 degré (facteur d'échelle)
- Il faut donc 5 1/2 correspondances pour retrouver $\boldsymbol{M}$

DLT = Transformation Linéaire Directe

- On l'utilise pour résoudre les systèmes homogènes linéaires, comme : trouver $\boldsymbol{x}$ tel que $\boldsymbol{A}\boldsymbol{x} = \mathbf{0}$ (soumis à la contrainte que $\boldsymbol{x} \neq \mathbf{0}$)
- C'est la méthode qu'on a utilisé pour trouver les homographies dans le tp1
- Il suffit de ramener le problème original en : $\min ||\boldsymbol{A}\boldsymbol{x}||$:
  - correspondances 2D-2D (homographie, modèle de caméra complet)
  - correspondances points-lignes (fondamentale : plus tard)
  - correspondances 3D-3D (alignement de nuage de points)
  - ...
- On résoud avec la SVD
- Cela minimise l'**erreur algébrique**

SVD = Décomposition en valeurs singulières

Pour toute matrice $\boldsymbol{A}$, $\boldsymbol{A}_{(n\times m)} = \boldsymbol{U}_{(n\times n)}\boldsymbol{\Sigma}_{(n\times m)}\boldsymbol{V}_{(m\times m)}^{\mathsf{T}}$:

- $\boldsymbol{U}$ et $\boldsymbol{V}$: matrices orthogonales ($\boldsymbol{U}\boldsymbol{U}^{\mathsf{T}} = \boldsymbol{V}\boldsymbol{V}^{\mathsf{T}} = \mathtt{I}$)
- $\boldsymbol{\Sigma} = \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_r)_{(n\times m)} = \begin{bmatrix} \begin{matrix} \sigma_1 & 0 & 0 & \ldots \\ 0 & \sigma_2 & 0 & \ldots \\ \vdots & \vdots & \ddots & \ldots \\ 0 & \ldots & \ldots & \sigma_r \end{matrix} & \mathbf{0} \\ \mathbf{0}^T & 0 \end{bmatrix}$ matrice diagonale

  avec les valeurs singulières ordonnées ($\sigma_1 > \sigma_2 > \ldots > \sigma_r > 0$)
- Décomposition qui n'est pas unique
- Reliée à la décomposition en valeurs/vecteurs propres de $\boldsymbol{A}^{\mathsf{T}}\boldsymbol{A}$

SVD : Applications

Rang d'une matrice

maximum entre

- #lignes linéairement indépendantes
- #colonnes linéairement indépendantes

$\Rightarrow$ nombre de valeurs singulières non nulles $= r$

Nullspace de $\boldsymbol{A}$

$\boldsymbol{x}$ tel que $\boldsymbol{A}\boldsymbol{x} = \mathbf{0}$
$\Rightarrow$ colonnes de $\boldsymbol{V}$ associées aux valeurs singulières nulles

Pseudo-inverse

inverse d'une matrice non-carrée
$\Rightarrow$ $\boldsymbol{A}^\dagger = \boldsymbol{V}\boldsymbol{\Sigma}^\dagger\boldsymbol{U}^\mathsf{T}$ avec $\boldsymbol{\Sigma}^\dagger = \mathrm{diag}({\sigma_1}^{-1}, {\sigma_2}^{-1}, \ldots, {\sigma_r}^{-1})_{(n \times m)}$

SVD et moindres carrées

Moindres carrées linéaires

Que faire quand on a plus de points que le minimum requis?

n équations de la forme $\sum_{j=1}^{m} v_{ij} x_j = y_i$ avec $n \geq m$

$\Rightarrow$ résoudre $\sum_{i=1}^{n} \boldsymbol{v}_i{}^{\mathsf{T}} \boldsymbol{x} = y_i$ avec $\boldsymbol{v}_i = (v_{i1}, v_{i2}, \ldots, v_{im})^{\mathsf{T}}$ et $\boldsymbol{x} = (x_1, x_2, \ldots, x_m)^{\mathsf{T}}$

$\Rightarrow$ résoudre $\boldsymbol{V}\boldsymbol{x} = \boldsymbol{y}$ avec $\boldsymbol{V} = \begin{bmatrix} \boldsymbol{v}_1 \\ \boldsymbol{v}_2 \\ \vdots \\ \boldsymbol{v}_n \end{bmatrix}$ et $\boldsymbol{y} = \begin{bmatrix} \boldsymbol{y}_1 \\ \boldsymbol{y}_2 \\ \vdots \\ \boldsymbol{y}_n \end{bmatrix}$

$\Rightarrow$ solution : $\boldsymbol{x} = \boldsymbol{V}^{\dagger} \boldsymbol{y}$ (en utilisant la SVD)

Cas particulier : résoudre $\boldsymbol{V}\boldsymbol{x} = \boldsymbol{0}$
$\Rightarrow$ méthode DLT du tp1

SVD et DLT

Résoudre $\boldsymbol{Ax} = \mathbf{0}$ est équivalent à $\min ||\boldsymbol{Ax}||$ sujet à $||\boldsymbol{x}|| = 1$

- $\boldsymbol{A} = \boldsymbol{U\Sigma V}^\top$
- on veut minimiser $||\boldsymbol{U\Sigma V}^\top \boldsymbol{x}||$
- donc $\min ||\boldsymbol{\Sigma V}^\top \boldsymbol{x}||$ ($\boldsymbol{U}$ est orthonormale)
- soit $\boldsymbol{y} = \boldsymbol{V}^\top \boldsymbol{x}$ alors on veut minimiser $||\boldsymbol{\Sigma y}||$ avec la contrainte $||\boldsymbol{x}|| = ||\boldsymbol{V}^\top \boldsymbol{x}|| = ||\boldsymbol{y}|| = 1$ ($\boldsymbol{V}$ est orthonormale)
- il suffit de choisir $\boldsymbol{y} = (0, 0, \ldots, 0, 1)^\top$ pour minimiser $||\boldsymbol{\Sigma y}||$ (car $\boldsymbol{\Sigma} = \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_r)$ avec $\sigma_1 > \sigma_2 > \ldots > \sigma_r$)
- donc $\boldsymbol{x} = \boldsymbol{V y}$ est la dernière colonne de $\boldsymbol{V}$

