SUIVI DE MOUVEMENT POUR LA RV/RA

Sources :

- Steven M. LaValle (http://vr.cs.uiuc.edu)
- Gordon Wetzstein (https://stanford.edu/class/ee267/)
- Hannes Kaufmann (https://www.ims.tuwien.ac.at/)
- Vincent Lepetit (https://www.labri.fr/perso/vlepetit/augmented_reality.php)
- Silvio Savarese (http://web.stanford.edu/class/cs231a/)
- Davide Scaramuzza (http://rpg.ifi.uzh.ch/teaching.html)

Objectifs

Mesurer et suivre dans le temps la position et/ou l'orientation d'un « objet » en 3D

Que suivre ?

- l'utilisateur
  (tête, mains, corps, yeux, expression faciale)
- l'environnement

Suivi de l'orientation

Objectif principal

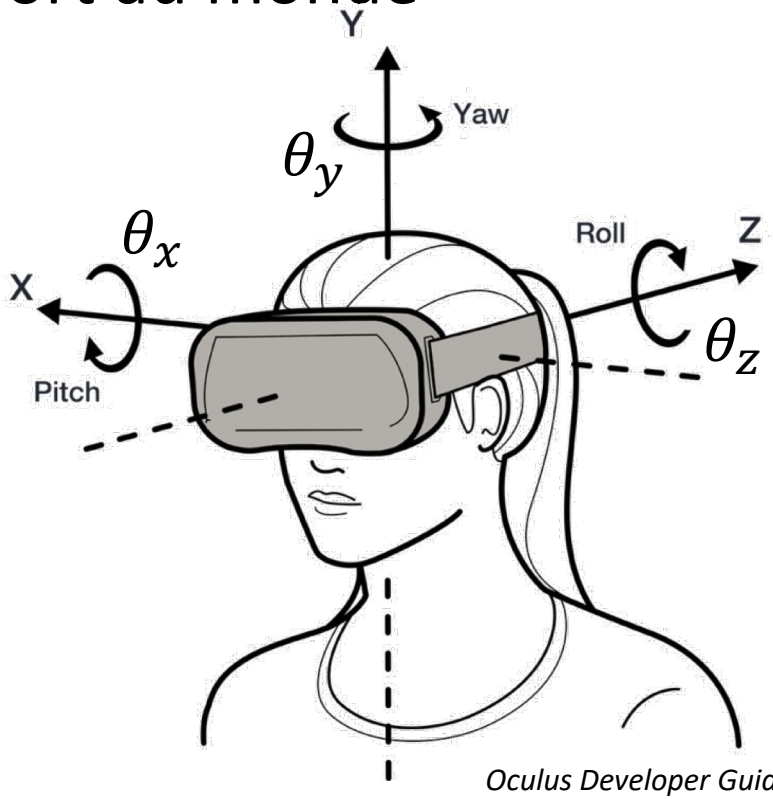
Mesurer l'orientation de la tête ou d'un périphérique (manette, smartphone)

Orientation = rotation par rapport au monde (référentiel inertiel)

Rotation représentée par :

- les angles d'Euler

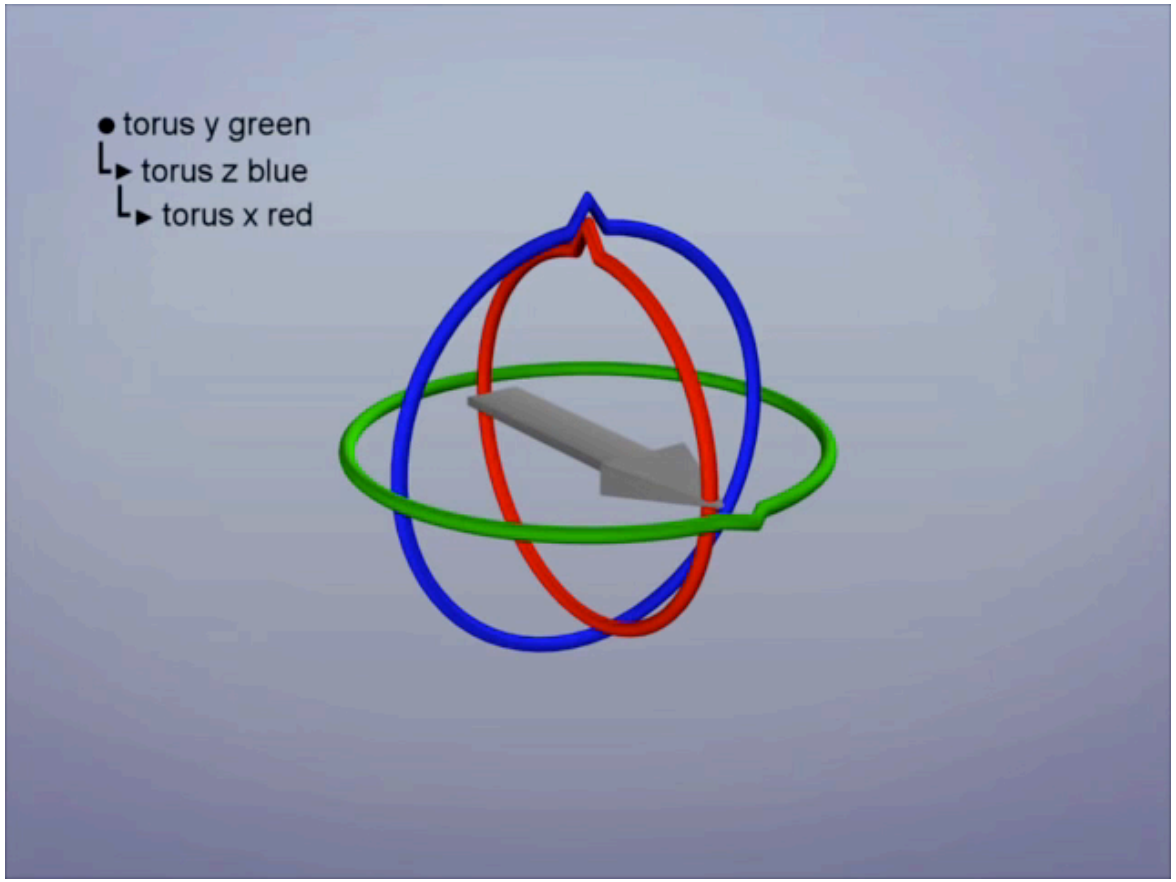
$$R = R_z(-\theta_z)R_x(-\theta_x)R_y(-\theta_y)$$



Oculus Developer Guide

Limitations des angles d'Euler

"Gimbal lock" : perte d'un degré de liberté



http://www.youtube.com/watch?v=zc8b2Jo7mno

Objectif principal

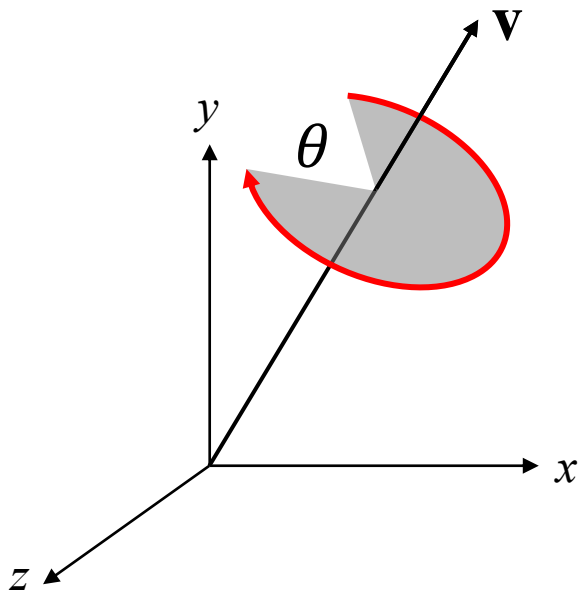
Mesurer l'orientation de la tête ou d'un périphérique (manette, smartphone)

Orientation = rotation par rapport au monde (référentiel inertiel)

Rotation représentée par :

- les angles d'Euler, ou
- un quaternion

$$q(\theta, \mathbf{v}) = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) \\ \mathbf{v}\sin\left(\frac{\theta}{2}\right) \end{pmatrix}$$
$$||\mathbf{v}|| = 1$$



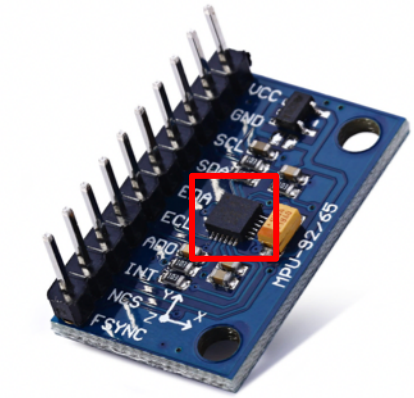
Inertial Measurement Unit (IMU)

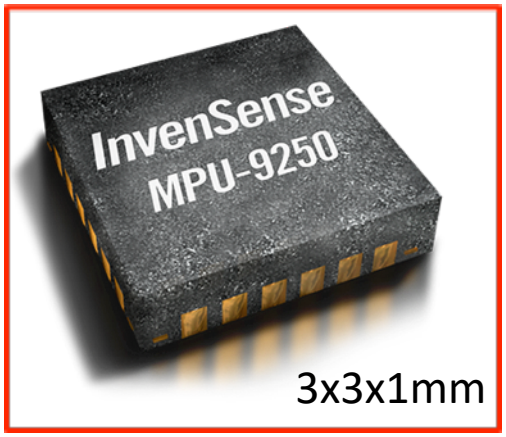
- **Gyroscope** : vitesse angulaire en radians/seconde

- **Accéléromètre** : accélération linéaire en m/s

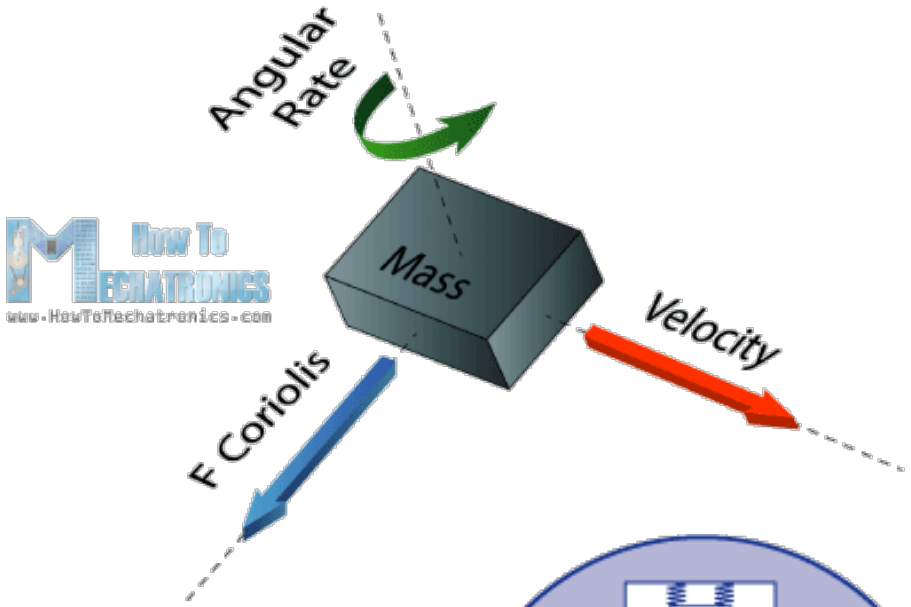
- **Magnétomètre** : intensité champ magnétique
  en μT (microtesla) ou Gauss (1 Gauss = 100 μT)

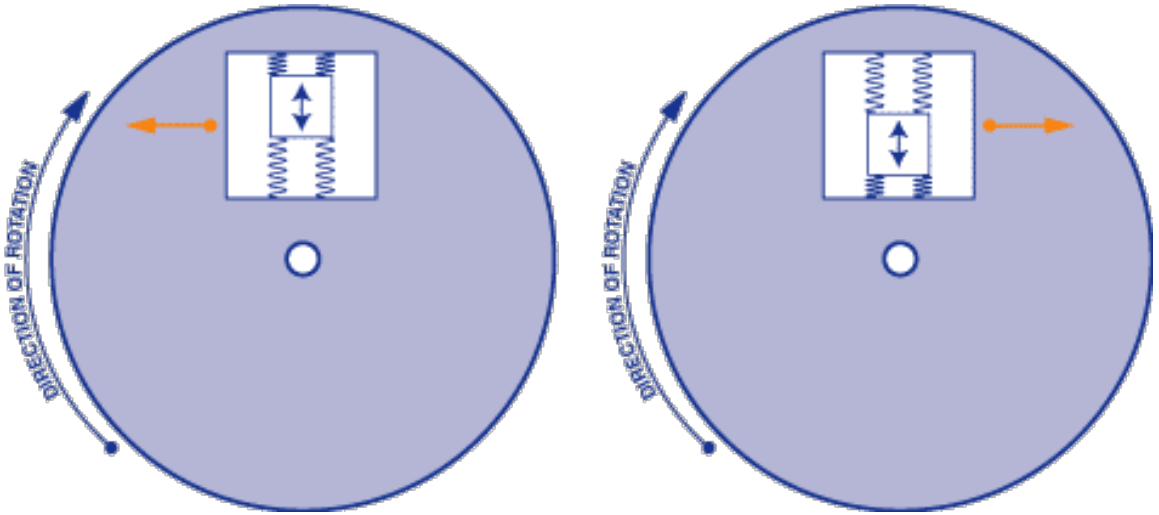
Attention : mesurés dans le repère du périphérique !





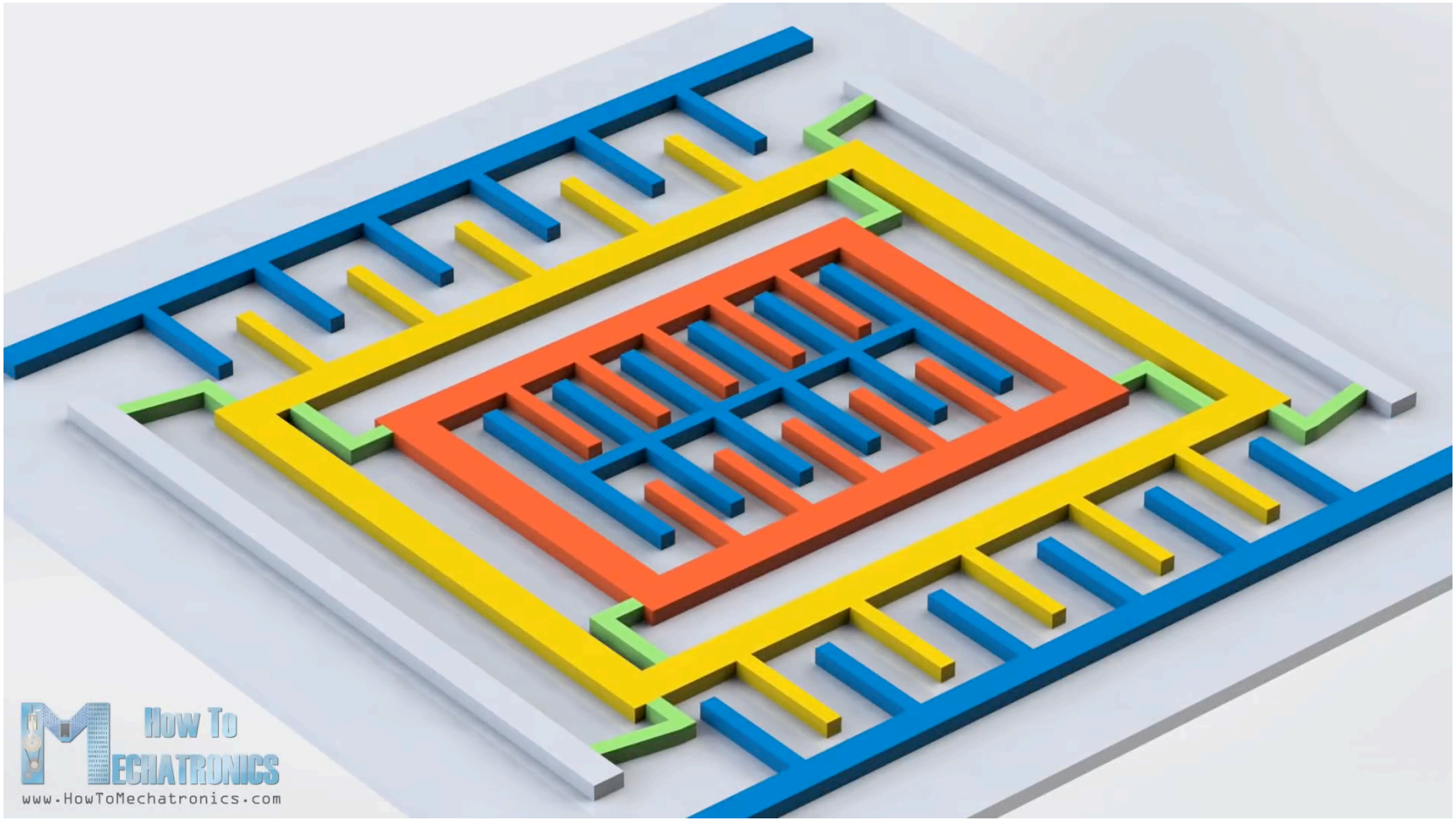
MEMS Gyroscope





https://learn.sparkfun.com/tutorials/gyroscope/all

MEMS Gyroscope



https://howtomechatronics.com/how-it-works/electrical-engineering/mems-accelerometer-gyrocope-magnetometer-arduino/

Vitesse angulaire

$$\omega = \frac{d\theta(t)}{dt}$$

donc $\theta(t) = \theta(0) + \int_0^t \omega(s)ds$

initialisation

intégration

Gyroscopes

Mesure imparfaite : $\tilde{\omega} = b + a\,\omega + \eta$

- biais b
- facteur d'échelle a

$\Big\}$ constant $\Longrightarrow$ calibration

- bruit Gaussien aditif $\Longrightarrow$ incertitude

Problèmes à résoudre

1. Calibration
2. Initialisation
3. Intégration
4. Propagation des incertitudes

Calibration

Mesurer simultanément $\tilde{\omega}'$ avec un gyroscope de meilleure qualité (et/ou déjà calibré)

Calculer la somme des erreurs au carré :

$$\Sigma_{i=1}^{N} (\tilde{\omega}_i - \tilde{\omega}'_i)^2$$

Trouver a et b minimisant cette erreur :

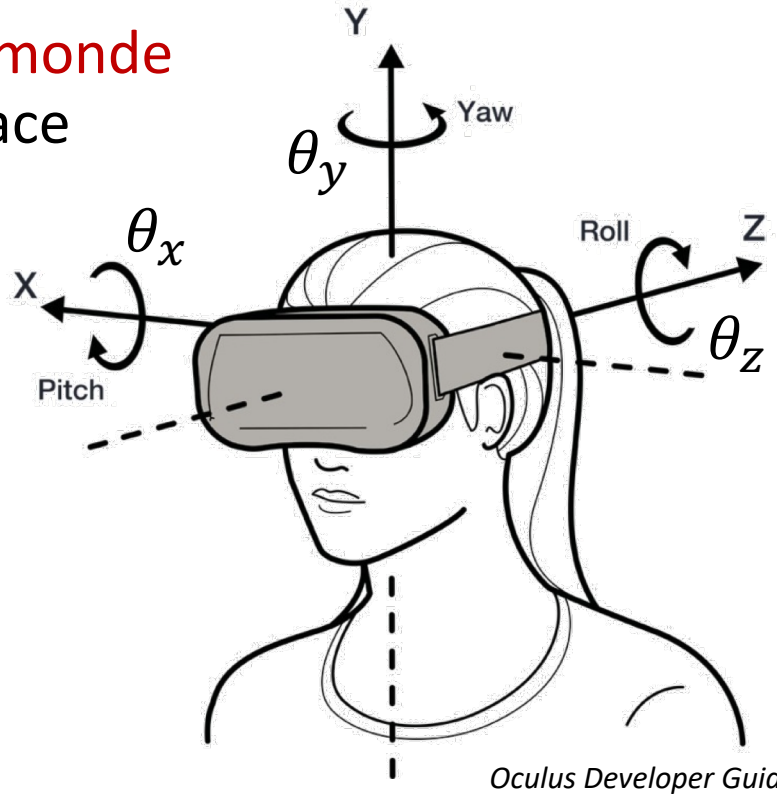
$$\Sigma_{i=1}^{N} (b + a\,\omega_i - \tilde{\omega}'_i)^2$$

(régression linéaire au sens des moindres carrées)

Initialisation

Orientation initiale

- $\theta(0) = 0$ dans le repère local du gyroscope
  $\Longrightarrow$ état au lancement du système
- $\theta(0) = 0$ dans le repère monde
  $\Longrightarrow$ capteur fixe dans l'espace



Oculus Developer Guide

Intégration

Développement limité d'ordre 1 (Taylor)
= approximation linéaire de la rotation

erreur d'approximation
$\Longrightarrow$ incertitude

angle précédent

$$\theta(t+\Delta t) \approx \theta(t) + \frac{d}{dt}\theta(t)\Delta t + O(\Delta t^2)$$

angle courant

$= \omega$

vitesse angulaire

Intégration

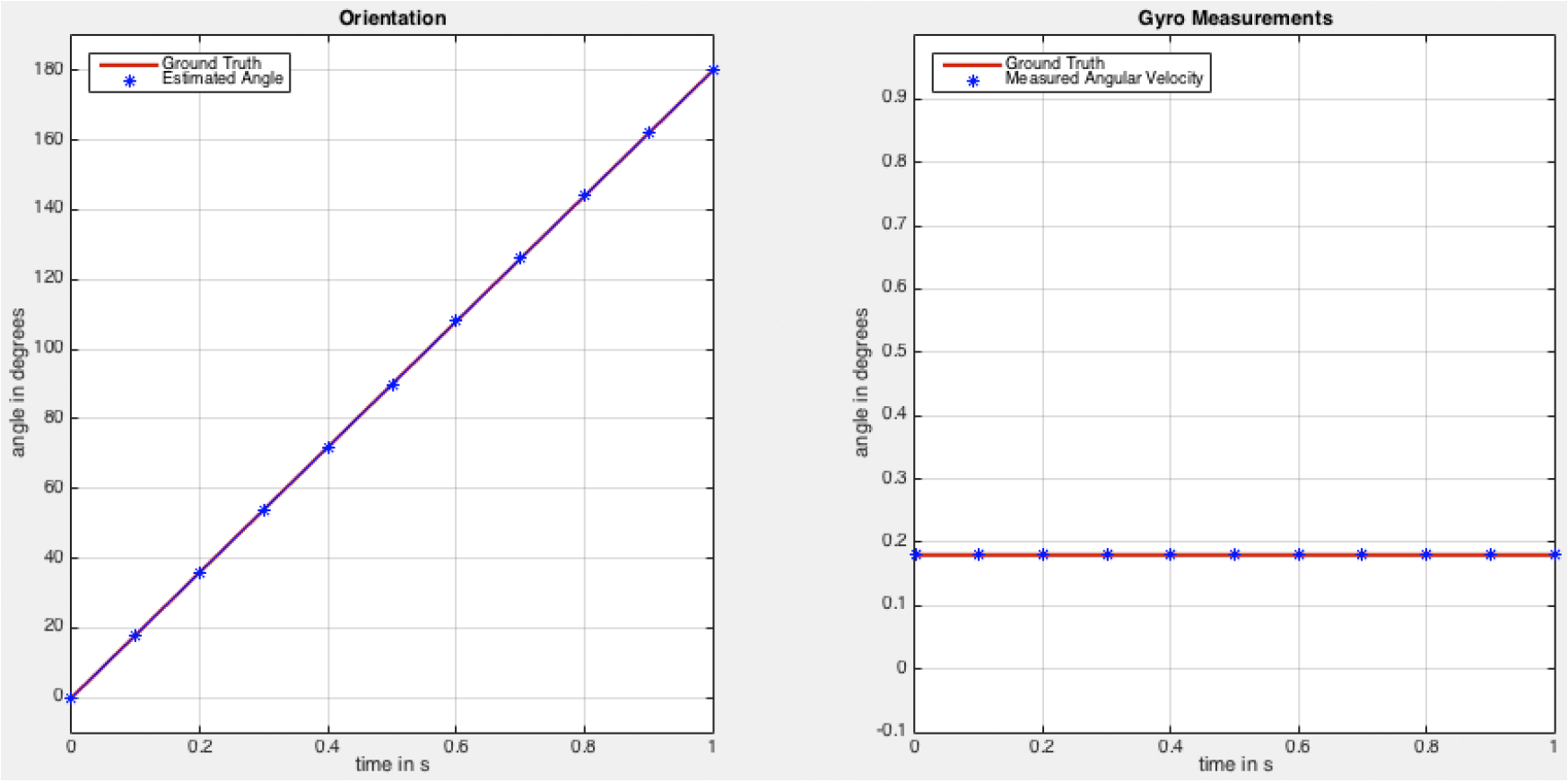
Discrétisation des mesures

$$\tilde{\theta}[k] = \tilde{\theta}[k-1] + \tilde{\omega}[k]\Delta t$$

avec Δt le temps entre deux mesures.

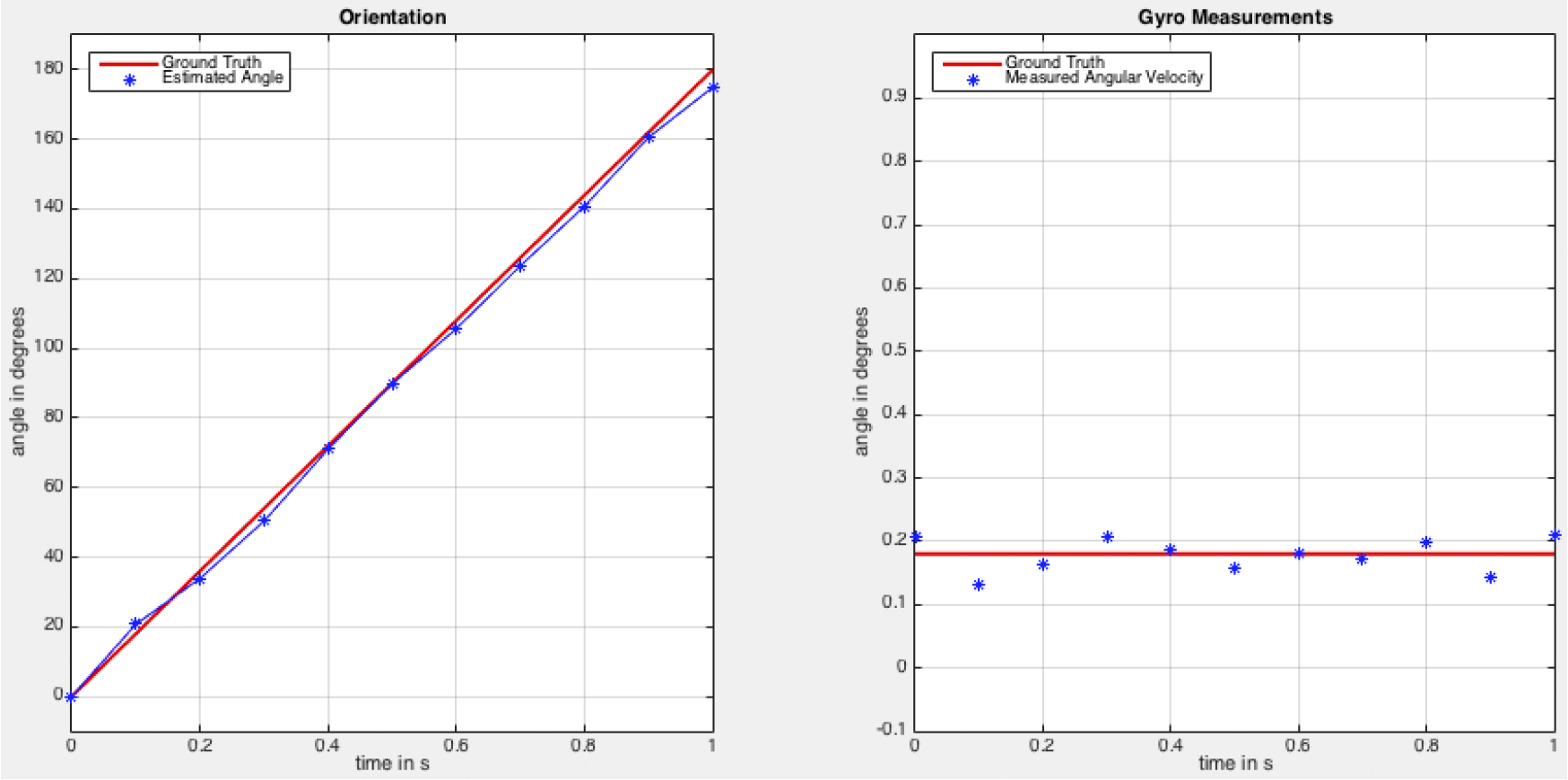
Propagation des incertitudes

Rotation linéaire en angle, pas de bruit



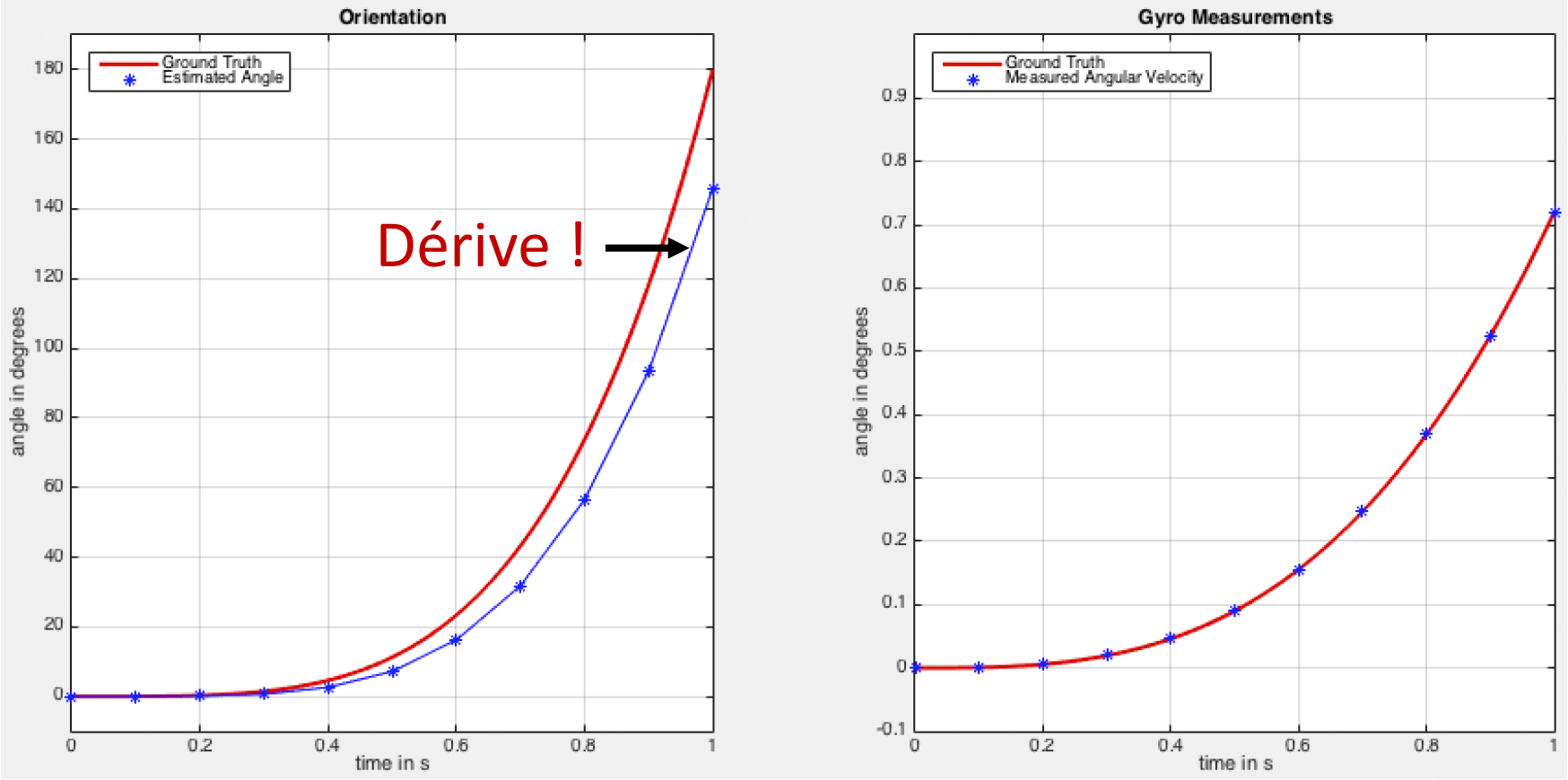
Propagation des incertitudes

Rotation linéaire en angle, **bruit**



Propagation des incertitudes

Rotation **non-linéaire**, pas de bruit



« Navigation à l'estime »

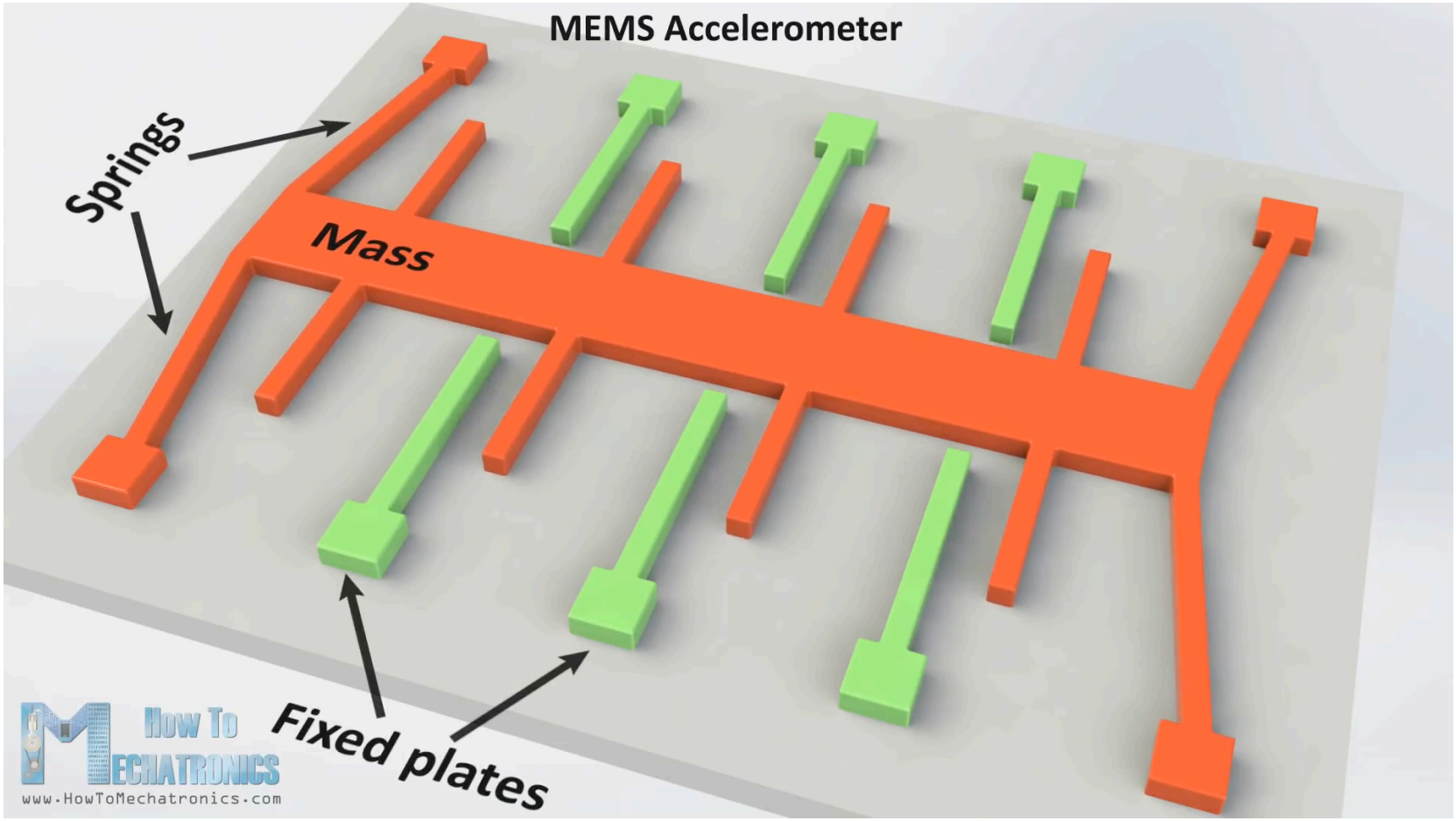
position initiale + nord magnétique (compas)
+ vitesse du navire (loch) + temps de parcours

⟹ **position** avec 2 à 5% d'incertitude

⟹ dérive

J. S. Bond – Wikipedia

MEMS accéléromètre



https://howtomechatronics.com/how-it-works/electrical-engineering/mems-accelerometer-gyrocope-magnetometer-arduino/

Accéléromètre

Mesure : $\tilde{a} = a^{(g)} + a^{(l)} + \eta$

gravité

accélération linéaire

bruit Gaussien

$a^{(g)}$ vecteur pointant vers le haut
avec une magnitude de 9.81 m/s^2 = 1g

Accéléromètre

Avantages

- ✓ pointe en moyenne vers le haut (*up vector*)
- ✓ pas de dérive dans le temps
  (le centre de gravité de la terre est fixe)

Inconvénients

- ✗ mesure bruitée
- ✗ peu fiable sur un temps court
  à cause du mouvement

$\Longrightarrow$ complémentaire au gyroscope

$\Longrightarrow$ correction de la dérive du *tilt* (*pitch* & *roll*)

Suivi de l'orientation en 2D

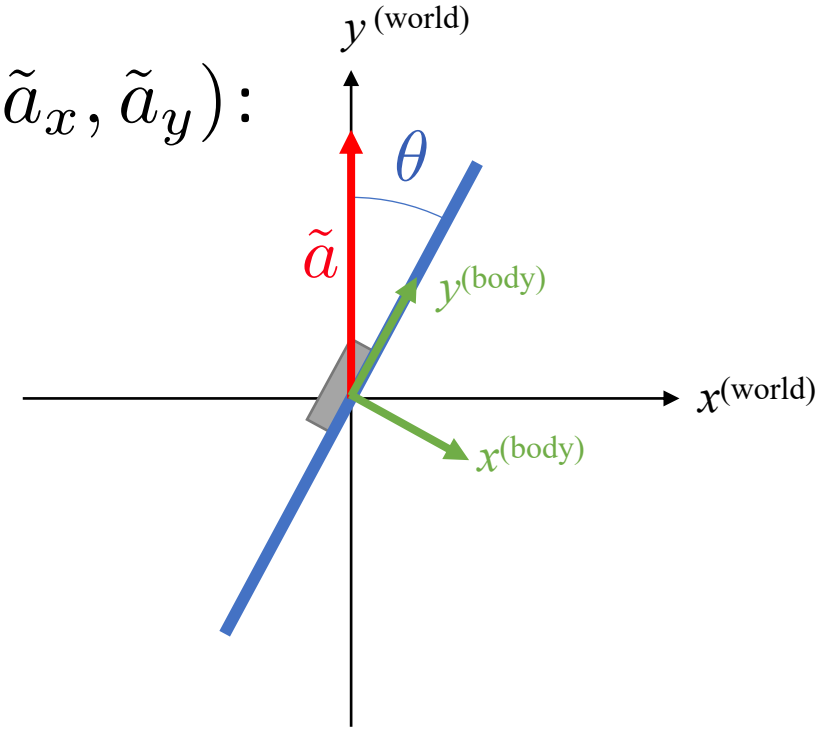
Un gyroscope :

$$\tilde{\theta}[k] = \tilde{\theta}[k-1] + \tilde{\omega}[k]\Delta t$$

Accéléromètre 2 axes : $\tilde{\mathbf{a}} = (\tilde{a}_x, \tilde{a}_y)$:

$$\theta_{acc} = \tan^{-1}\left(\frac{\tilde{a}_x}{\tilde{a}_y}\right)$$

$\Longrightarrow$ combiner et filtrer
les deux mesures
= ***sensor fusion***

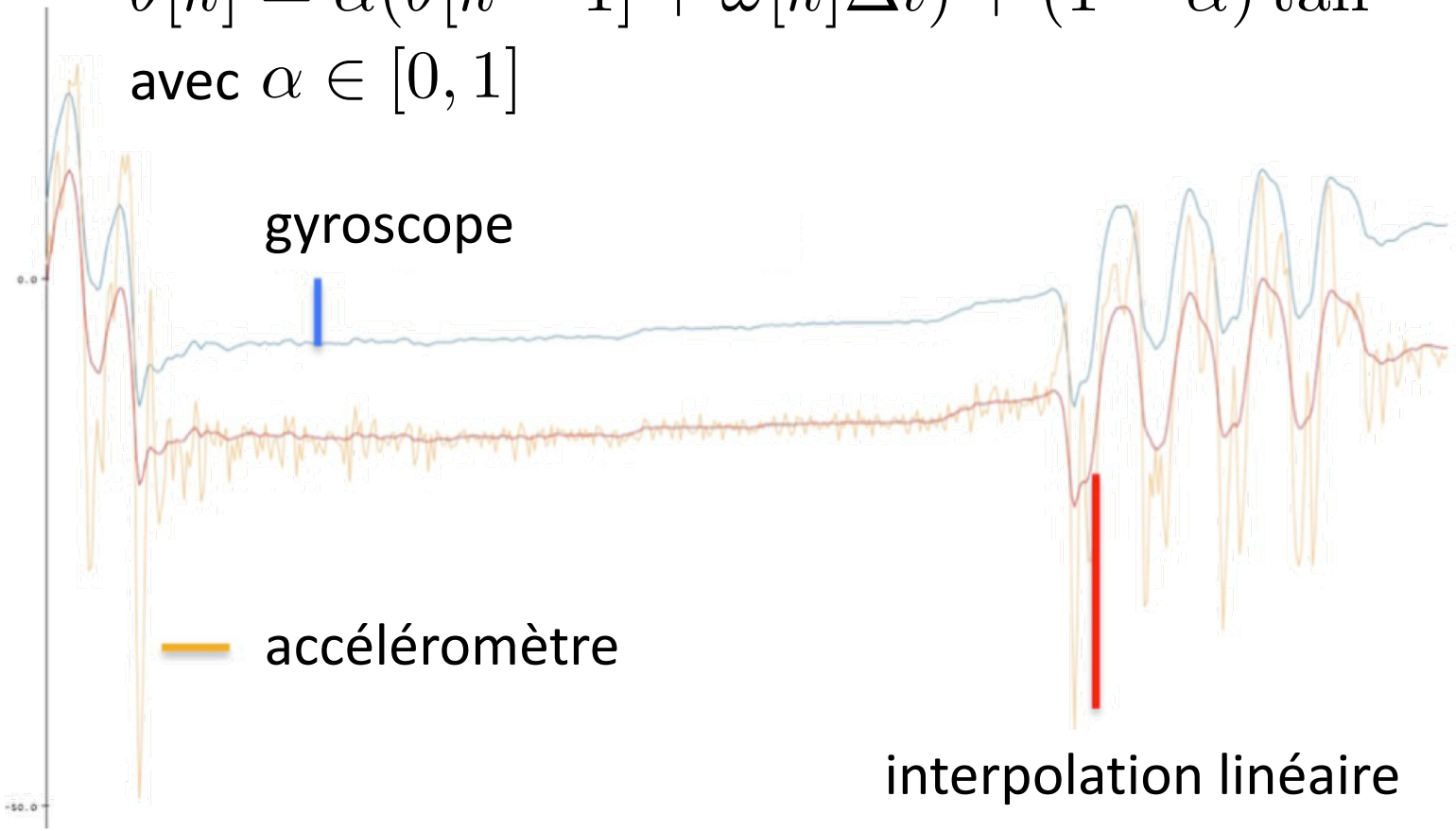


Sensor fusion

Filtre simple : interpolation linéaire :

$$\tilde{\theta}[k] = \alpha(\tilde{\theta}[k-1] + \tilde{\omega}[k]\Delta t) + (1-\alpha)\tan^{-1}\left(\frac{\tilde{a}_x}{\tilde{a}_y}\right)$$

avec $\alpha \in [0, 1]$



Sensor fusion

Interpolation linéaire :

$$\tilde{\theta}[k] = \alpha(\tilde{\theta}[k-1] + \tilde{\omega}[k]\Delta t) + (1-\alpha)\tan^{-1}\left(\frac{\tilde{a}_x}{\tilde{a}_y}\right)$$

avec $\alpha \in [0, 1]$

Gyroscope dérive **progressivement** dans le temps

$\Longrightarrow$ choisir α proche de 1

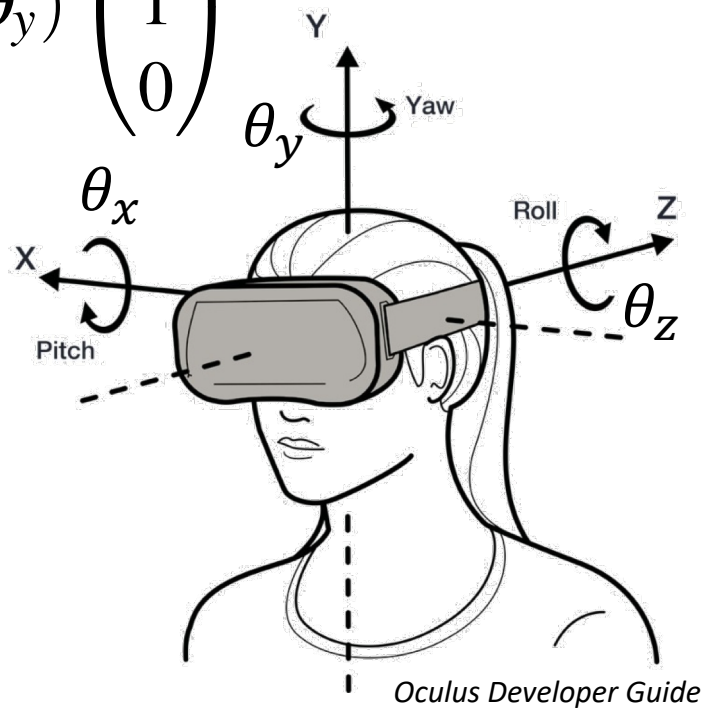
Mesure du *tilt* (*pitch* & *roll*) en 3D

Accéléromètres à 3 axes et pas de forces externes

up-vector dans le repère monde

$$\hat{a} = \frac{\tilde{a}}{||\tilde{a}||} = R_z(-\theta_z)R_x(-\theta_x)R_y(-\theta_y)\begin{pmatrix}0\\1\\0\end{pmatrix}$$

vecteur gravité normalisé
dans le repère du capteur



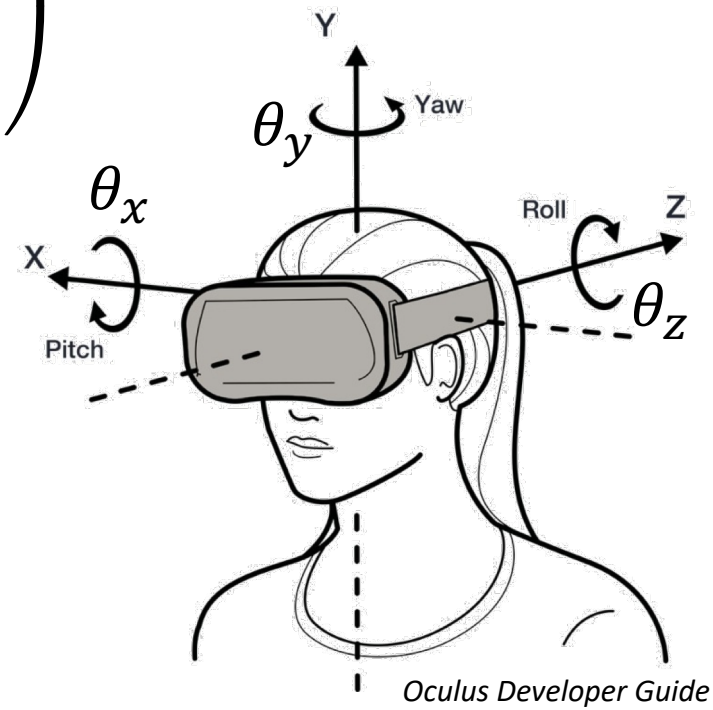
Oculus Developer Guide

Mesure du *tilt* (*pitch* & *roll*) en 3D

Accéléromètres à 3 axes et pas de forces externes

$$\hat{a} = \frac{\tilde{a}}{||\tilde{a}||} = \begin{pmatrix} -\cos(-\theta_x)\sin(-\theta_z) \\ \cos(-\theta_x)\cos(-\theta_z) \\ \sin(-\theta_x) \end{pmatrix}$$

$\Longrightarrow$ indépendant de θ_y

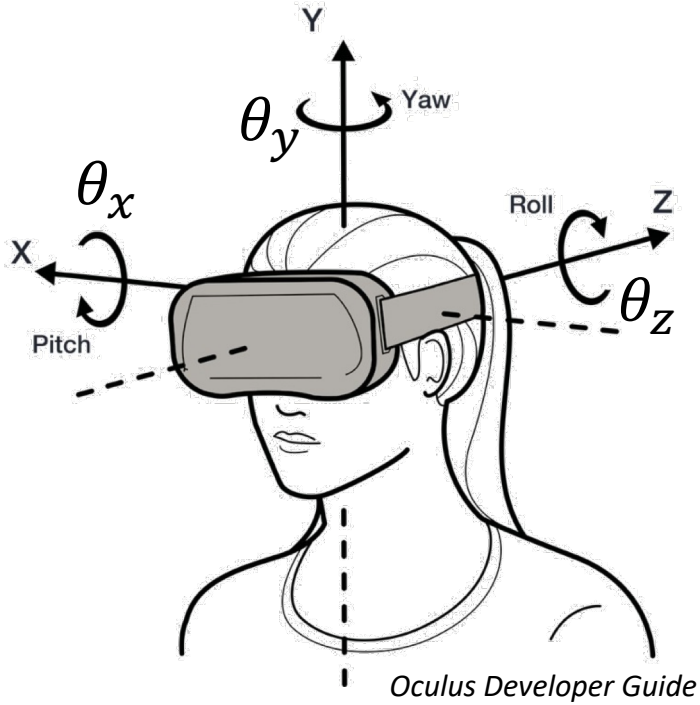


Oculus Developer Guide

Mesure du *roll* θ_z

$$\frac{\hat{a}_x}{\hat{a}_y} = \frac{-\sin(-\theta_z)}{\cos(-\theta_z)} = -\tan(-\theta_z)$$

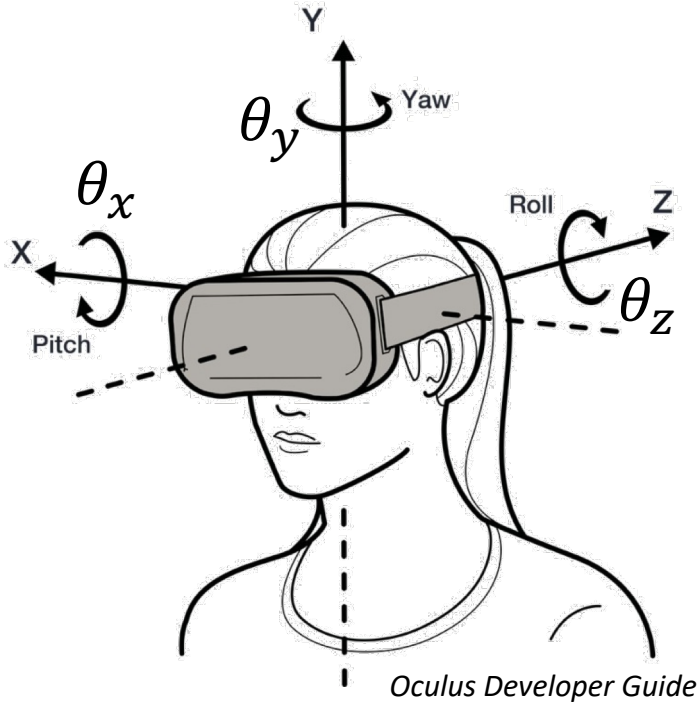
$$\Longrightarrow \theta_z = -\tan^{-1}\left(-\frac{\hat{a}_x}{\hat{a}_y}\right)$$



Oculus Developer Guide

Mesure du *pitch* θ_x

$$\theta_x = -\sin^{-1}(\hat{a}_z)$$ ambigu



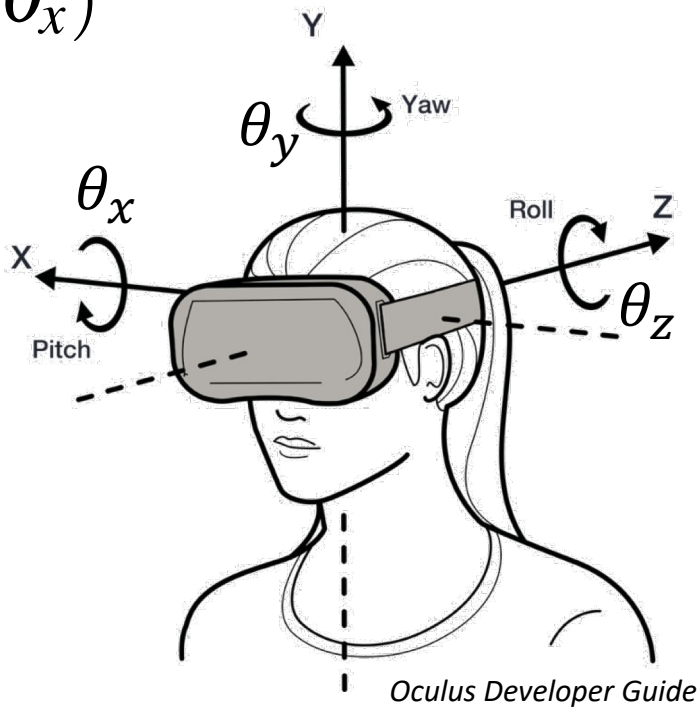
Oculus Developer Guide

Mesure du *pitch* θ_x

$$\frac{\hat{a}_z}{\sqrt{\hat{a}_x^2+\hat{a}_y^2}} = \frac{\sin(-\theta_x)}{\sqrt{\cos^2(-\theta_x)\underbrace{(\sin^2(-\theta_z)+\cos^2(-\theta_z))}_{=1}}}$$

$$= \frac{\sin(-\theta_x)}{\cos(-\theta_x)} = \tan(-\theta_x)$$

$$\Longrightarrow \theta_x = -\tan^{-1}\left(\frac{\hat{a}_z}{\sqrt{\hat{a}_x^2+\hat{a}_y^2}}\right)$$



Oculus Developer Guide

Suivi de l'orientation 3D

1. **Intégration** de la vitesse mesurée par un gyroscope 3 axes
2. **Calcul du *tilt***
   à partir d'un accéléromètre 3 axes
3. ***Sensor fusion***

Gyroscope 3 axes

Mesure le vecteur 3D : $\tilde{\omega} = (\tilde{\omega}_x, \tilde{\omega}_y, \tilde{\omega}_z)$

- axe de rotation : $\frac{\tilde{\omega}}{||\tilde{\omega}||}$
- angle : $||\tilde{\omega}||\Delta t$

} quaternion $\Delta\tilde{q}$

Intégration : $\tilde{q}[k] = \tilde{q}[k-1]\Delta\tilde{q}[k]$

multiplication de deux quaternions

Initialisation : $\tilde{q}[0] = \begin{pmatrix} 1 \\ \mathbf{0} \end{pmatrix}$

Accéléromètre 3 axes

Mesure le vecteur 3D : $\tilde{\mathbf{a}} = (\tilde{a}_x, \tilde{a}_y, \tilde{a}_z)$

…dans le repère du capteur

Transformation dans l'espace monde :

$$q_a^{(\text{world})} = \tilde{q}[k] q_a^{(\text{body})} \tilde{q}[k]^{-1}$$

avec : $q_a^{(\text{body})} = \begin{pmatrix} 0 \\ \tilde{\mathbf{a}} \end{pmatrix}$

Correction du *tilt*

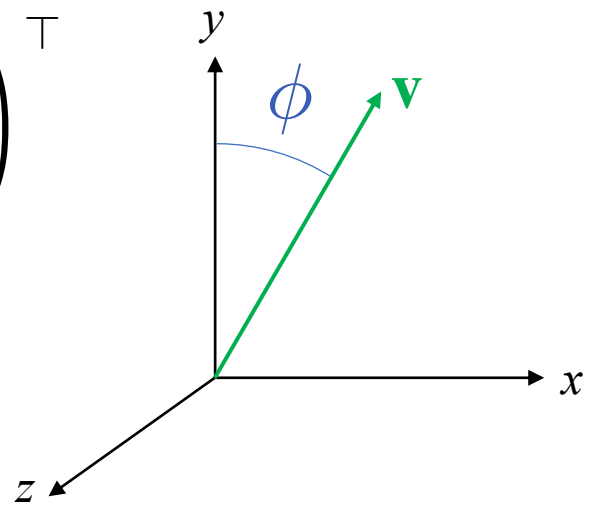
Si les capteurs étaient **parfaits** :

$$q_a^{(\text{world})} = (0,\ 9.81,\ 0)^\top$$

…mais dérive du gyroscope et bruit de l'accéléromètre

Solution : correction du *tilt* pour aligner $q_a^{(\text{world})}$ avec le *up-vector*

$$\mathbf{v} = \left(\frac{q_{a_x}^{(\text{world})}}{||q_a^{(\text{world})}||}, \frac{q_{a_y}^{(\text{world})}}{||q_a^{(\text{world})}||}, \frac{q_{a_z}^{(\text{world})}}{||q_a^{(\text{world})}||} \right)^\top$$



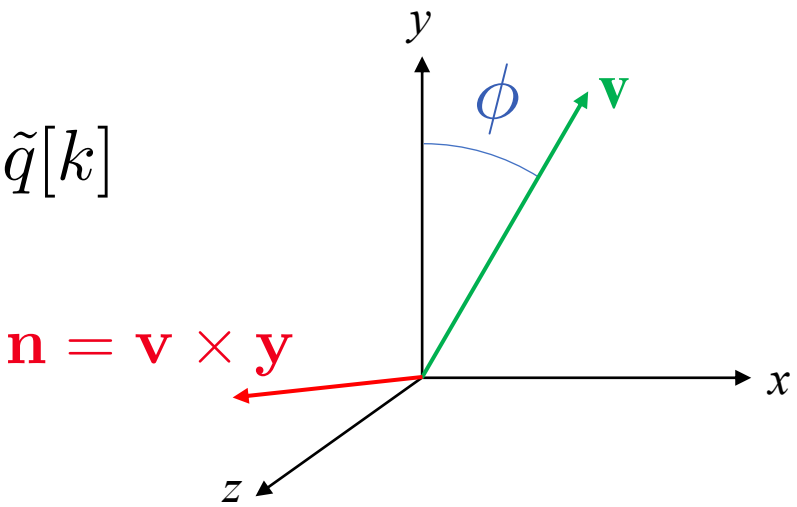
Correction du *tilt*

$$\left.\begin{aligned} \cos(\phi) &= \mathbf{v} \cdot \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \Rightarrow \phi = \cos^{-1}(v_y) \\ \mathbf{n} &= \mathbf{v} \times \mathbf{y} = \begin{pmatrix} -v_z \\ 0 \\ v_x \end{pmatrix} \end{aligned}\right\} q_{\text{tilt}} = q\left(\phi, \frac{\mathbf{n}}{||\mathbf{n}||}\right)$$

***Sensor fusion* :**

$$\tilde{q}_c[k] = q\left((1-\alpha)\phi, \frac{\mathbf{n}}{||\mathbf{n}||}\right)\tilde{q}[k]$$

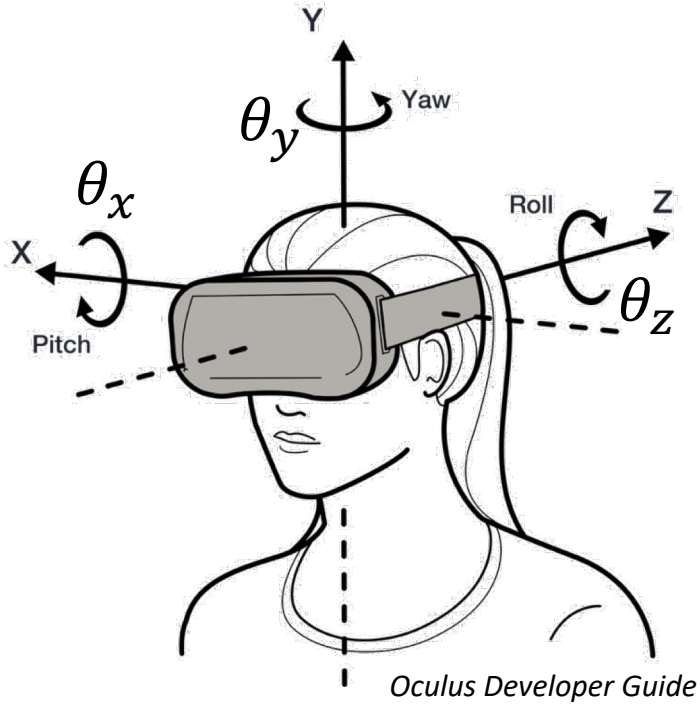
$$\alpha \in [0, 1]$$



Correction du *yaw*

Nécessité d'un compas (pointant vers le Nord)