Ceci explique pourquoi on choisit toujours la dernière colonne de $\boldsymbol{V}$ pour résoudre $\boldsymbol{Ax} = \mathbf{0}$ avec les méthodes DLT

Résoudre $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{0}$ est équivalent à $\min ||\boldsymbol{A}\boldsymbol{x}||$ sujet à $||\boldsymbol{x}|| = 1$

- $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$
- on veut minimiser $||\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top\boldsymbol{x}||$
- donc $\min ||\boldsymbol{\Sigma}\boldsymbol{V}^\top\boldsymbol{x}||$ ($\boldsymbol{U}$ est orthonormale)
- soit $\boldsymbol{y} = \boldsymbol{V}^\top\boldsymbol{x}$ alors on veut minimiser $||\boldsymbol{\Sigma}\boldsymbol{y}||$| avec la contrainte $||\boldsymbol{x}|| = ||\boldsymbol{V}^\top\boldsymbol{x}|| = ||\boldsymbol{y}|| = 1$ ($\boldsymbol{V}$ est orthonormale)
- il suffit de choisir $\boldsymbol{y} = (0, 0, \ldots, 0, 1)^\top$ pour minimiser $||\boldsymbol{\Sigma}\boldsymbol{y}||$ (car $\Sigma = \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_r)$ avec $\sigma_1 > \sigma_2 > \ldots > \sigma_r$)
- donc $\boldsymbol{x} = \boldsymbol{V}\boldsymbol{y}$ est la dernière colonne de $\boldsymbol{V}$

Ceci explique pourquoi on choisit toujours la dernière colonne de $\boldsymbol{V}$ pour résoudre $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{0}$ avec les méthodes DLT

SVD et DLT

Résoudre $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{0}$ est équivalent à $\min ||\boldsymbol{A}\boldsymbol{x}||$ sujet à $||\boldsymbol{x}|| = 1$

- $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\mathsf{T}$
- on veut minimiser $||\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\mathsf{T}\boldsymbol{x}||$
- donc $\min ||\boldsymbol{\Sigma}\boldsymbol{V}^\mathsf{T}\boldsymbol{x}||$ ($\boldsymbol{U}$ est orthonormale)
- soit $\boldsymbol{y} = \boldsymbol{V}^\mathsf{T}\boldsymbol{x}$ alors on veut minimiser $||\boldsymbol{\Sigma}\boldsymbol{y}||$| avec la contrainte $||\boldsymbol{x}|| = ||\boldsymbol{V}^\mathsf{T}\boldsymbol{x}|| = ||\boldsymbol{y}|| = 1$ ($\boldsymbol{V}$ est orthonormale)
- il suffit de choisir $\boldsymbol{y} = (0, 0, \ldots, 0, 1)^\mathsf{T}$ pour minimiser $||\boldsymbol{\Sigma}\boldsymbol{y}||$ (car $\boldsymbol{\Sigma} = \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_r)$ avec $\sigma_1 > \sigma_2 > \ldots > \sigma_r$)
- donc $\boldsymbol{x} = \boldsymbol{V}\boldsymbol{y}$ est la dernière colonne de $\boldsymbol{V}$

Ceci explique pourquoi on choisit toujours la dernière colonne de $\boldsymbol{V}$ pour résoudre $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{0}$ avec les méthodes DLT

SVD et DLT

Résoudre $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{0}$ est équivalent à $\min ||\boldsymbol{A}\boldsymbol{x}||$ sujet à $||\boldsymbol{x}|| = 1$

- $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathsf{T}}$
- on veut minimiser $||\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathsf{T}}\boldsymbol{x}||$
- donc $\min ||\boldsymbol{\Sigma}\boldsymbol{V}^{\mathsf{T}}\boldsymbol{x}||$ ($\boldsymbol{U}$ est orthonormale)
- soit $\boldsymbol{y} = \boldsymbol{V}^{\mathsf{T}}\boldsymbol{x}$ alors on veut minimiser $||\boldsymbol{\Sigma}\boldsymbol{y}|||$ avec la contrainte $||\boldsymbol{x}|| = ||\boldsymbol{V}^{\mathsf{T}}\boldsymbol{x}|| = ||\boldsymbol{y}|| = 1$ ($\boldsymbol{V}$ est orthonormale)
- il suffit de choisir $\boldsymbol{y} = (0, 0, \ldots, 0, 1)^{\mathsf{T}}$ pour minimiser $||\boldsymbol{\Sigma}\boldsymbol{y}||$ (car $\boldsymbol{\Sigma} = \operatorname{diag}(\sigma_1, \sigma_2, \ldots, \sigma_r)$ avec $\sigma_1 > \sigma_2 > \ldots > \sigma_r$)
- donc $\boldsymbol{x} = \boldsymbol{V}\boldsymbol{y}$ est la dernière colonne de $\boldsymbol{V}$

Ceci explique pourquoi on choisit toujours la dernière colonne de $\boldsymbol{V}$ pour résoudre $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{0}$ avec les méthodes DLT

Résoudre $\boldsymbol{Ax} = \mathbf{0}$ est équivalent à $\min ||\boldsymbol{Ax}||$ sujet à $||\boldsymbol{x}|| = 1$