⟹ **magnétomètre**

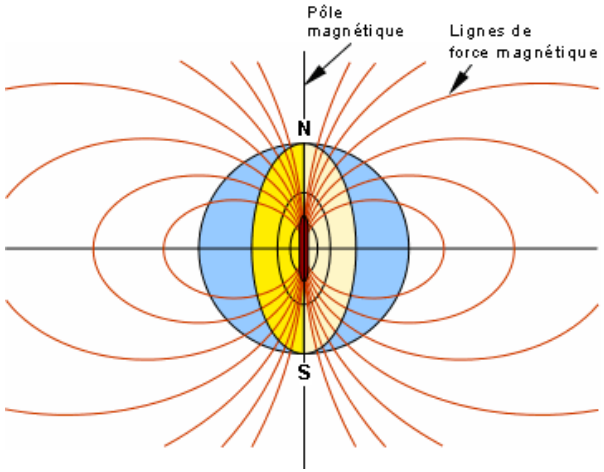


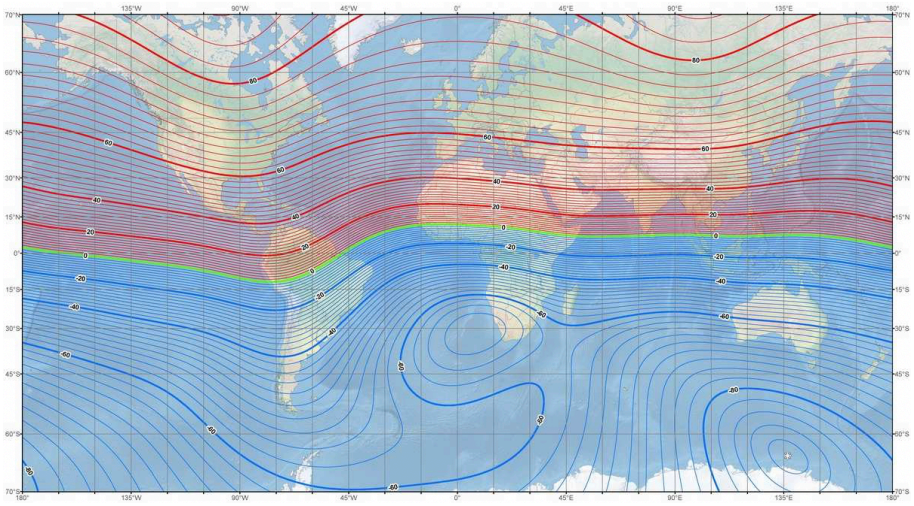
Oculus Developer Guide

Magnétomètre

Mesure le champ magnétique

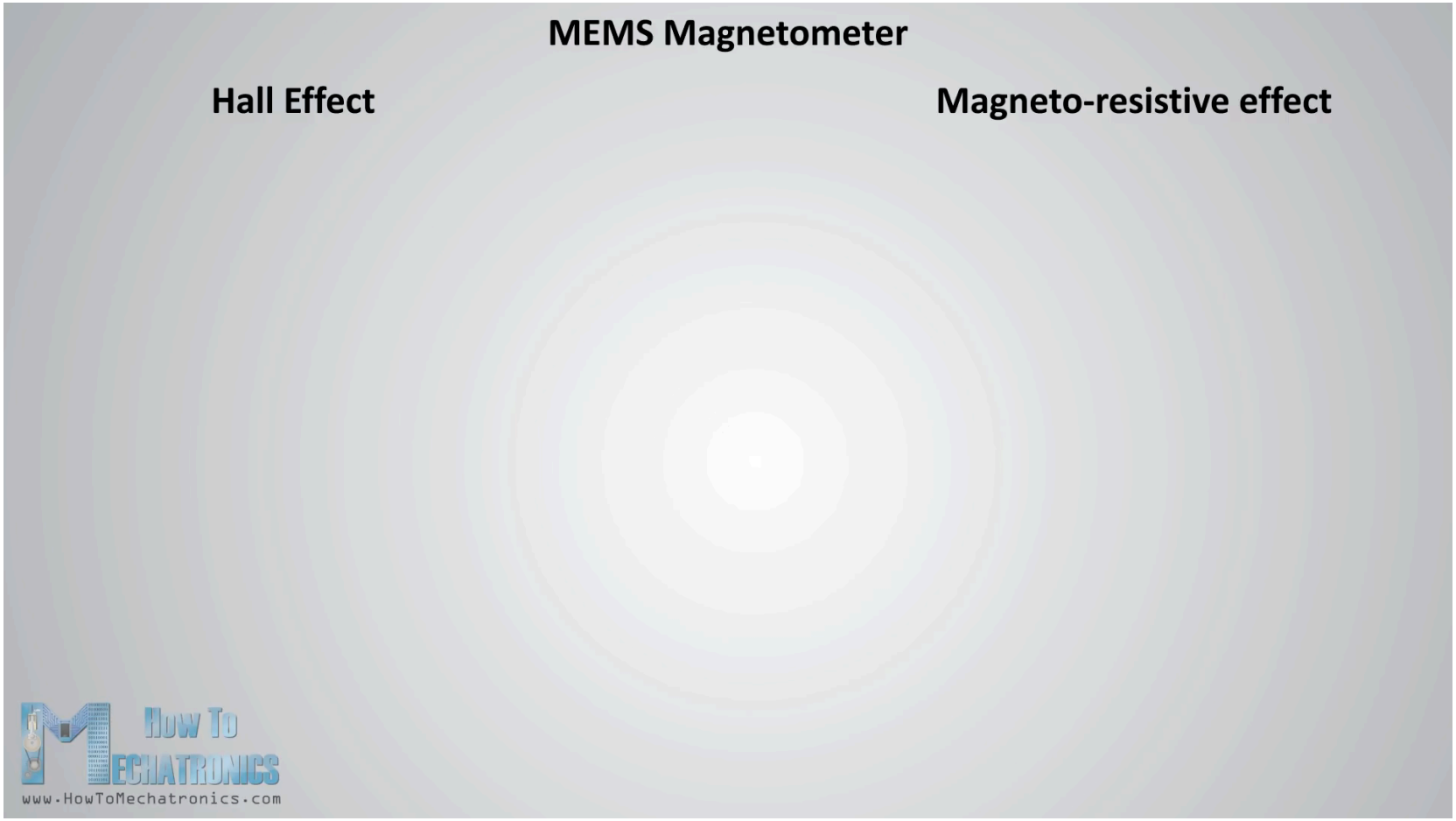
- vecteur 3D : **pas horizontal**
  ⟹ projection sur le plan xz
- **perturbation** par les matériaux ferromagnétiques (dans le circuit)
  ⟹ calibration





NOAA/NGDC & CIRES

MEMS magnétomètre



Suivi de la position et de l'orientation

Utiliser l'accéléromètre ?

Mesure : $\tilde{a} = a^{(g)} + a^{(l)} + \eta$

gravité — accélération linéaire — bruit Gaussien

Intégrer deux fois $a^{(l)}$ pour obtenir la position

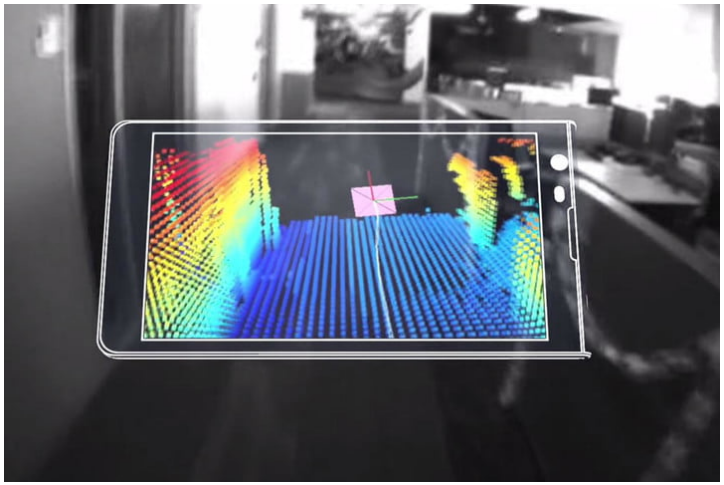
$\Longrightarrow$ dérive de la mesure croît **quadratiquement** !

$\Longrightarrow$ pas de correction avec un IMU seul…

Inside-Out / Outside-In tracking

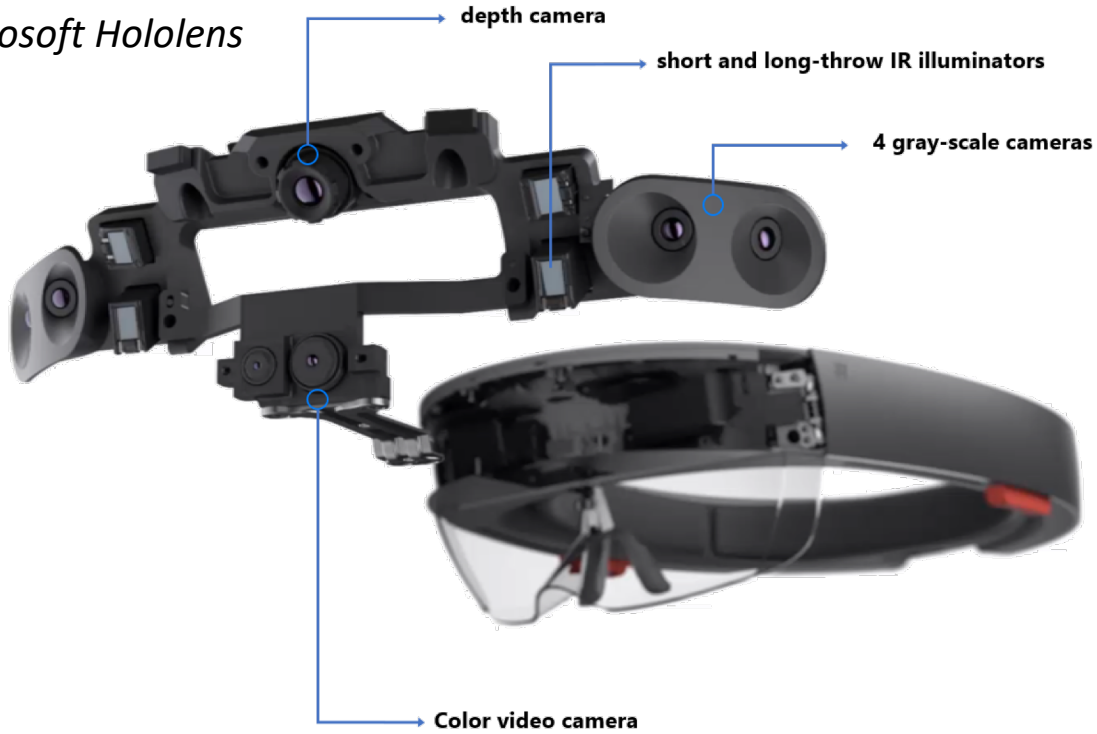
Inside-Out : capteurs sur le dispositif à localiser

⟹ nécessite de cartographier l'environnement
Simultaneous Localization And Mapping (SLAM)



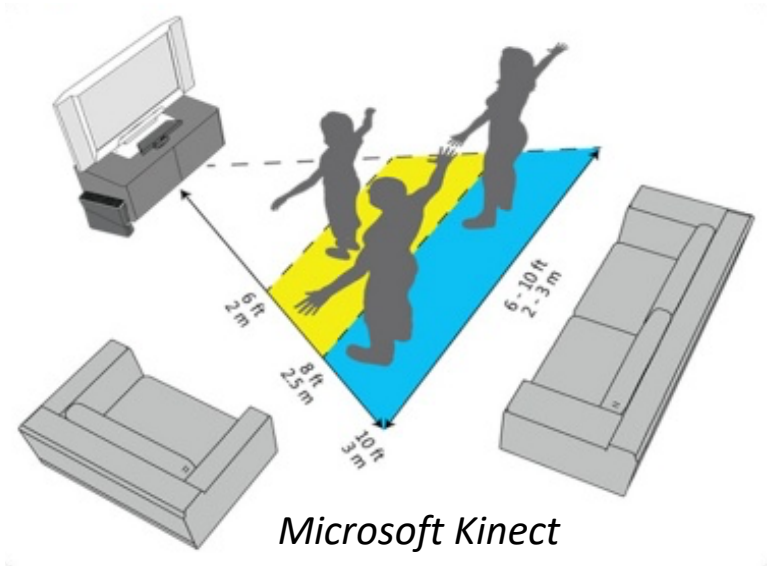
Google Project Tango!

Microsoft Hololens

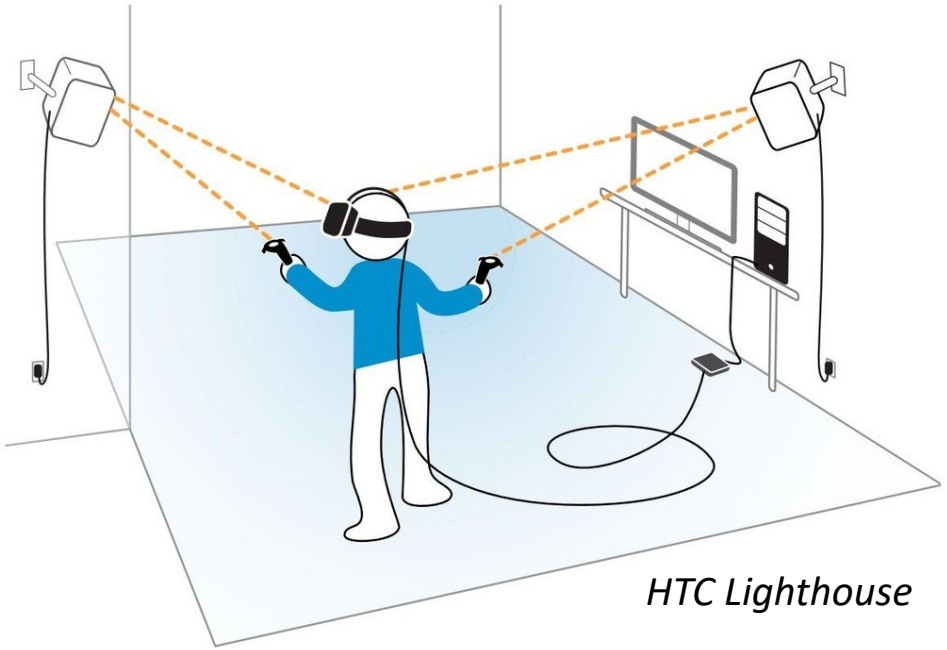


Inside-Out / Outside-In tracking

Outside-In : capteurs localisés à des positions fixes
(zone de suivi limitée)



Microsoft Kinect



HTC Lighthouse

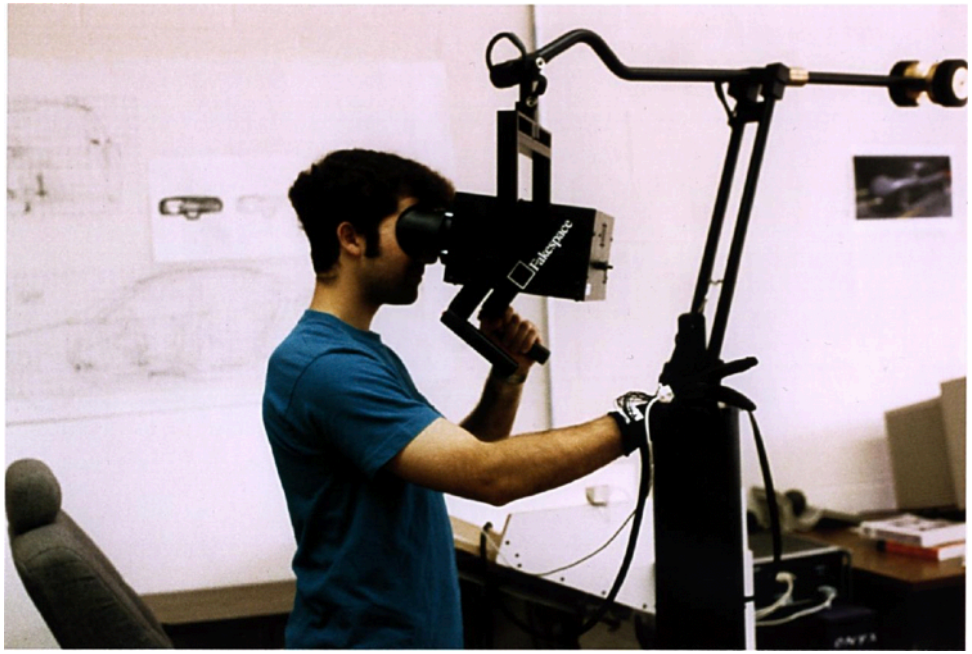
Systèmes o*utside-in*

- Mécaniques
- Ultra-sons
- Magnétiques
- Optiques (basés marqueurs)
- GPS, wifi…

Systèmes mécaniques

Bras articulé avec mesure des rotations aux articulations par un potentiomètre

$\Longrightarrow$ position reconstruire par cinématique directe



fakespace BOOM!

microscribe

Systèmes mécaniques

Avantages

- ✓ latence très faible
- ✓ très précis

Inconvénients

- ✗ encombrant
- ✗ inconfortable

Immersion CyberForce

Gypsy 6

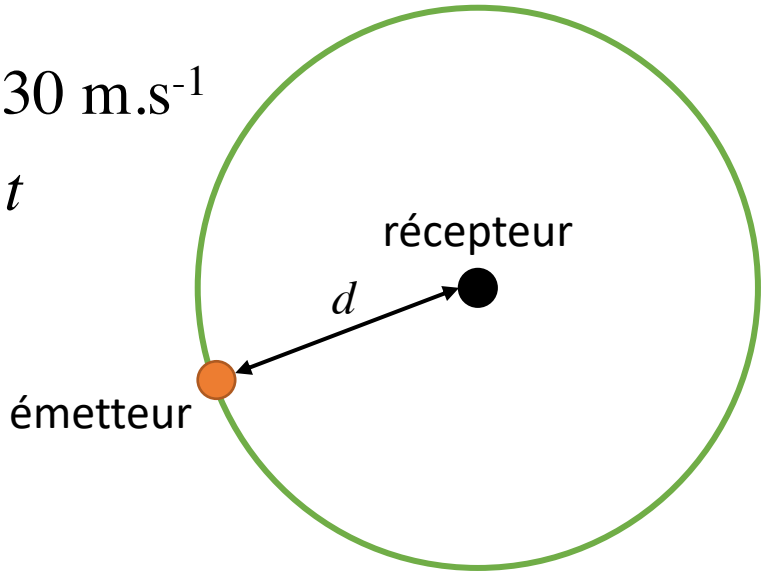
Systèmes ultra-sons

1 émetteur, 1 récepteur $\Longrightarrow$ *time of arrival / flight* (synchronisés)