- $\boldsymbol{A} = \boldsymbol{U\Sigma V}^\mathsf{T}$
- on veut minimiser $||\boldsymbol{U\Sigma V}^\mathsf{T}\boldsymbol{x}||$
- donc $\min ||\boldsymbol{\Sigma V}^\mathsf{T}\boldsymbol{x}||$ ($\boldsymbol{U}$ est orthonormale)
- soit $\boldsymbol{y} = \boldsymbol{V}^\mathsf{T}\boldsymbol{x}$ alors on veut minimiser $||\boldsymbol{\Sigma y}|||$ avec la contrainte $||\boldsymbol{x}|| = ||\boldsymbol{V}^\mathsf{T}\boldsymbol{x}|| = ||\boldsymbol{y}|| = 1$ ($\boldsymbol{V}$ est orthonormale)
- il suffit de choisir $\boldsymbol{y} = (0, 0, \dots, 0, 1)^\mathsf{T}$ pour minimiser $||\boldsymbol{\Sigma y}||$ (car $\boldsymbol{\Sigma} = \mathrm{diag}(\sigma_1, \sigma_2, \dots, \sigma_r)$ avec $\sigma_1 > \sigma_2 > \dots > \sigma_r$)
- donc $\boldsymbol{x} = \boldsymbol{Vy}$ est la dernière colonne de $\boldsymbol{V}$

Ceci explique pourquoi on choisit toujours la dernière colonne de $\boldsymbol{V}$ pour résoudre $\boldsymbol{Ax} = \mathbf{0}$ avec les méthodes DLT

SVD et DLT

Résoudre $\boldsymbol{Ax} = \mathbf{0}$ est équivalent à $\min ||\boldsymbol{Ax}||$ sujet à $||\boldsymbol{x}|| = 1$

- $\boldsymbol{A} = \boldsymbol{U\Sigma V}^\mathsf{T}$
- on veut minimiser $||\boldsymbol{U\Sigma V}^\mathsf{T}\boldsymbol{x}||$
- donc $\min ||\boldsymbol{\Sigma V}^\mathsf{T}\boldsymbol{x}||$ ($\boldsymbol{U}$ est orthonormale)
- soit $\boldsymbol{y} = \boldsymbol{V}^\mathsf{T}\boldsymbol{x}$ alors on veut minimiser $||\boldsymbol{\Sigma y}|||$ avec la contrainte $||\boldsymbol{x}|| = ||\boldsymbol{V}^\mathsf{T}\boldsymbol{x}|| = ||\boldsymbol{y}|| = 1$ ($\boldsymbol{V}$ est orthonormale)
- il suffit de choisir $\boldsymbol{y} = (0, 0, \ldots, 0, 1)^\mathsf{T}$ pour minimiser $||\boldsymbol{\Sigma y}||$ (car $\boldsymbol{\Sigma} = \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_r)$ avec $\sigma_1 > \sigma_2 > \ldots > \sigma_r$)
- donc $\boldsymbol{x} = \boldsymbol{Vy}$ est la dernière colonne de $\boldsymbol{V}$

Ceci explique pourquoi on choisit toujours la dernière colonne de $\boldsymbol{V}$ pour résoudre $\boldsymbol{Ax} = \mathbf{0}$ avec les méthodes DLT

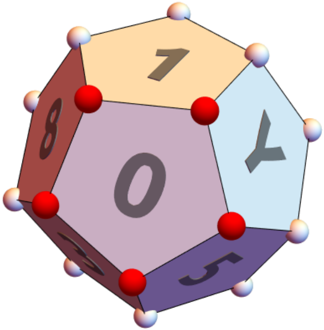
SVD et DLT

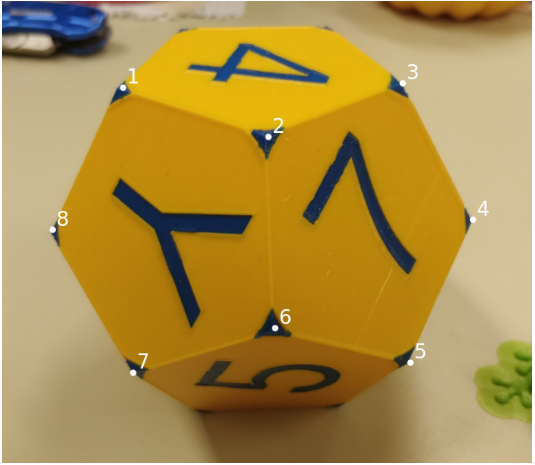
Résoudre $\boldsymbol{Ax} = \mathbf{0}$ est équivalent à $\min ||\boldsymbol{Ax}||$ sujet à $||\boldsymbol{x}|| = 1$

- $\boldsymbol{A} = \boldsymbol{U\Sigma V}^\mathsf{T}$
- on veut minimiser $||\boldsymbol{U\Sigma V}^\mathsf{T}\boldsymbol{x}||$
- donc $\min ||\boldsymbol{\Sigma V}^\mathsf{T}\boldsymbol{x}||$ ($\boldsymbol{U}$ est orthonormale)
- soit $\boldsymbol{y} = \boldsymbol{V}^\mathsf{T}\boldsymbol{x}$ alors on veut minimiser $||\boldsymbol{\Sigma y}|||$ avec la contrainte $||\boldsymbol{x}|| = ||\boldsymbol{V}^\mathsf{T}\boldsymbol{x}|| = ||\boldsymbol{y}|| = 1$ ($\boldsymbol{V}$ est orthonormale)
- il suffit de choisir $\boldsymbol{y} = (0, 0, \ldots, 0, 1)^\mathsf{T}$ pour minimiser $||\boldsymbol{\Sigma y}||$ (car $\boldsymbol{\Sigma} = \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_r)$ avec $\sigma_1 > \sigma_2 > \ldots > \sigma_r$)
- donc $\boldsymbol{x} = \boldsymbol{Vy}$ est la dernière colonne de $\boldsymbol{V}$

Ceci explique pourquoi on choisit toujours la dernière colonne de $\boldsymbol{V}$ pour résoudre $\boldsymbol{Ax} = \mathbf{0}$ avec les méthodes DLT

Avec un dodécahèdre :





Nous avons obtenu $\boldsymbol{M}$.
Comment décomposer $\boldsymbol{M} = \boldsymbol{K}\boldsymbol{R}[\mathtt{I}|\boldsymbol{t}]$ pour obtenir $\boldsymbol{K}$, $\boldsymbol{R}$, et $\boldsymbol{t}$?