Vitesse du son : $v = 330\ \mathrm{m.s^{-1}}$

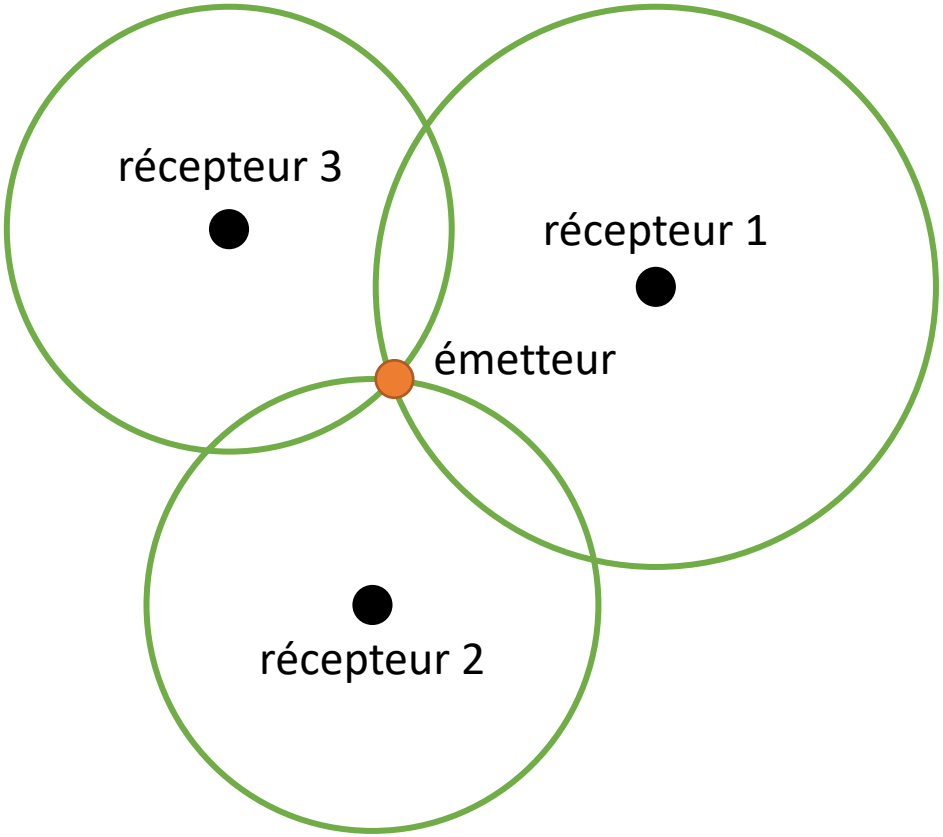
Mesure de la durée : t

$\Longrightarrow d = t.v$



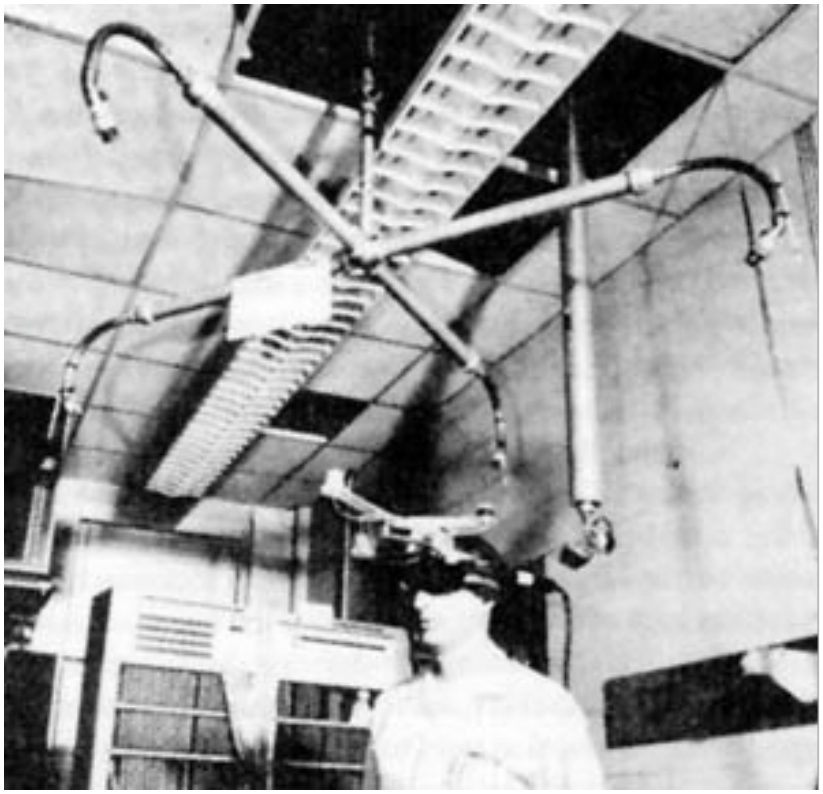
Systèmes ultra-sons

1 émetteur, 3 récepteurs (en 2D) $\Longrightarrow$ triangulation

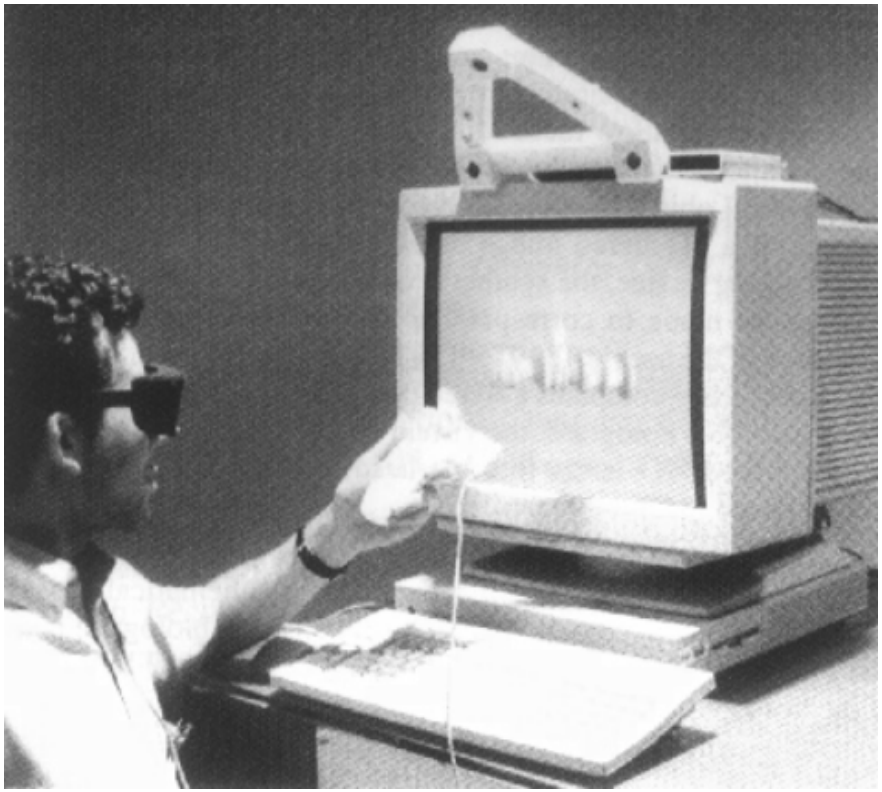


Systèmes ultra-sons

1 émetteur, 3 récepteurs $\Longrightarrow$ triangulation



Ivan Sutherland's "Ultimate Display" (1965)



Logitech 6DOF mouse

Systèmes ultra-sons

Avantages

- ✓ léger, peu encombrant
- ✓ très peu coûteux

Inconvénients

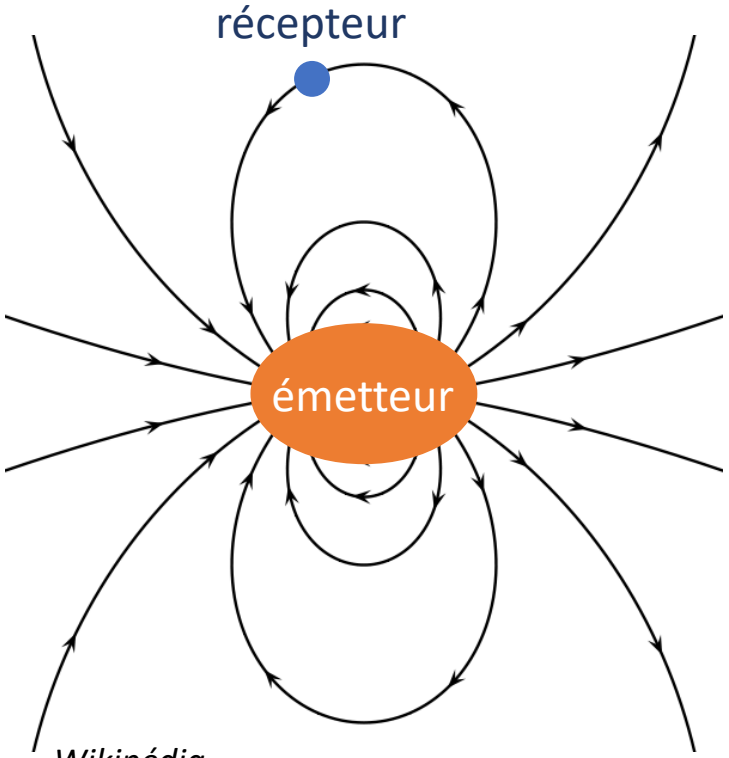
- ✗ contraintes de « visibilité »
- ✗ interférences acoustiques
- ✗ rafraichissement lent

Systèmes magnétiques

1 émetteur d'un champ électromagnétique (3 bobines)

1 récepteur mesurant la tension électrique induite selon 3 axes (magnétomètre)

- intensité dépend de la distance à l'émetteur ⟹ position
- mesures selon 3 directions orthogonales ⟹ orientation



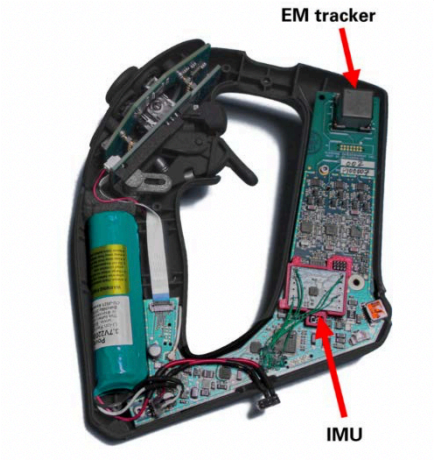
Wikipédia

Systèmes magnétiques

Exemples

MotionStar (2003)

Flock of Birds (1992)



Sixense STEM (2015)

Razer Hydra (2011)

Systèmes magnétiques

Avantages

- ✓ peu encombrant
- ✓ assez peu coûteux
- ✓ pas d'occlusions

Inconvénients

- ✗ distorsions dues aux objets métalliques
- ✗ champ d'action réduit (10-15m^2)
- ✗ précision de l'ordre du degré / mm

Systèmes optiques

Une ou plusieurs caméras

Un ou plusieurs marqueurs

actif passifs
- Marqueur coloré
- Motif binaire (QR code)
- Billes rétro-réfléchissantes
- LED (infra-rouge)

Oculus Rift CV1

AR Toolkit (1999)

Systèmes optiques

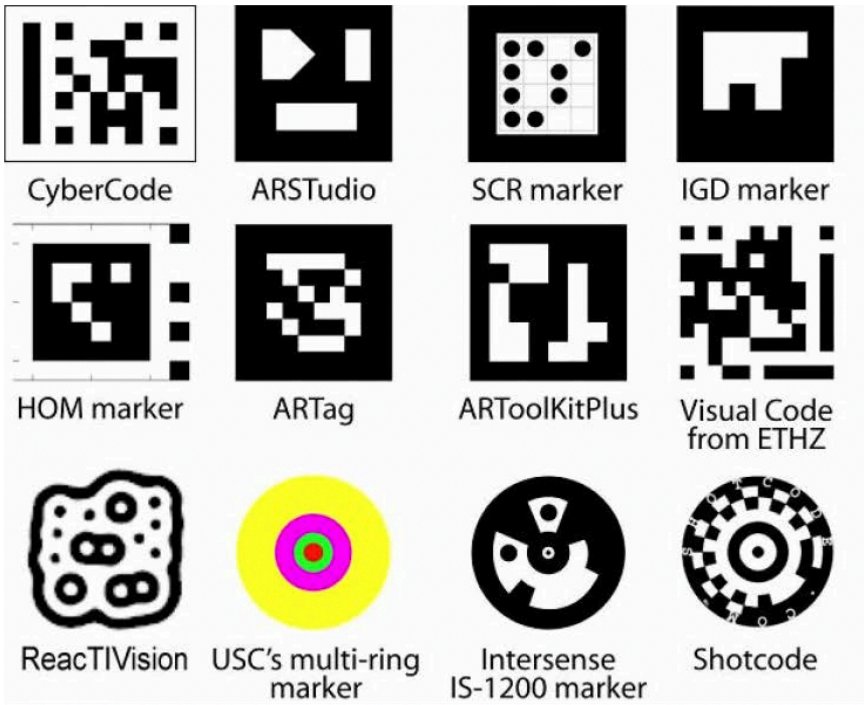
Caméra + marqueur(s)

détecter les 4 coins d'un marqueur
$\Longrightarrow$ **estimation de la pose**

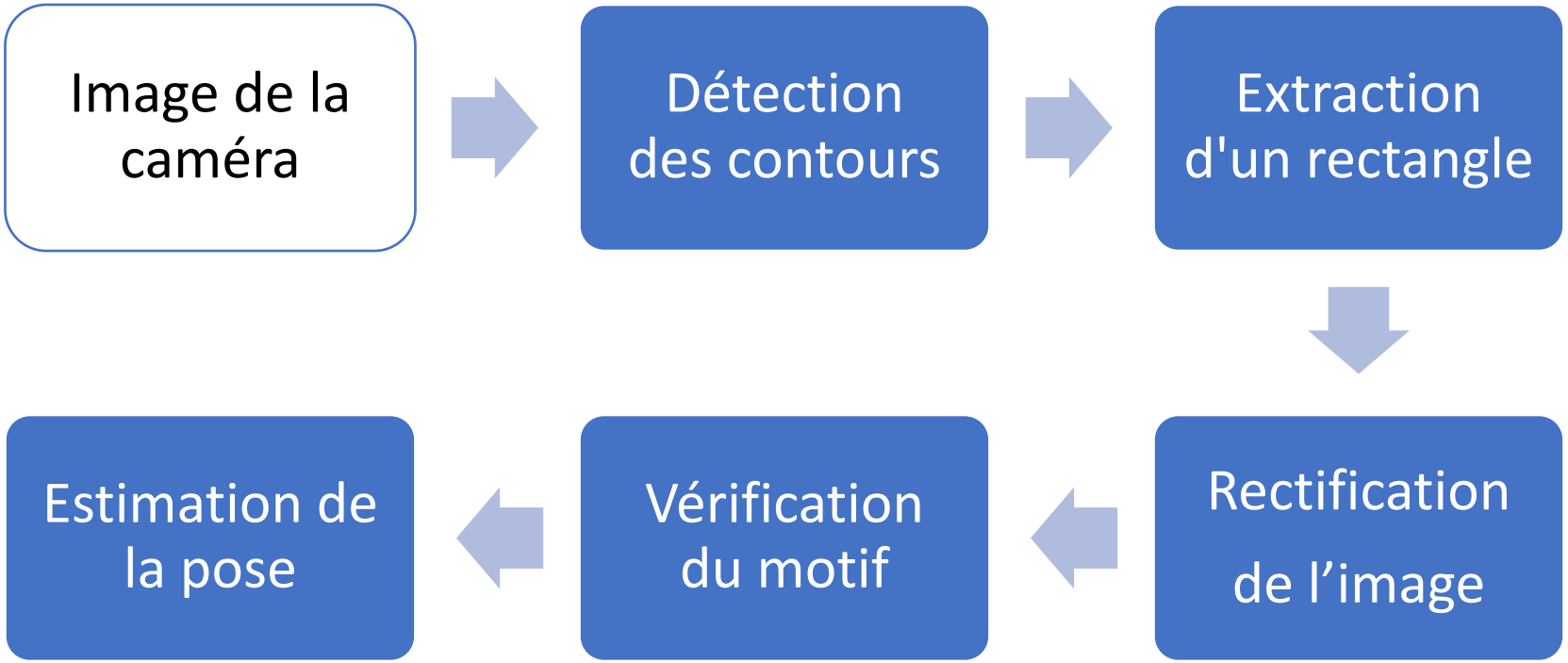
AR Toolkit (1999)

Techniques standards
de **vision par ordinateur**

Nombreuses **bibliothèques**
commerciales et open-source
(par exemple *ArUco* dans *OpenCV*)

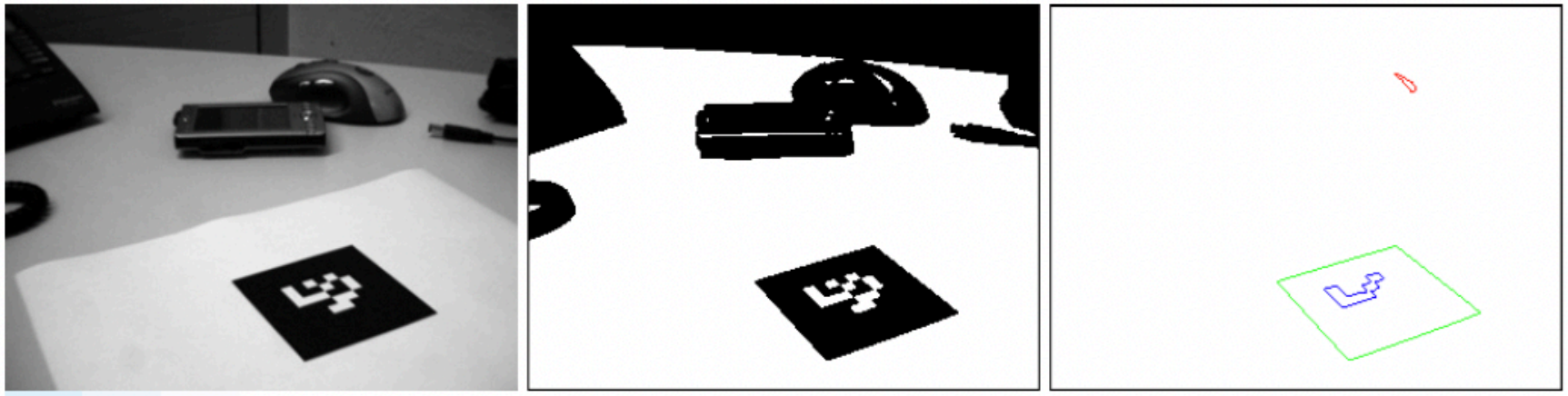


Chaîne de traitement



Extraction du marqueur

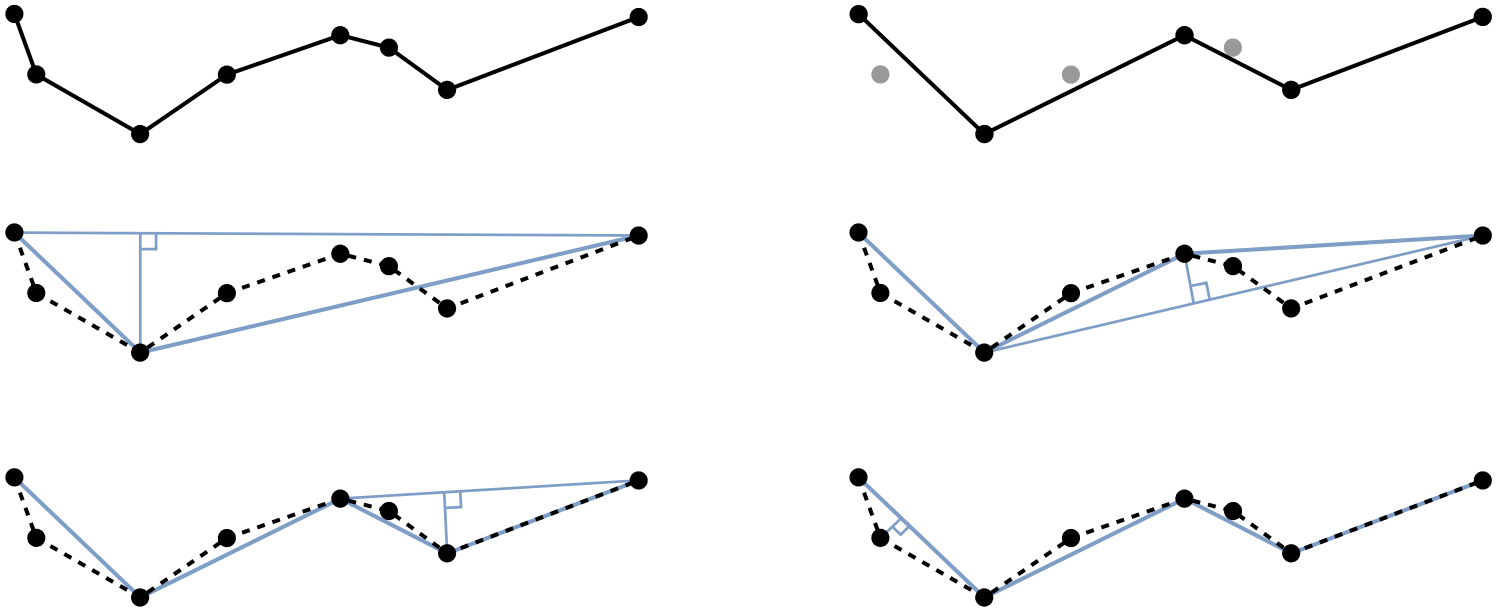
1. Seuillage adaptatif
2. Détection et chaînage des contours
3. Filtrage : simplification de polygones
   $\Longrightarrow$ polygones convexes à 4 côtés conservés