Méthode indirecte:

$$\begin{aligned} \boldsymbol{M} &= \left[\boldsymbol{H} \quad | \boldsymbol{m}_4\right] \\ &= \boldsymbol{H}\left[\mathrm{I} \quad | \boldsymbol{H}^{-1}\boldsymbol{m}_4\right] \\ &= \boldsymbol{K}\boldsymbol{R}\left[\mathrm{I} \quad | \boldsymbol{t}\right] \end{aligned}$$

donc

- on retrouve la position $\boldsymbol{t}$ avec $\boldsymbol{H}^{-1}\boldsymbol{m}_4$
- on retrouve l'orientation $\boldsymbol{R}$ et les paramètres internes $\boldsymbol{K}$ avec $\boldsymbol{H} = \boldsymbol{K}\boldsymbol{R}$
  $\rightarrow$ comment faire ??

Décomposition QR

Pour une matrice carrée $\boldsymbol{A}_{(n \times n)}$:

$$\boldsymbol{A} = \boldsymbol{Q}_{(n \times n)} \boldsymbol{R}_{(n \times n)}$$

avec

- $\boldsymbol{Q}$ une matrice orthogonale : $\boldsymbol{Q}\boldsymbol{Q}^{\mathsf{T}} = \mathtt{I}$
- $\boldsymbol{R}$ une matrice triangulaire supérieure droite

Attention: Ici $\boldsymbol{R}$ est une matrice triangulaire supérieure, alors que $\boldsymbol{R}$ représente normalement une rotation ...

Décomposition QR

Pour une matrice carrée $\boldsymbol{A}_{(n \times n)}$:

$$\boldsymbol{A} = \boldsymbol{Q}_{(n \times n)} \boldsymbol{R}_{(n \times n)}$$

avec

- $\boldsymbol{Q}$ une matrice orthogonale : $\boldsymbol{Q}\boldsymbol{Q}^{\mathsf{T}} = \mathtt{I}$
- $\boldsymbol{R}$ une matrice triangulaire supérieure droite

Attention: Ici $\boldsymbol{R}$ est une matrice triangulaire supérieure, alors que $\boldsymbol{R}$ représente normalement une rotation ...

Décomposition RQ

Pour une matrice carrée $\boldsymbol{A}_{(n \times n)}$:

$$\boldsymbol{A} = \boldsymbol{R}_{(n \times n)} \boldsymbol{Q}_{(n \times n)}$$

avec

- $\boldsymbol{R}$ une matrice triangulaire supérieure droite
- $\boldsymbol{Q}$ une matrice orthogonale : $\boldsymbol{Q}\boldsymbol{Q}^{\mathsf{T}} = \mathtt{I}$
- En Mathematica, il n'y a pas de fonction directe pour trouver la décomposition RQ, par contre elle est facile à trouver si on sait trouver une décomposition QR...
- C'est ce qu'il nous faut pour décomposer $\boldsymbol{H} = \boldsymbol{K}\boldsymbol{R}$

Rappel : $\boldsymbol{K} = \begin{bmatrix} f & s & cx \\ 0 & \alpha f & cy \\ 0 & 0 & 1 \end{bmatrix}$

Méthode directe:

$$\boldsymbol{H} \propto \boldsymbol{K}\boldsymbol{R} = \rho \begin{bmatrix} f\boldsymbol{r}_1{}^\mathsf{T} + s\boldsymbol{r}_2{}^\mathsf{T} + cx\boldsymbol{r}_3{}^\mathsf{T} \\ \alpha f\boldsymbol{r}_2{}^\mathsf{T} + cy\boldsymbol{r}_3{}^\mathsf{T} \\ \boldsymbol{r}_3{}^\mathsf{T} \end{bmatrix}$$

donc:

- $\rho = \|\boldsymbol{h}_3\|$
- $\boldsymbol{r}_3 = \boldsymbol{h}_3/\rho$
- $cx = \boldsymbol{h}_1{}^\mathsf{T}\boldsymbol{h}_3/\rho^2$
- $cy = \boldsymbol{h}_2{}^\mathsf{T}\boldsymbol{h}_3/\rho^2$
- $\ldots$

Voir Forsyth-Ponce pour le reste

Solution optimale

Idéalement, on minimise l'erreur géométrique...

Minimisation de l'erreur de reprojection

$$\sum_i \mathrm{dist}^2(\boldsymbol{p}_i, \boldsymbol{M}\boldsymbol{p}_i)$$

Si on a des contraintes sur $\boldsymbol{K}$:

$$\sum_i \mathrm{dist}^2(\boldsymbol{p}_i, \boldsymbol{K}\boldsymbol{R}\left[\mathrm{I} \quad | \boldsymbol{t}\right] \boldsymbol{p}_i)$$

$\Rightarrow$ requiert une méthode non-linéaire . . .