Simplification du polygone

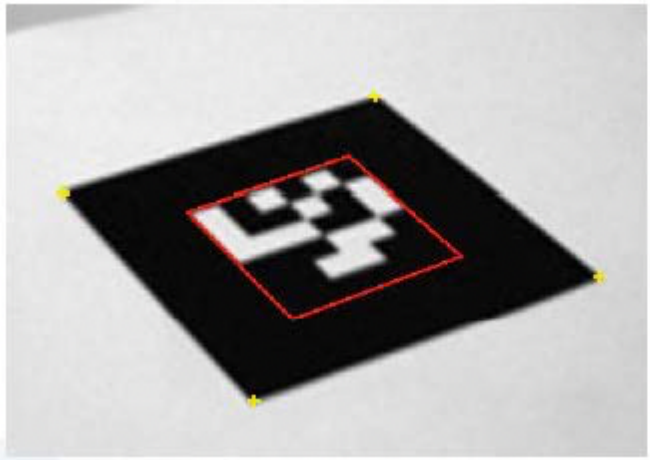
Algorithme de Douglas-Peucker

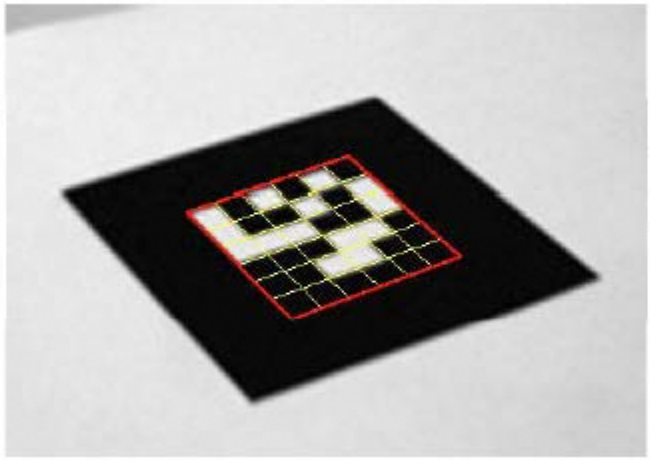
simplification récursive basée sur la distance au point le plus éloigné du segment courant



Vérification du motif

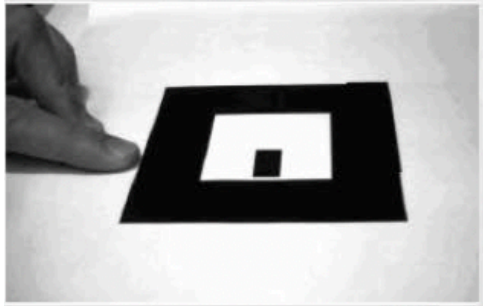
1. Rectification de l'image à partir des coins
2. Extraction du motif
   (par échantillonnage / binarisation adaptative)
3. Identification du motif (avec correction des erreurs)
4. Raffinement de la position des coins (sous pixellique)
5. Estimation de la pose complète (position + rotation)



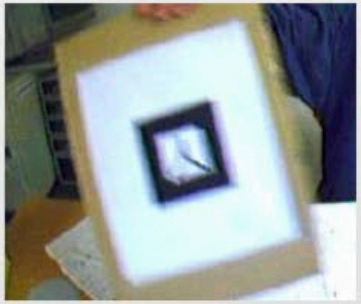


1	0	1	0	1	0
1	0	0	1	0	1
1	1	1	0	0	1
0	0	0	1	1	0
0	0	1	1	0	0
0	0	0	0	0	0

Challenges

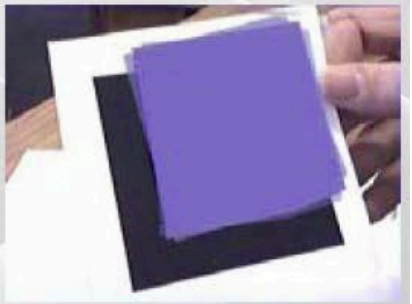


Occlusion
(image by M. Fiala)

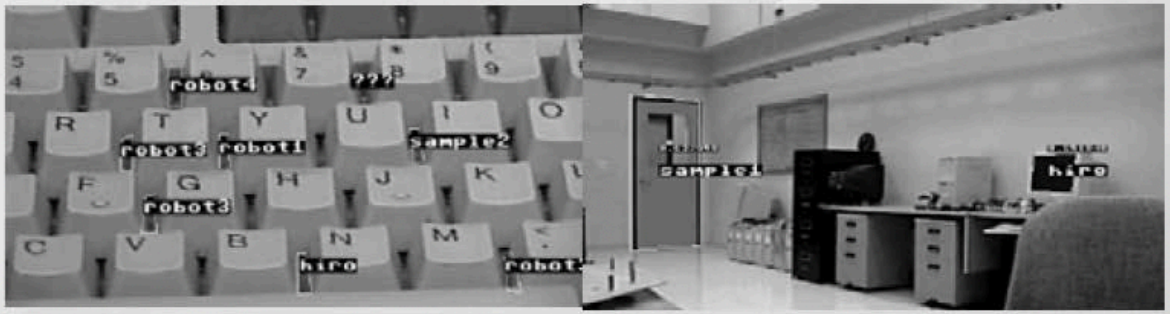


Unfocused camera, motion blur

Dark/unevenly lit scene, vignetting



Jittering
(Photoshop illustration)



False positives and inter-marker confusion
(image by M. Fiala)

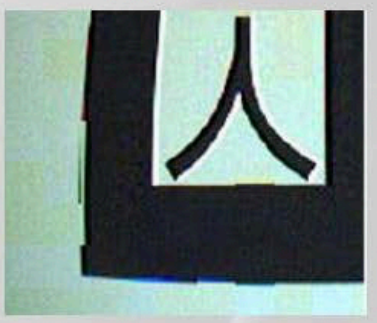
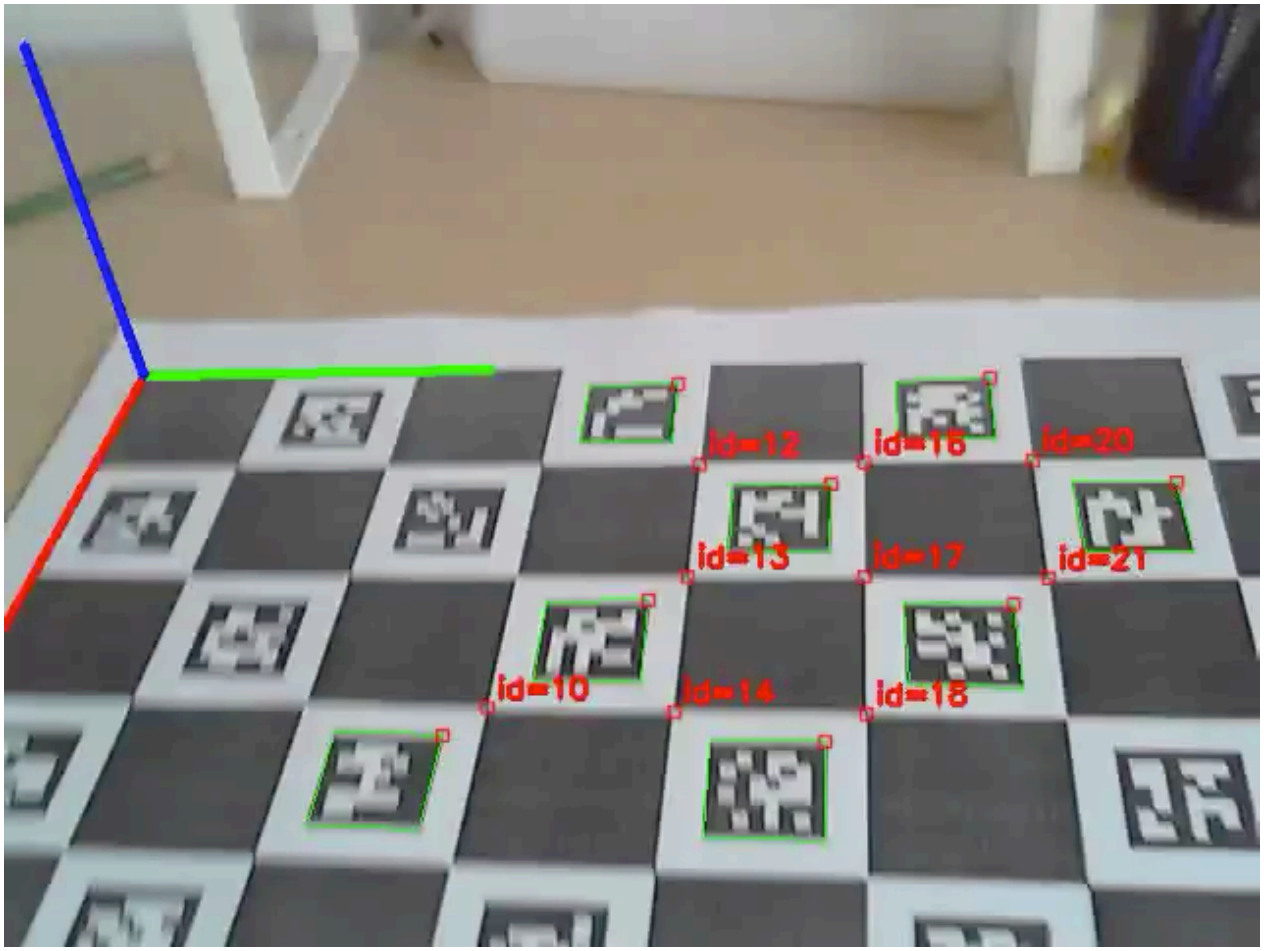


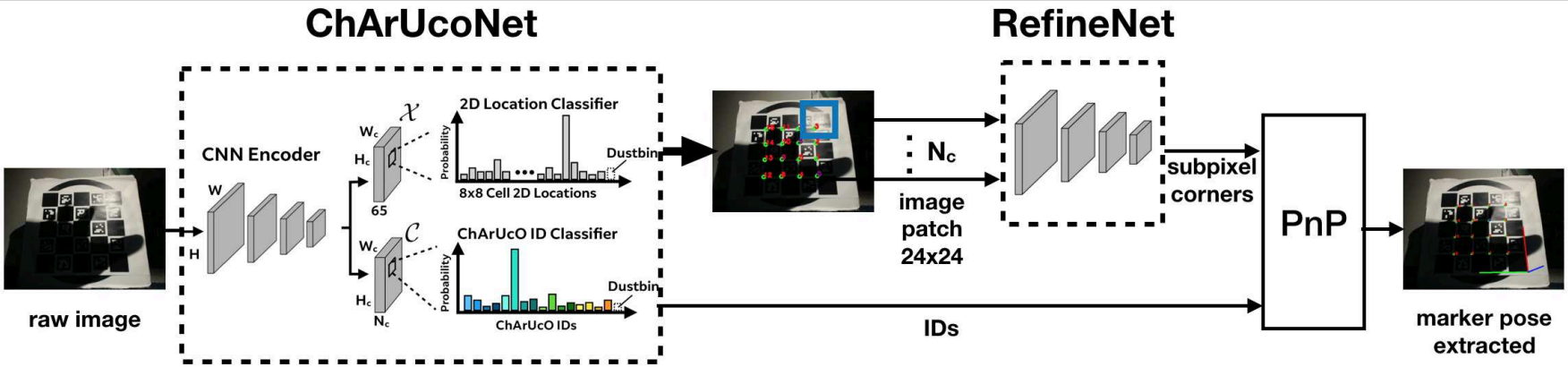
Image noise
(e.g. poor lens, block coding / compression, neon tube)

Marqueurs binaires *(fiducial)*

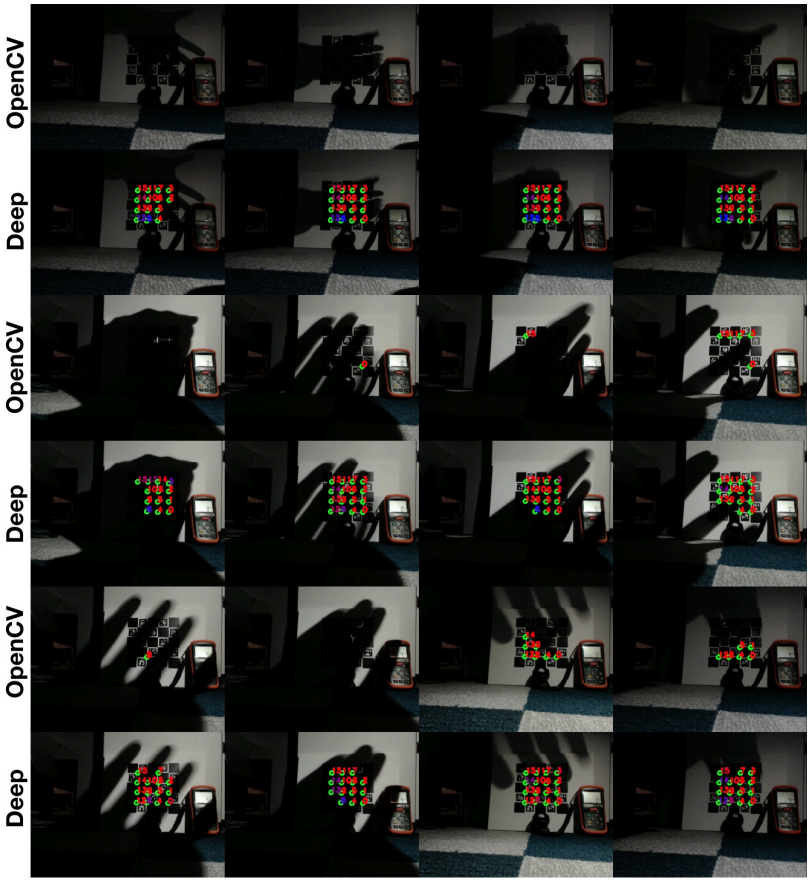


https://www.youtube.com/watch?v=Nj44m_N_9FY

Avec du *Deep learning* [Hu et al. 2019]



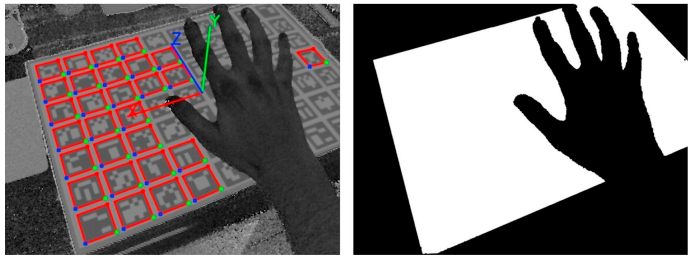
Avec du *Deep learning* [Hu et al. 2019]



Robustesse à l'éclairage

Robustesse au flou de mouvement

Caméra + marqueur(s)



Avantages

- ✓ très peu coûteux
- ✓ assez robuste avec plusieurs marqueurs

Inconvénients

[Romero-Ramirez et al. 2019]

- ✗ trop loin : marqueur trop petit
  trop près : marqueur incomplet
  $\Longrightarrow$ marqueurs fractales
- ✗ marqueurs peu appréciés par les utilisateurs
  $\Longrightarrow$ image « naturelle »
- ✗ variation des conditions lumineuses
  $\Longrightarrow$ lumière infrarouge

Caméra + image de référence

https://www.youtube.com/watch?v=R5rvRa0FqTQ

Marqueurs actifs

LED infrarouges,
ID encodé par leur fréquence
$\Longrightarrow$ caméra haute vitesse

Alimentation nécessaire
(batteries ou câbles)
$\Longrightarrow$ mouvements ou durée de capture restreinte

Coût d'équipement et de maintenance important
...mais grand volume de suivi

www.cs.utexas.edu

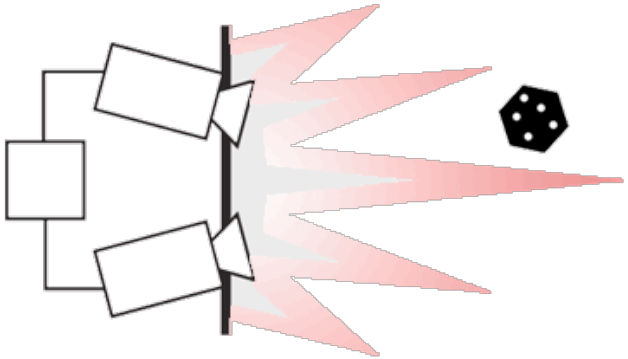
Marqueurs passifs

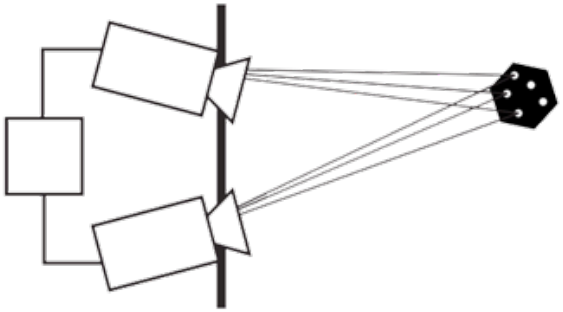
Sphères rétro-réfléchissantes

⟹ marqueurs ambigües

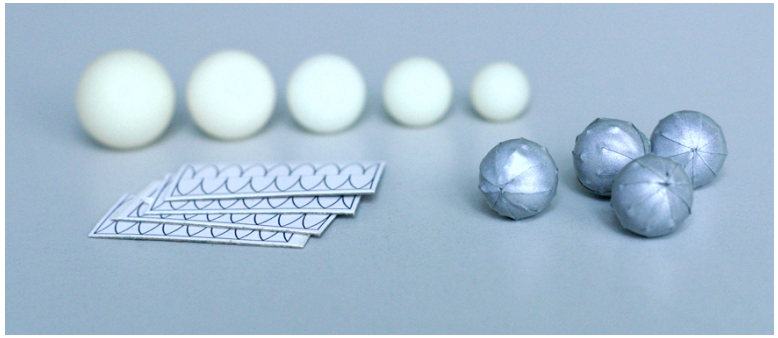
www.ps-tech.com

Lampes et **caméras** infrarouges





Coût d'équipement important
(plus le volume à suivre est grand)

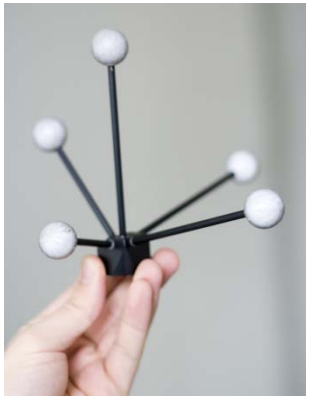


Marqueurs passifs

http://optitrack.com/products/prime-41/

Chaîne de traitement

1. **Calibration** des caméras
2. **Détection** des blobs
3. **Mise en correspondance** entre les vues
4. **Reconstruction** des positions 3D
5. **Ajustement** d'une constellation rigide
6. **Estimation de la pose** de a constellation (position et orientation)



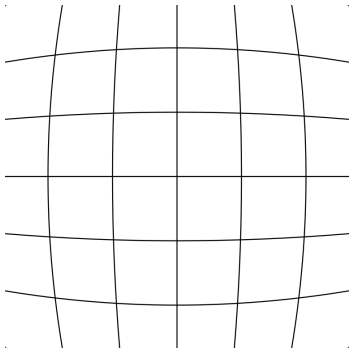
Calibration des caméras

a) Paramètres intrinsèques

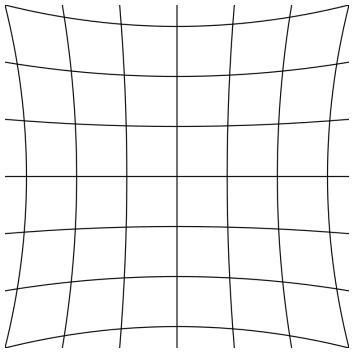
- distance focale (f_x, f_y) en mm
  avec $f_y = a\, f_x$ où a est l'aspect ratio des pixels
- centre optique (c_x, c_y) en pixels

$$\mathbf{K} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}$$

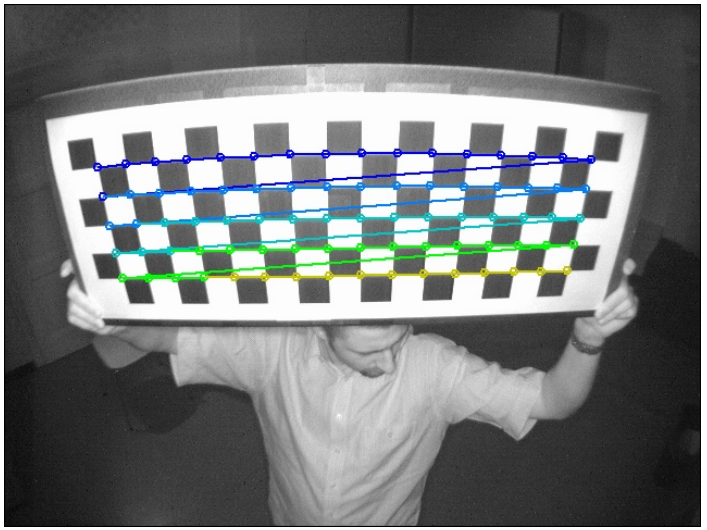
- distorsions radiales et tangentielles (négligeables)
  $\Longrightarrow$ déformation inverse de l'image pour les corriger