Caméra affine

Et si on utilisait un modèle de caméra affine?

$$\boldsymbol{p} - \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \end{bmatrix} \tilde{\boldsymbol{p}}_w = \boldsymbol{p} - \begin{bmatrix} {\boldsymbol{m}_1}^\mathsf{T} \\ {\boldsymbol{m}_2}^\mathsf{T} \end{bmatrix} \tilde{\boldsymbol{p}}_w = \mathbf{0}$$

et le système est donc

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$$

avec $\boldsymbol{A} = \begin{bmatrix} {\tilde{\boldsymbol{p}}_w^1}^\mathsf{T} & \mathbf{0}^\mathsf{T} \\ \mathbf{0}^\mathsf{T} & {\tilde{\boldsymbol{p}}_w^1}^\mathsf{T} \\ \vdots & \\ {\tilde{\boldsymbol{p}}_w^n}^\mathsf{T} & \mathbf{0}^\mathsf{T} \\ \mathbf{0}^\mathsf{T} & {\tilde{\boldsymbol{p}}_w^n}^\mathsf{T} \end{bmatrix}$, $\boldsymbol{x} = \begin{pmatrix} \boldsymbol{m}_1 \\ \boldsymbol{m}_2 \end{pmatrix}$ et $\boldsymbol{b} = \begin{bmatrix} \boldsymbol{p}^1 \\ \vdots \\ \boldsymbol{p}^n \end{bmatrix}$

- On résout avec la pseudo-inverse ou SVD
- $\boldsymbol{M}$ n'est pas définie à un facteur d'échelle, donc exactement 8 degrés de libertés : seulement 4 correspondances suffisent
- Cela minimise l'erreur géométrique :
  $\|\boldsymbol{A}\boldsymbol{x}\| = \sum_i^n (\boldsymbol{p}_i - \boldsymbol{M}\tilde{\boldsymbol{p}}_w^i)^2 = \sum_i^n d_{\text{geom}}(\boldsymbol{p}_i, \boldsymbol{M}\tilde{\boldsymbol{p}}_w^i)$

Calibrage planaire

Idée

- Image d'un plan
- Géométrie euclidienne connue
- Paramètres internes calculés à partir de plusieurs vues
- Pose par rapport au plan pour chaque vue

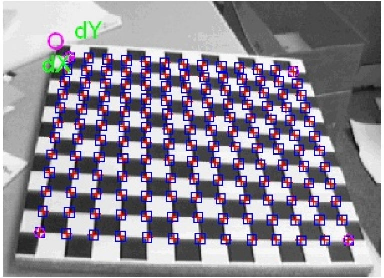
Plan euclidien

On connaît un système d'axe euclidien dans ce plan. On connaît par exemple:

- l'angle entre les lignes
- la distance entre les points
- le fait que l'échiquier est constitué de "carrés"

Avec une grille de calibrage:

- Trouver des points saillants
- Retrouver automatiquement la configuration des points (i.e. leur coordonnée)
  - Pas si difficile
  - En OpenCV : findChessboardCorners



Modèle de projection → transformation

- Pour chaque damier retrouvé, on peut fixer l'origine du monde sur un des coins
- Sans pertes de généralité, on fixe le plan du damier en $Z = 0$
- Projection d'un plan en $Z = 0$

$$\boldsymbol{K}\boldsymbol{R}[\mathrm{I}_{3\times 3}|\boldsymbol{t}]\begin{pmatrix} X \\ Y \\ 0 \\ 1 \end{pmatrix} = \boldsymbol{K}\boldsymbol{R}\begin{bmatrix} 1 & 0 & \\ 0 & 1 & \boldsymbol{t} \\ 0 & 0 & \end{bmatrix}\begin{pmatrix} X \\ Y \\ 1 \end{pmatrix} = \boldsymbol{H}\begin{pmatrix} X \\ Y \\ 1 \end{pmatrix}$$

- Donc chaque damier fournit une homographie qui contient de l'information sur $\boldsymbol{K}$

Quelle contraintes peut-on trouver dans

$$\boldsymbol{H} = \boldsymbol{K}\boldsymbol{R}\begin{bmatrix} 1 & 0 & \\ 0 & 1 & \boldsymbol{t} \\ 0 & 0 & \end{bmatrix}?$$

On peut reformuler avec les colonnes de $\boldsymbol{H}$ et $\boldsymbol{R}$:

$$\boldsymbol{H} = \begin{bmatrix} h_1 & h_2 & h_3 \end{bmatrix} = \boldsymbol{K}\begin{bmatrix} r_1 & r_2 & \boldsymbol{R}\boldsymbol{t} \end{bmatrix}$$

donc clairement

$$h_1 = \boldsymbol{K}r_1 \ \text{ et } \ h_2 = \boldsymbol{K}r_2$$

et inversement

$$r_1 = \boldsymbol{K}^{-1}h_1 \ \text{ et } \ r_2 = \boldsymbol{K}^{-1}h_2$$

On sait que $\boldsymbol{R}$ est orthogonale, donc $\boldsymbol{r}_1{}^{\mathsf{T}}\boldsymbol{r}_2 = 0$.

Puisque $r_1 = \boldsymbol{K}^{-1}h_1$ et $r_2 = \boldsymbol{K}^{-1}h_2$, on obtient la contrainte

$$\boldsymbol{h}_1\boldsymbol{K}^{\text{-}\mathsf{T}}\boldsymbol{K}^{\text{-}1}\boldsymbol{h}_2 = 0$$

On voit que la contrainte est sur $\boldsymbol{\omega} = \boldsymbol{K}^{\text{-}\mathsf{T}}\boldsymbol{K}^{\text{-}1}$, qu'on appelle l'image de la conique absolue.

Intuition algébrique, équation 2

On sait que $\boldsymbol{R}$ est orthonormale, donc $\|\boldsymbol{r}_1\| = \|\boldsymbol{r}_2\|$.

Puisque $r_1 = \boldsymbol{K}^{-1} h_1$ et $r_2 = \boldsymbol{K}^{-1} h_2$, on obtient la contrainte

$$\boldsymbol{h}_1 \boldsymbol{K}^{\text{-}\mathsf{T}} \boldsymbol{K}^{\text{-}1} \boldsymbol{h}_1 = \boldsymbol{h}_2 \boldsymbol{K}^{\text{-}\mathsf{T}} \boldsymbol{K}^{\text{-}1} \boldsymbol{h}_2$$

Chaque homographie ajoute donc deux contraintes sur $\boldsymbol{\omega} = \boldsymbol{K}^{\text{-}\mathsf{T}} \boldsymbol{K}^{\text{-}1}$. On va utiliser ces contraintes pour estimer ω, puis $\boldsymbol{K}$.

Essayons de trouver d'où vient $\boldsymbol{\omega}$, l'image de la conique absolue ...

Conique

Définition

Une conique est une courbe d'équation du second degré dans le plan

- Issu de l'intersection d'un cône avec un plan
- Équation pour un point $\boldsymbol{p} = (x, y)^\mathsf{T}$: $ax^2 + bxy + cy^2 + dx + ey + f = 0$
- Sous forme matricielle :

$$\tilde{\boldsymbol{p}}^\mathsf{T} \boldsymbol{C} \tilde{\boldsymbol{p}} = 0$$

avec $\boldsymbol{C} = \begin{bmatrix} a & b/2 & d/2 \\ b/2 & c & e/2 \\ d/2 & e/2 & f \end{bmatrix}$

- Défini à un facteur d'échelle prêt (donc 5 degrés de liberté)

Transformation projective d'une conique

- Soit une homographie $\boldsymbol{H}$ telle que $\tilde{\boldsymbol{p}}' \propto \boldsymbol{H}\tilde{\boldsymbol{p}}$
- Pour un point $\boldsymbol{p}$ sur la conique $\boldsymbol{C}$:

$$\begin{aligned}
\tilde{\boldsymbol{p}}^{\mathsf{T}}\boldsymbol{C}\tilde{\boldsymbol{p}} &= 0 \\
\tilde{\boldsymbol{p}}'^{\mathsf{T}}(\boldsymbol{H}^{-1})^{\mathsf{T}}\boldsymbol{C}\boldsymbol{H}^{-1}\tilde{\boldsymbol{p}}' &= 0 \\
\tilde{\boldsymbol{p}}'^{\mathsf{T}}\boldsymbol{H}^{-\mathsf{T}}\boldsymbol{C}\boldsymbol{H}^{-1}\tilde{\boldsymbol{p}}' &= 0
\end{aligned}$$

- Donc la conique $\boldsymbol{C}$ transformée par $\boldsymbol{H}$ devient $\boldsymbol{C}' = \boldsymbol{H}^{-\mathsf{T}}\boldsymbol{C}\boldsymbol{H}^{-1}$.

Conique absolue $\boldsymbol{\Omega}_\infty$

- Conique $\boldsymbol{\Omega}_\infty$ de points dans la plan à l'infini $\boldsymbol{\pi}_\infty = (0,0,0,1)^\top$
- Donc pour des points de la forme $(x,y,z,0)^\top$, donc à l'infini
- Les points qui satisfont cette conique sont purement imaginaires (i.e il n'y a aucun point réel qui appartient à cette conique)

Ω_∞ est la conique de matrice : $\boldsymbol{C} = \mathtt{I}$ pour des points de la forme $(x,y,z)^\mathsf{T}$

Projection de $\boldsymbol{\pi}_\infty$

- Projection des points de $\boldsymbol{\pi}_\infty$

$$\boldsymbol{p} = \boldsymbol{K}\boldsymbol{R}[\mathrm{I}|\boldsymbol{t}]\begin{pmatrix} x \\ y \\ z \\ 0 \end{pmatrix} = \boldsymbol{K}\boldsymbol{R}\begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

- seuls l'orientation de la caméra et ses paramètres internes influencent la projection de ces points

Analogie: La position du soleil ou des étoiles ne change pas si on se déplace, seulement si on change d'orientation.

Comme $\boldsymbol{\Omega}_\infty$ est dans $\boldsymbol{\pi}_\infty$ on peut travailler avec les plans projectifs

- Posons $\boldsymbol{H} = \boldsymbol{K}\boldsymbol{R}$
- Appliquons à $\boldsymbol{\Omega}_\infty = \mathrm{I}$

$$(\boldsymbol{K}\boldsymbol{R})^{-\top}\mathtt{I}(\boldsymbol{K}\boldsymbol{R})^{-1} = \boldsymbol{K}^{-\top}\boldsymbol{R}\mathtt{I}\boldsymbol{R}^{-1}\boldsymbol{K}^{-1} = \boldsymbol{K}^{-\top}\boldsymbol{K}^{-1}$$

Image de la Conique Absolue (*Image of the Absolute Conic (IAC)*)

$$\boldsymbol{\omega} = \boldsymbol{K}^{-\top}\boldsymbol{K}^{-1}$$

Pour $\boldsymbol{K} = \begin{bmatrix} f & 0 & u \\ 0 & f\alpha & v \\ 0 & 0 & 1 \end{bmatrix}$

Au long,

$$\boldsymbol{\omega} = \boldsymbol{K}^{-\top}\boldsymbol{K}^{-1} \propto \begin{bmatrix} \alpha^2 & 0 & -\alpha^2 u \\ 0 & 1 & -v \\ -\alpha^2 u & -v & f^2\alpha^2 + u^2\alpha^2 + v^2 \end{bmatrix}$$

Pour le moment, l'IAC est inconnue, nous allons l'estimer

$$\boldsymbol{\omega} = \begin{bmatrix} w_1 & 0 & w_4 \\ 0 & w_2 & w_5 \\ w_4 & w_5 & w_3 \end{bmatrix}$$

Revenons à notre transformation $\boldsymbol{H}$

Rappel :

$$\boldsymbol{h}_1^\top \omega \boldsymbol{h}_2 = 0, \text{ et } \boldsymbol{h}_1^\top \omega \boldsymbol{h}_1 - \boldsymbol{h}_2^\top \omega \boldsymbol{h}_2 = 0$$

où $\boldsymbol{h}_i$ est la $i^{\text{ème}}$ colonne de $\boldsymbol{H}$

Nous avons donc des contraintes linéaires:

$$\begin{bmatrix} g_1 g_2 & g_4 g_5 & g_7 g_8 & g_2 g_7 + g_1 g_8 & g_5 g_7 + g_4 g_8 \\ g_1^2 - g_2^2 & g_4^2 - g_5^2 & g_7^2 - g_8^2 & 2g_1 g_7 - 2g_2 g_8 & 2g_4 g_7 - 2g_5 g_8 \end{bmatrix} \begin{pmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \end{pmatrix} = \mathbf{0}$$

avec

$$\boldsymbol{H} = \begin{bmatrix} g_1 & g_2 & g_3 \\ g_4 & g_5 & g_6 \\ g_7 & g_8 & g_9 \end{bmatrix}$$

Nous pouvons alors retrouver l'IAC et ensuite calculer les paramètres internes.