Distorsion en barillet



Distorsion en coussinet



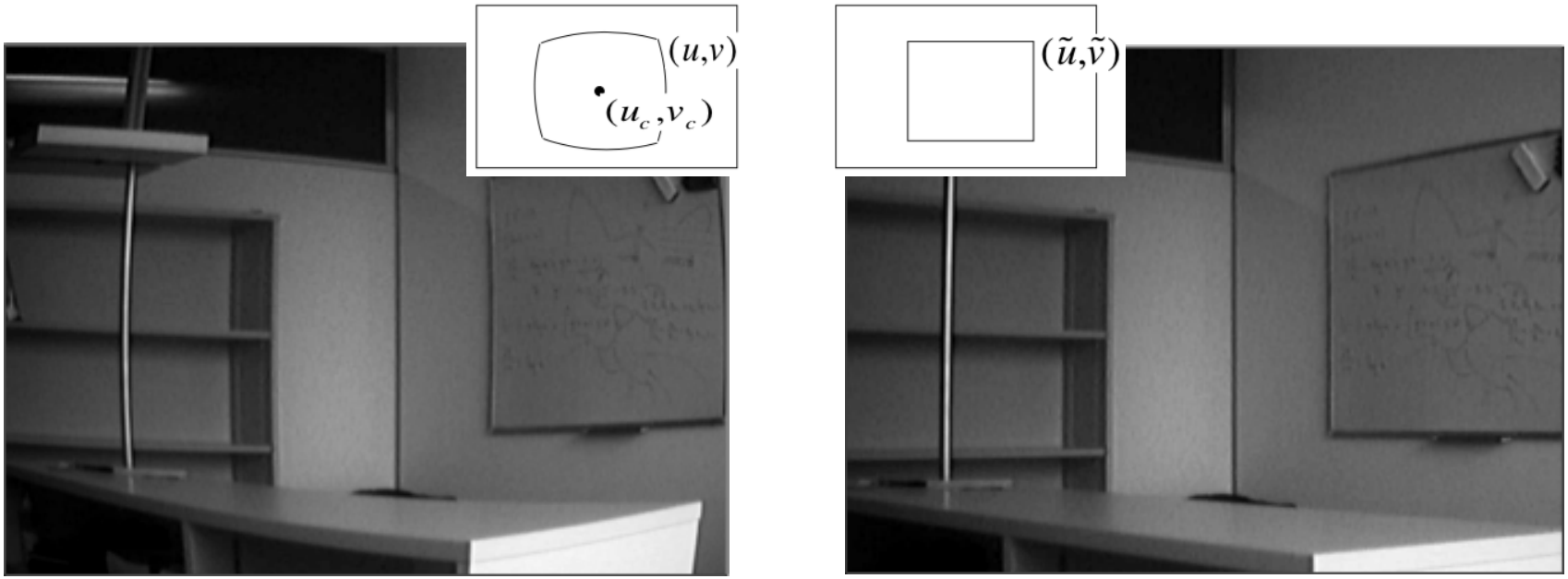
Correction des distorsions

[Brown & Conrady 1966]

Modélisé par : $L(r) = 1 + \kappa_1 r + \kappa_2 r^2 + \kappa_3 r^3 + \cdots$

avec $r^2 = (u - u_c)^2 + (v - v_c)^2$

Correction : $\begin{cases} \tilde{u} & = u_c + L(r)(u - u_c) \\ \tilde{v} & = v_c + L(r)(v - v_c) \end{cases}$



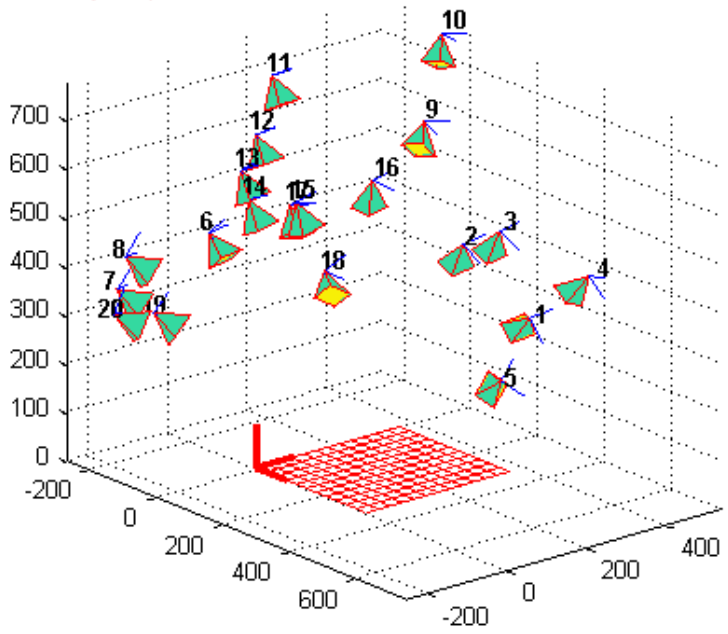
Calibration des caméras

b) Paramètres extrinsèques

- rotation $\mathbf{R}$
- translation $\mathbf{t}$

$\Longrightarrow$ matrice de projection ou caméra :

$$\mathbf{M} = \mathbf{K}\left(\mathbf{R}|\mathbf{t}\right)$$



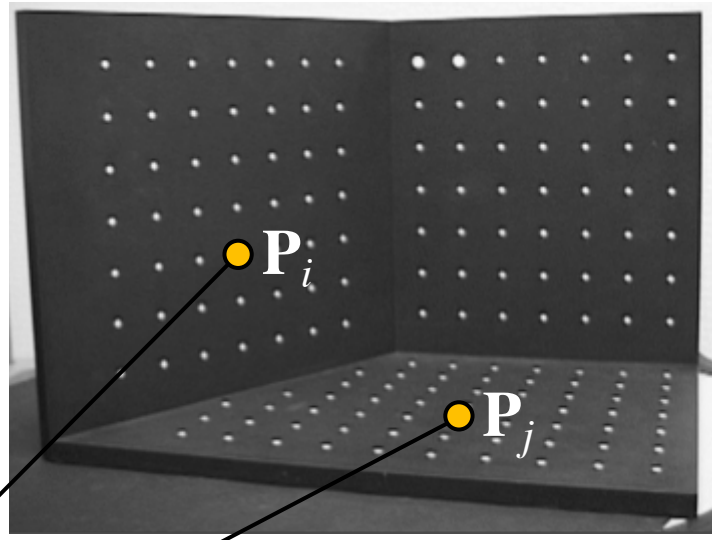
Camera Calibration Toolbox for Matlab

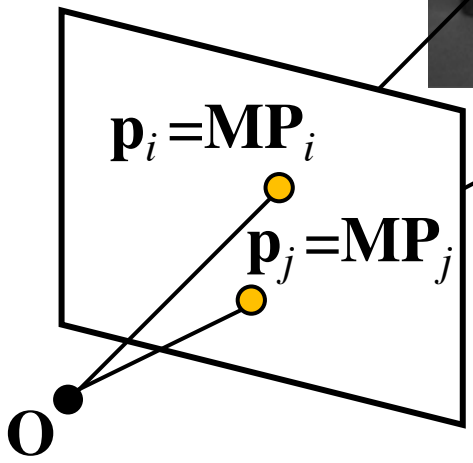
Calibration des caméras

[Tsai1987, Quan and Lan 1999]

$$\mathbf{M} = \begin{pmatrix} M_{11} & M_{12} & M_{13} & M_{14} \\ M_{21} & M_{22} & M_{23} & M_{24} \\ M_{31} & M_{32} & M_{33} & M_{34} \end{pmatrix}$$

$$\mathbf{p} = \begin{pmatrix} u \\ v \\ w \end{pmatrix} \quad \mathbf{P} = \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$





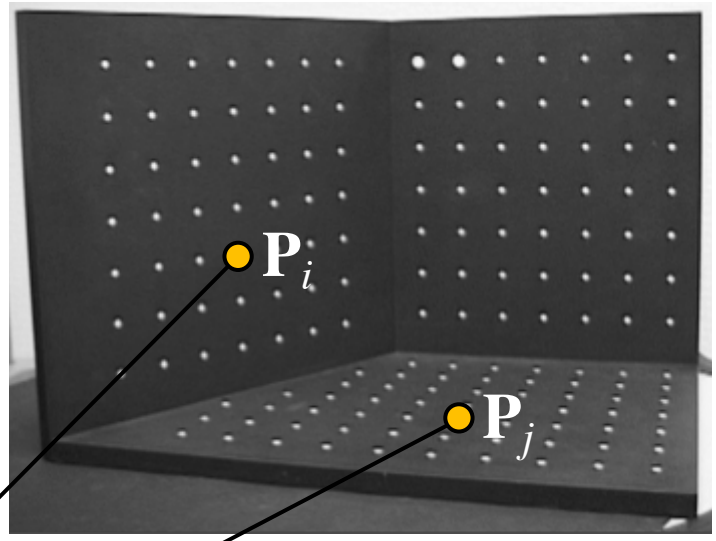
Direct Linear Transform (DLT)

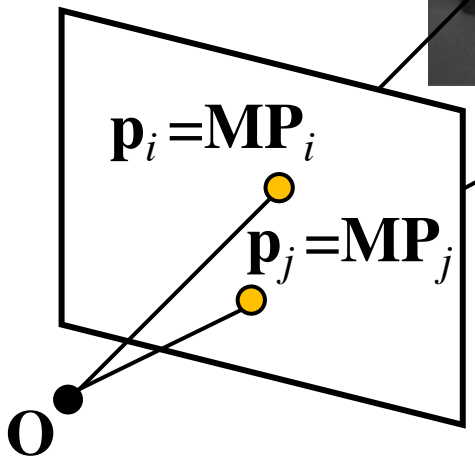
[Sutherland, 1974]

$$\mathbf{p}_i = \mathbf{M}\mathbf{P}_i$$

$$\Rightarrow \begin{cases} u_i = \dfrac{M_{11}X_i + M_{12}Y_i + M_{13}Z_i + M_{14}}{M_{31}X_i + M_{32}Y_i + M_{33}Z_i + M_{34}} \\ v_i = \dfrac{M_{21}X_i + M_{22}Y_i + M_{23}Z_i + M_{24}}{M_{31}X_i + M_{32}Y_i + M_{33}Z_i + M_{34}} \end{cases}$$

[Tsai1987, Quan and Lan 1999]





Direct Linear Transform (DLT)

[Sutherland, 1974]

$$\mathbf{p}_i = \mathbf{M}\mathbf{P}_i$$

$$\Longrightarrow \begin{pmatrix} X_i & Y_i & Z_i & 1 & 0 & 0 & 0 & 0 & -X_i u_i & -Y_i u_i & -Z_i u_i & -u_i \\ 0 & 0 & 0 & 0 & X_i & Y_i & Z_i & 1 & -X_i v_i & -Y_i v_i & -Z_i v_i & -v_i \end{pmatrix} \begin{pmatrix} M_{11} \\ \vdots \\ M_{34} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

En concaténant les équations pour tous les points, on obtient le système linéaire :

$$\mathbf{B}_{m\times n}\,\mathbf{m}_n = \mathbf{0}_n$$

11 inconnus $\Longrightarrow$ 6 correspondances 2D-3D nécessaires

…en pratique, bien plus pour être robuste au bruit !

Direct Linear Transform (DLT)

[Sutherland, 1974]

$\mathbf{B}_{m\times n}\,\mathbf{m}_n = \mathbf{0}_n \Longrightarrow$ système d'équations linéaires homogènes rectangulaire ($m > n$)

- Le vecteur nul est solution(triviale)
- Pour trouver une solution non-nulle, minimisation au sens des moindres carrés :

$$\min_{||\mathbf{m}||^2=1} ||\mathbf{Bm}||^2$$

Algèbre linéaire $\Longrightarrow \mathbf{B}^\mathsf{T}\mathbf{B}\ \mathbf{m} = \lambda\ \mathbf{m}$

$\Longrightarrow \mathbf{m}$ vecteur propre de $\mathbf{B}^\mathsf{T}\mathbf{B}$

$\|\mathbf{m}\|^2=1 \Longrightarrow$ associé à la plus petite valeur propre

Direct Linear Transform (DLT)

[Sutherland, 1974]

Décomposition en valeurs singulières (SVD)

$$\mathbf{B} = \mathbf{U}_{m\times m}\,\mathbf{S}_{m\times n}\,\mathbf{V}^{\mathsf{T}}{}_{n\times n}$$

$\mathbf{U}$, $\mathbf{V}$ orthogonales

$$\mathbf{U}^{\mathsf{T}}\mathbf{U} = \mathbf{V}^{\mathsf{T}}\mathbf{V} = \mathbf{I}$$

$$\mathbf{S} = \begin{pmatrix} diag(\sigma_1, \sigma_2, \ldots, \sigma_r) & 0 \\ 0 & 0 \end{pmatrix}$$

avec $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r,\ r \leq n$

$$\begin{pmatrix} M_{11} \\ \vdots \\ M_{34} \end{pmatrix}$$ = dernière colonne de $\mathbf{V} \Longrightarrow \mathbf{M}$

Extraction de $\mathbf{K}$, $\mathbf{R}$ et $\mathbf{t}$ depuis $\mathbf{M}$

$$\mathbf{M} = \mathbf{K}(\mathbf{R}|\mathbf{t})$$

Si $\mathbf{K}$ inconnue, les 3 premières colonnes de $\mathbf{M}$ correspondent à $\mathbf{KR} = \mathbf{M_3}$

$$\mathbf{M_3}\mathbf{M_3}^\top = (\mathbf{KR})(\mathbf{KR})^\top = \mathbf{KRR}^\top\mathbf{K}^\top = \mathbf{KK}^\top$$

Comme $\mathbf{K}$ est triangulaire supérieure, obtenue par **factorisation de Cholesky**

$$\mathbf{K} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}$$

Extraction de $\mathbf{K}$, $\mathbf{R}$ et $\mathbf{t}$ depuis $\mathbf{M}$

$\mathbf{M} = \mathbf{K}(\mathbf{R}|\mathbf{t})$

Une fois $\mathbf{K}$ connue, $(\mathbf{R}|\mathbf{t}) \propto \mathbf{K}^{-1}\mathbf{M}$

Pas garantie que $\mathbf{R}$ soit bien une matrice de rotation

$\Longrightarrow$ ortho-normalisation :

- décomposition en valeurs singulières (SVD) :

$$\mathbf{K}^{-1}\mathbf{M} = \mathbf{U}\mathbf{S}\mathbf{V}^{\top}$$

- $\mathbf{R} = \mathbf{U}\mathbf{V}^{\top}$

Linear Perspective-n-Point (P*n*P)

Si $\mathbf{K}$ connu, préférer P*n*P :

$\Longrightarrow$ estimer $\mathbf{R}$ et $\mathbf{t}$ à partir de *n* correspondances 2D/3D

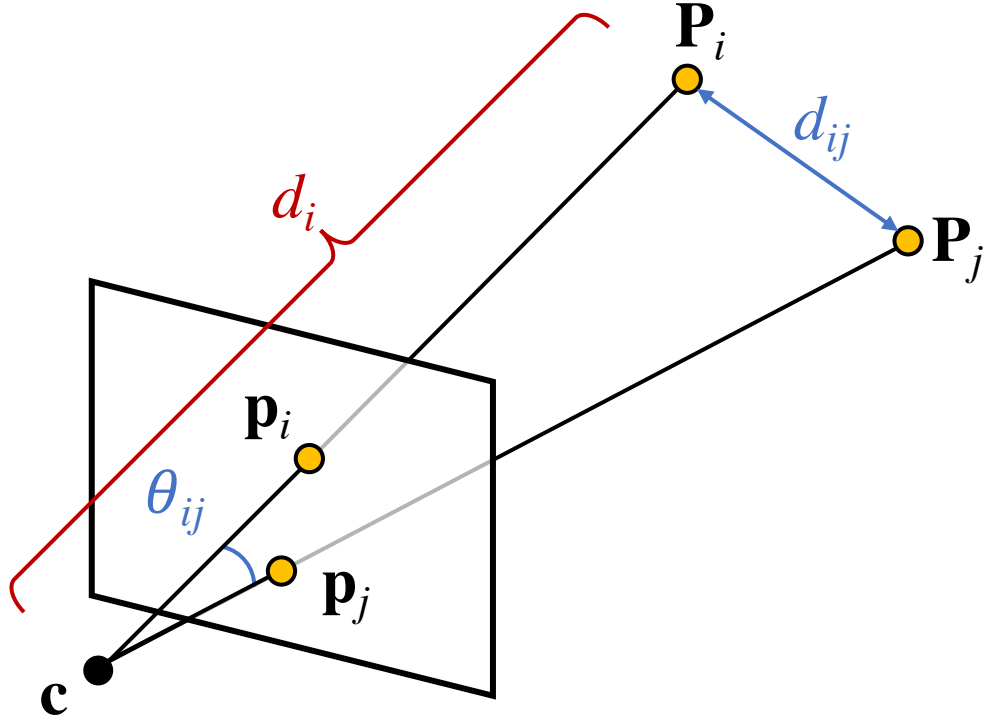
Linear Perspective-n-Point (P*n*P)

Relation : $d_{ij}^2 = d_i^2 + d_j^2 - 2 d_i d_j \cos\theta_{ij}$

avec : $d_{ij}^2 = ||\mathbf{P}_i - \mathbf{P}_j||^2$

$$\cos\theta_{ij} = \hat{\mathbf{p}}_i \cdot \hat{\mathbf{p}}_j$$

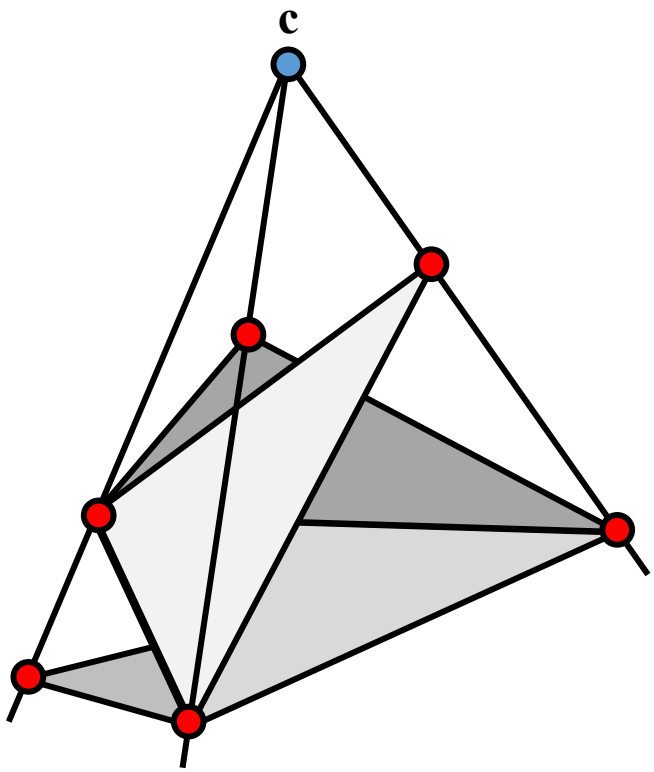
où : $$\hat{\mathbf{p}}_i = \frac{(\mathbf{p}_i - \mathbf{c})}{||(\mathbf{p}_i - \mathbf{c})||} = \frac{\mathbf{K}^{-1}\mathbf{p}_i}{||\mathbf{K}^{-1}\mathbf{p}_i||}$$



Linear Perspective-n-Point (P*n*P)