Décomposition de ω

On peut maintenant retrouver les paramètres de $\boldsymbol{K}$.
Deux méthodes possibles:

- Calcul direct
- Décomposition de Cholesky

Exercice

- Combien faut-il d'homographie(s) pour calibrer complètement la caméra?
- Si le ratio d'image est 1 et qu'on connaît le point principal, combien en faut-il?

Décomposition de Cholesky

Pour une matrice symétrique définie positive $\boldsymbol{A}_{(n \times n)}$:

$$\boldsymbol{A} = \boldsymbol{L}_{(n \times n)} \boldsymbol{L}^{\mathsf{T}}$$

avec

- $\boldsymbol{L}$ une matrice triangulaire inférieure
- une matrice est définie positive : $\forall \boldsymbol{x} \neq \boldsymbol{0}$, $\boldsymbol{x}^{\mathsf{T}} \boldsymbol{A} \boldsymbol{x} > 0$

Calcul de pose

Une fois $\boldsymbol{K}$ connue...
Comment évaluer les paramètres externes (pose), $\boldsymbol{R}$ et $\boldsymbol{t}$?

Décomposer $\boldsymbol{N} \propto \boldsymbol{K}^{-1}\boldsymbol{H} = \boldsymbol{R}\begin{bmatrix} 1 & 0 & \\ 0 & 1 & \boldsymbol{t} \\ 0 & 0 & \end{bmatrix}$

- $\rho = 1/||\boldsymbol{n}_1|| = 1/||\boldsymbol{n}_2||$
- $\boldsymbol{r}_1 = \rho\, \boldsymbol{n}_1$
- $\boldsymbol{r}_2 = \rho\, \boldsymbol{n}_2$
- $\boldsymbol{r}_3 = \boldsymbol{r}_1 \times \boldsymbol{r}_2$
- $\boldsymbol{t} = \rho \begin{bmatrix} \boldsymbol{r}_1 & \boldsymbol{r}_2 & \boldsymbol{r}_3 \end{bmatrix}^{-1} \boldsymbol{n}_3$

Optimisation non-linéaire

- Solution linéaire non optimale
- erreur algébrique vs erreur de reprojection
- Minimiser l'erreur de reprojection
- ATTENTION, ce n'est PAS de l'ajustement de faisceau (*Bundle Adjustment*)

Optimisation de l'erreur de reprojection

Pour minimiser l'erreur géométrique, on va tomber dans l'optimisation non linéaire...

Pour toutes les vues i

$$\arg\min_{\boldsymbol{R}_i,\boldsymbol{c}_i,\boldsymbol{K}} \sum_{i,n} \mathrm{dist}(\boldsymbol{p}_i, \boldsymbol{K}\boldsymbol{R}_i \begin{bmatrix} 1 & 0 & \\ 0 & 1 & \boldsymbol{t}_i \\ 0 & 0 & \end{bmatrix} \tilde{\boldsymbol{p}}_i)^2$$

- Plus difficile que pour les homographies
  - Contraintes sur $\boldsymbol{R}$

Problème de la rotation

Première solution: Paramétrisation

- Euler : $\boldsymbol{R} = \boldsymbol{R}_x(r_x)\boldsymbol{R}_y(r_y)\boldsymbol{R}_z(r_z)$

- Fonctions compliquées
- Relativement difficile à optimiser
- Lent

Deuxième solution: Optimisation itérative (recommandée)

- À chaque étape de l'algorithme d'opimisation (*e.g.* Levenberg-Marquardt)
- Rotation par rapport à l'étape précédente est **semblable**
- Simplification de la fonction à chaque itération.
- Nous avons:

$$\boldsymbol{R}^{n+1} = \boldsymbol{R}'\boldsymbol{R}^{n}, \quad \text{ou} \quad \boldsymbol{R}' = \boldsymbol{R}_x(\phi)\boldsymbol{R}_y(\theta)\boldsymbol{R}_z(\rho)$$

- r_x, r_y, r_z sont **petits**
- Approximation de $\boldsymbol{R}'$:

$$\begin{aligned} \cos x &= 1 \\ \sin x &= x \end{aligned} \qquad \boldsymbol{R}' \approx \begin{pmatrix} \theta\rho & -\theta & 1 \\ \rho+\phi & \rho\phi - 1 & -\theta \\ 1-\rho\phi & \rho+\phi & \theta\phi \end{pmatrix}$$

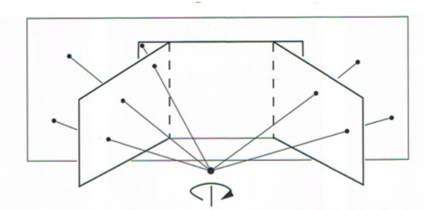
Calibrage par Rotation pure

Définition

Caméra qui tourne parfaitement autour de son centre optique ($\boldsymbol{c} = \boldsymbol{0}$)

R. I. Hartley. Self-calibration from multiple views with a rotating camera. InProc. 3rdEuropean Conf. on Computer Vision, Stockholm, volume 1, pages 471–478, 1994

- Aucun effet de profondeur
- Images de caméra reliée par des homographies



Deux caméras avec

- $\boldsymbol{H}_1 = \boldsymbol{K}$ (sans rotation, ni translation)
- $\boldsymbol{H}_2 = \boldsymbol{K}\boldsymbol{R}$

Relier ensemble en faisant avec $\text{Image}_1 \xrightarrow{H_1^{-1}} \text{Monde} \xrightarrow{H_2} \text{Image}_2$:

$$\boldsymbol{H}_{12} = \boldsymbol{K}\boldsymbol{R}\boldsymbol{K}^{-1}$$

Retrouver la rotation

Rotation conjuguée

Une matrice $\boldsymbol{H} = \boldsymbol{URU}^{-1}$, où $\boldsymbol{R}$ est une matrice orthogonale et $\boldsymbol{U}$ est une transformation projective.

Pour une matrice de rotation $\boldsymbol{R}$, nous avons

- valeurs propres de la forme $\lambda(1, e^{i\theta}, e^{-i\theta)}$,
  (une réelle et deux complexes conjuguées)
- vecteurs propres correspondants $(\boldsymbol{a}, \boldsymbol{i}, \boldsymbol{j})$

  θ l'angle

  $\boldsymbol{a}$ axe de rotation de la matrice, (et $\boldsymbol{i}, \boldsymbol{j}$ perpendiculaires à $\boldsymbol{a}$)

Pour une rotation conjuguée, l'angle est préservé, et les vecteurs propres sont prémultipliés par $\boldsymbol{U}$.

Exemple simple: ratio d'image à 1, point principal connu et mis à l'origine

Supposons $\boldsymbol{K} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix}$. Dans ce cas $\boldsymbol{H} = \begin{bmatrix} r_1 & r_2 & fr_3 \\ r_4 & r_5 & fr_6 \\ \frac{r_7}{f} & \frac{r_8}{f} & r_9 \end{bmatrix}$

- $\boldsymbol{H}$: presque une matrice de rotation
- solutions analytiques pour f

$$h_1^2 + h_2^2 + h_3^2/f^2 = h_4^2 + h_5^2 + h_6^2/f^2$$

ou bien

$$h_1 h_4 + h_2 h_5 + \frac{h_3 h_6}{f^2} = 0$$

Distortion radiale

Distortion radiale

On a supposé les images absentes de toute distortion géométrique

Modèle

La distorsion radiale s'applique sur les points dans la caméra normalisée

$$\hat{\boldsymbol{p}} = \boldsymbol{K}^{-1}\tilde{\boldsymbol{p}} = (\hat{x}, \hat{y}, 1)^{\mathsf{T}}$$

Modèle :

$$\begin{pmatrix} x^* \\ y^* \end{pmatrix} = L(\hat{r}) \begin{pmatrix} \hat{x} \\ \hat{y} \end{pmatrix}$$

avec

- $\hat{r}^2 = (\hat{x}^2 + \hat{y}^2)$ et k_1, k_2 les coefficients de distorsion
- $L(\hat{r}) = (1 + k_1\hat{r}^2 + k_2\hat{r}^4)$

On peut ensuite calculer les coordonnées dans l'image :

$$\tilde{\boldsymbol{p}} = \boldsymbol{K}(x^*, y^*, 1)^{\mathsf{T}}$$

(a) (b) (c)

Figure: (a) Barrel (b) Pincushion (c) Fish-eye

Correction de la distorsion

Pour corriger la distorsion radiale, il faut inverser la fonction de distortion

Avec $\boldsymbol{K} = \begin{bmatrix} f & 0 & cx \\ 0 & f & cy \\ 0 & 0 & 1 \end{bmatrix}$ et (cx, cy) le centre de distortion :

- $x = fx^* + cx = fL(\hat{r})\hat{x} + cx$ et sa version corrigée $x_d = f\hat{x} + cx$
- il faut connaître les coefficients de distorsion
- il faut retrouver $\hat{r}$ pour chaque point
- une solution : construire une table $L(\hat{r})$ et l'utiliser pour retrouver $\hat{r}$ en fonction de $r^* = \sqrt{(x^*)^2 + (y^*)^2}$

On peut aussi voir le problème comme un problème d'inverse-mapping :

- on peut calculer $\boldsymbol{p}_d$ à partir de $\boldsymbol{p}$ en appliquant la formule
- on trouve l'image dé-distorsionnée en allant de l'image corrigée vers l'image distorsionnée

Estimer les paramètres de distorsion

Une méthode :

- Supposez le centre de distorsion au centre de l'image, et la calibration "connue" (ou proche)
- Itérez jusqu'à ce que le modèle minimise l'erreur algébrique par exemple :
  1. Initialisez k_1 et k_2
  2. Calculez $\hat{x}$ et $\hat{y}$
  3. Posez un système $\boldsymbol{Ax} = \boldsymbol{b}$ avec $\boldsymbol{x} = (k_1, k_2)^\mathsf{T}$ en utilisant le modèle plus haut
  4. Résoudre avec la pseudo-inverse (ou SVD)
  5. Recommencez en 2) jusqu'à convergence
- Éventuellement, réestimez les paramètres internes et la distorsion radiale avec une méthode non-linéaire

Autre méthode

- La méthode standard est plutôt d'utiliser une minimisation non linéaire de la distance géométrique entre des correspondances (e.g points 3D-2D comme dans DLT), en incluant tous les paramètres
- L'ajustement de faisceaux permet (entre autres) de retrouver les paramètres internes de $\boldsymbol{K}$, la distorsion radiale, la pose et la meilleure position des points 3D qui minimise la distance géométrique
- Ces méthodes (non linéaires) donnent souvent de meilleurs résultats :
  - minimisent l'erreur géométrique
  - doivent être bien paramétrisées
  - exigent un bon estimé d'une solution de départ