6 DOF $\Longrightarrow$ **3 correspondances** suffisantes
(2 DOF chacune)

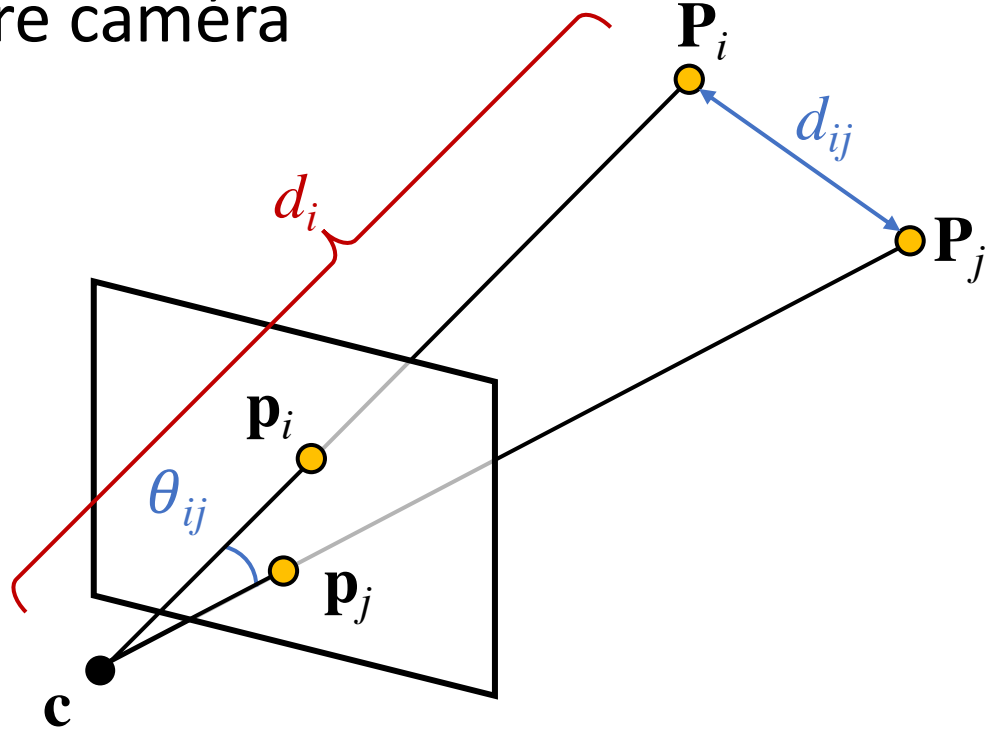
…mais ambiguïté (4 solutions)



[Wu and Hu, 2006]

Linear Perspective-3-Point (P3P)

1. Trouver les d_i satisfaisants ces relations
   (résoudre polynôme de degré 8 $\Longrightarrow$ 3 points nécessaires + 1 contrainte)
2. Calculer les positions $\mathbf{P}_i^{\mathbf{c}} = d_i \mathbf{K}^{-1}\mathbf{p}_i$
   de $\mathbf{P}_i$ dans le repère caméra
3. Estimer $\mathbf{R}$ et $\mathbf{t}$ par
   alignement 3D



Estimer $\mathbf{R}$ et $\mathbf{t}$ par alignement 3D

1. Calculer les **centroïdes** des ensembles de points :

$$\bar{\mathbf{P}} = \frac{1}{N}\sum_{i=0}^{N}\mathbf{P}_i \qquad \bar{\mathbf{P}}^{\mathbf{c}} = \frac{1}{N}\sum_{i=0}^{N}\mathbf{P}_i^{\mathbf{c}}$$

$$\Longrightarrow \mathbf{t} = \bar{\mathbf{P}}^{\mathbf{c}} - \bar{\mathbf{P}}$$

2. Trouver la **rotation** entre 2 ensembles de points :

$$\left\{\hat{\mathbf{P}}_i = \mathbf{P}_i - \bar{\mathbf{P}}\right\} \qquad \left\{\hat{\mathbf{P}}_i^{\mathbf{c}} = \mathbf{P}_i^{\mathbf{c}} - \bar{\mathbf{P}}^{\mathbf{c}}\right\}$$

maximisant l'alignement :

$$\mathbf{L} = \sum_i (\hat{\mathbf{P}}_i)^{\top}(\mathbf{R}\hat{\mathbf{P}}_i^{\mathbf{c}})$$

$\Longrightarrow \mathbf{R} = \mathbf{U}\mathbf{V}^{\top}$ où $\mathbf{L} = \mathbf{U}\mathbf{S}\mathbf{V}^{\top}$ (SVD)

Linear Perspective-n-Point (P*n*P)

Seulement 3 points $\Longrightarrow$ **imprécisions**

Utiliser *n* points

- algo. itératifs [Lu, Hager, and Mjolsness, 2000]
- formes closes [Moreno-Noguer, Lepetit, and P. Fua, 2007]
- + rejet des points aberrants (*outliers*)

Recherche la **pose sans passer par les distances**
[Ke and Roumeliotis, CVPR'17]

- Plus rapide, plus précis et plus robuste
- Disponible dans OpenCV > 3.3 :
  `solvePnP()` avec `SOLVEPNP_AP3P`

Minimiser erreur de reprojection

Moindres carrés non linéaires :

$$\min \sum_i ||\mathbf{K}(\mathbf{R}|\mathbf{t})\mathbf{P}_i - \mathbf{p}_i||^2$$

Approche plus flexible, sans limite sur le nombre de points, pouvant être très précise

…mais méthode d'optimisation **itérative**
(Gauss-Newton, Levenberg-Marquardt)

$\Longrightarrow$ Jacobienne de l'énergie et solution initiale requises
(e.g., solution linéaire)

$\Longrightarrow$ rendre robuste aux *outliers* (RANSAC)

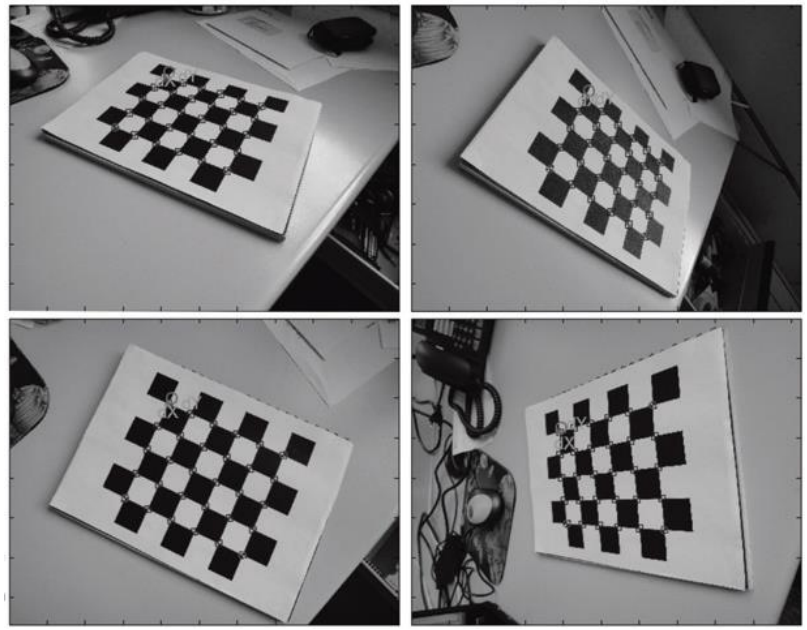
Calibration des caméras

Grille planaire $\Longrightarrow Z = 0$

$$\mathbf{p} = \begin{pmatrix} u \\ v \\ w \end{pmatrix} = \mathbf{K}(\mathbf{R}|\mathbf{t}) \begin{pmatrix} X \\ Y \\ 0 \\ 1 \end{pmatrix}$$

$$= \mathbf{K} \underbrace{\begin{pmatrix} r_{11} & r_{12} & t_1 \\ r_{21} & r_{22} & t_2 \\ r_{31} & r_{32} & t_3 \end{pmatrix}}_{\mathbf{H}\ \textbf{homographie}} \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix}$$

$$= \begin{pmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{pmatrix}$$



[Zhang 1999]

Homographie

8 degrés de liberté + facteur d'échelle s

$$\mathbf{p} = \begin{pmatrix} u \\ v \\ w \end{pmatrix} = s \begin{pmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix}$$

$$\Rightarrow \begin{pmatrix} \frac{u}{w} \\ \frac{v}{w} \end{pmatrix} = \begin{pmatrix} \frac{h_1 X + h_2 Y + h_3}{h_7 X + h_8 Y + 1} \\ \frac{h_4 X + h_5 Y + h_6}{h_7 X + h_8 Y + 1} \end{pmatrix}$$

$\Longrightarrow$ DLT ou P*n*P avec **4 points** pour estimer $\mathbf{H}$

Facteur d'échelle s

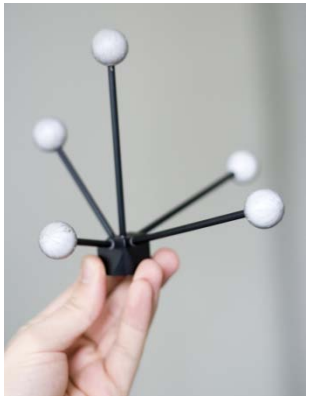
Rotation $\Longrightarrow$ 2 premières colonnes normalisées

$$\mathbf{K}\begin{pmatrix} r_{11} & r_{12} & t_1 \\ r_{21} & r_{22} & t_2 \\ r_{31} & r_{32} & t_3 \end{pmatrix} = s\begin{pmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & 1 \end{pmatrix}$$

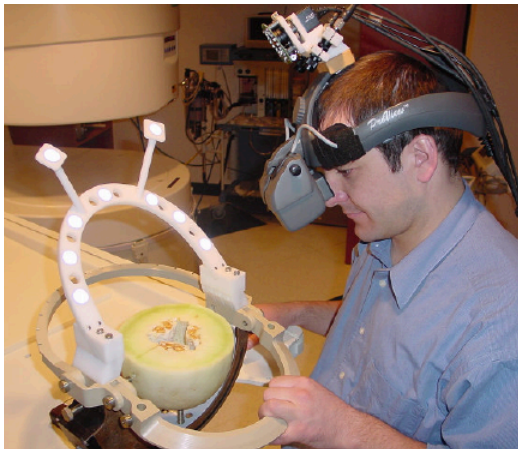
$$\Longrightarrow \; s = \frac{2}{\sqrt{h_1^2 + h_4^2 + h_7^2} + \sqrt{h_2^2 + h_5^2 + h_8^2}}$$

Chaîne de traitement

1. **Calibration** des caméras
2. **Détection** des blobs
3. **Mise en correspondance** entre les vues
4. **Reconstruction** des positions 3D
5. **Ajustement** d'une constellation rigide
6. **Estimation de la pose** de a constellation (position et orientation)



Détection des blobs



Seuillage pour détecter les candidats,
moyenne pondérée par luminance :

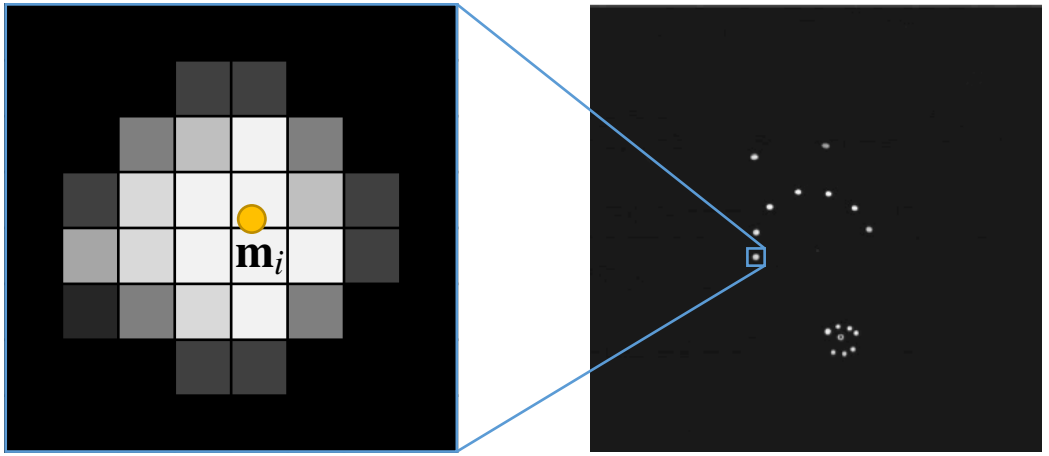
$$\mathbf{m}_i = \frac{\Sigma_{\mathbf{p}_j} I(\mathbf{p}_j)\mathbf{p}_j}{\Sigma_{\mathbf{p}_j} I(\mathbf{p}_j)}$$

précision de l'ordre du 10ème de pixel (voire 100ème)

Problème :

superpositions

(segmentation basée contours, *Hough circle transform, machine learning*)



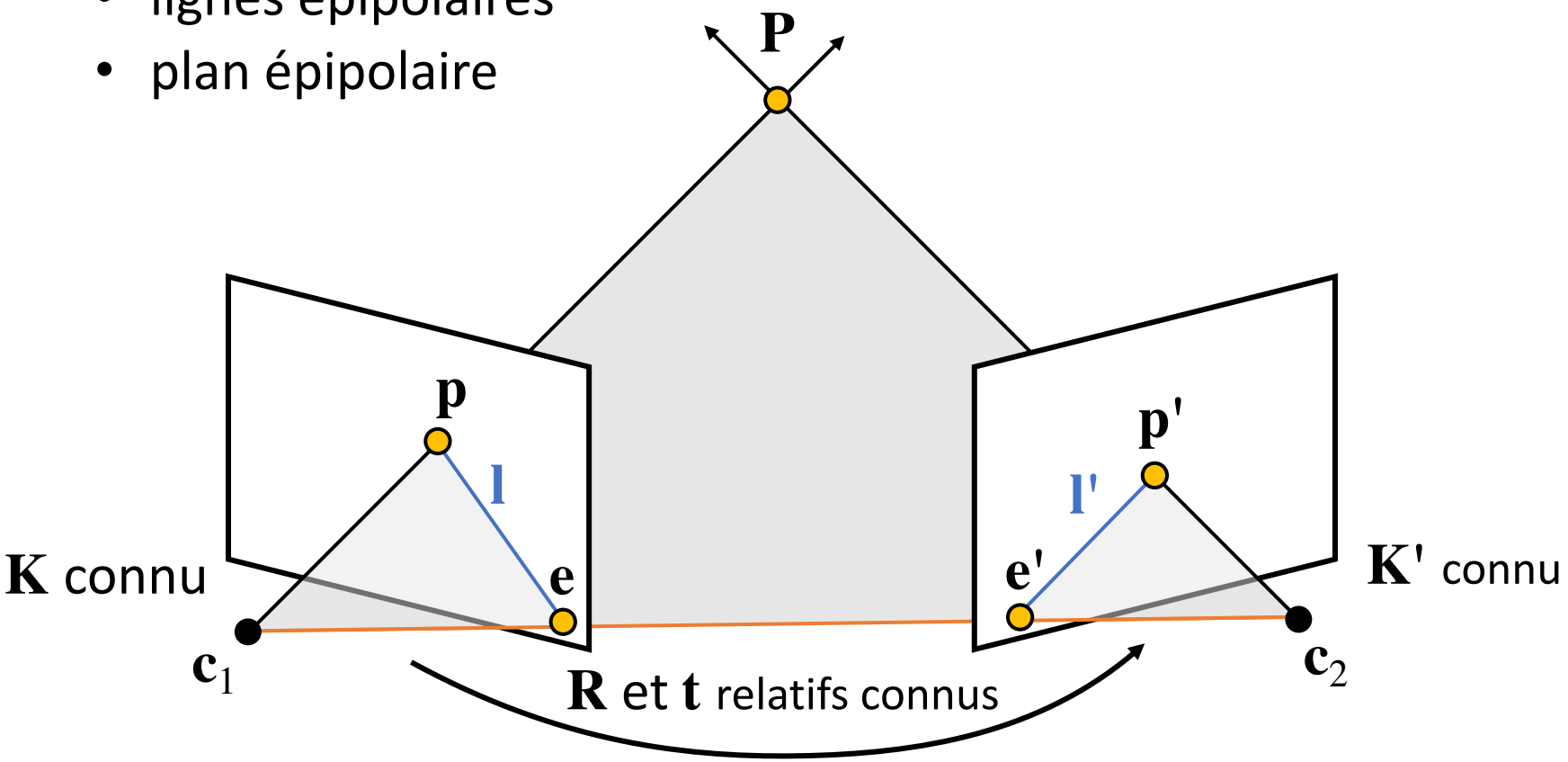
Mise en correspondance

Marqueurs actifs : ID encodé par la fréquence,
mais nécessite des caméras haute vitesse

Marqueurs passif : géométrie épipolaire

Géométrie épipolaire

- ligne de base
- épipoles **e** et **e'**
- lignes épipolaires
- plan épipolaire



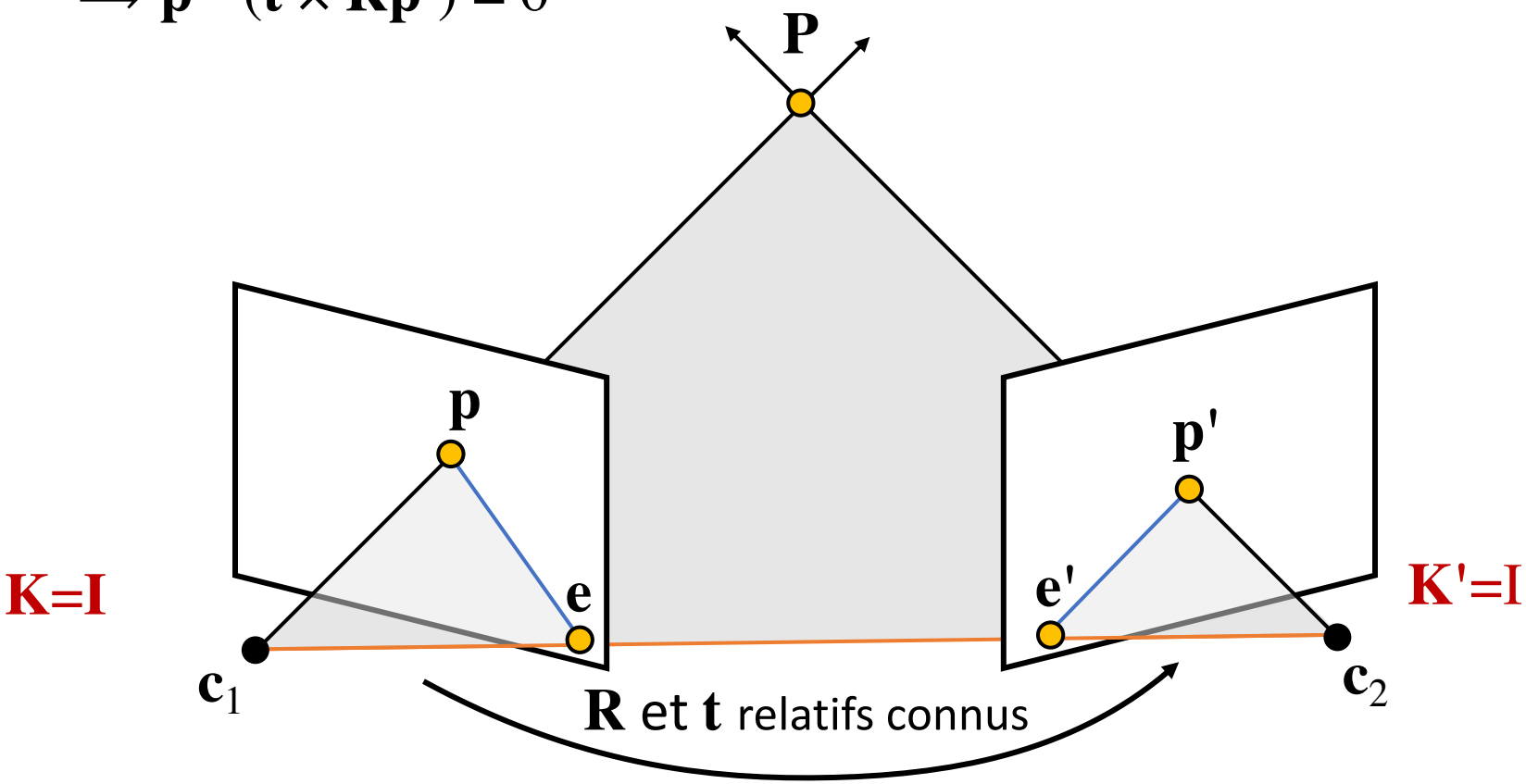
Contrainte épipolaire

caméras canoniques

Si $\mathbf{K} = \mathbf{K}' = \mathbf{I}$, $\mathbf{p}'$ dans le repère de la 1ère caméra vaut $\mathbf{Rp}' + \mathbf{t}$

$\Longrightarrow \mathbf{t} \times (\mathbf{Rp}' + \mathbf{t}) = \mathbf{t} \times \mathbf{Rp}'$ vecteur orthogonal au plan épipolaire

$\Longrightarrow \mathbf{p} \bullet (\mathbf{t} \times \mathbf{Rp}') = 0$



Produit scalaire/vectoriel et multiplication de matrices

$$\mathbf{a} \cdot \mathbf{b} = \begin{pmatrix} a_x & a_y & a_z \end{pmatrix} \begin{pmatrix} b_x \\ b_y \\ b_z \end{pmatrix} = \mathbf{a}^\top \mathbf{b}$$

$$\mathbf{a} \times \mathbf{b} = \begin{pmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{pmatrix} \mathbf{b} = [\mathbf{a}_\times]\mathbf{b}$$

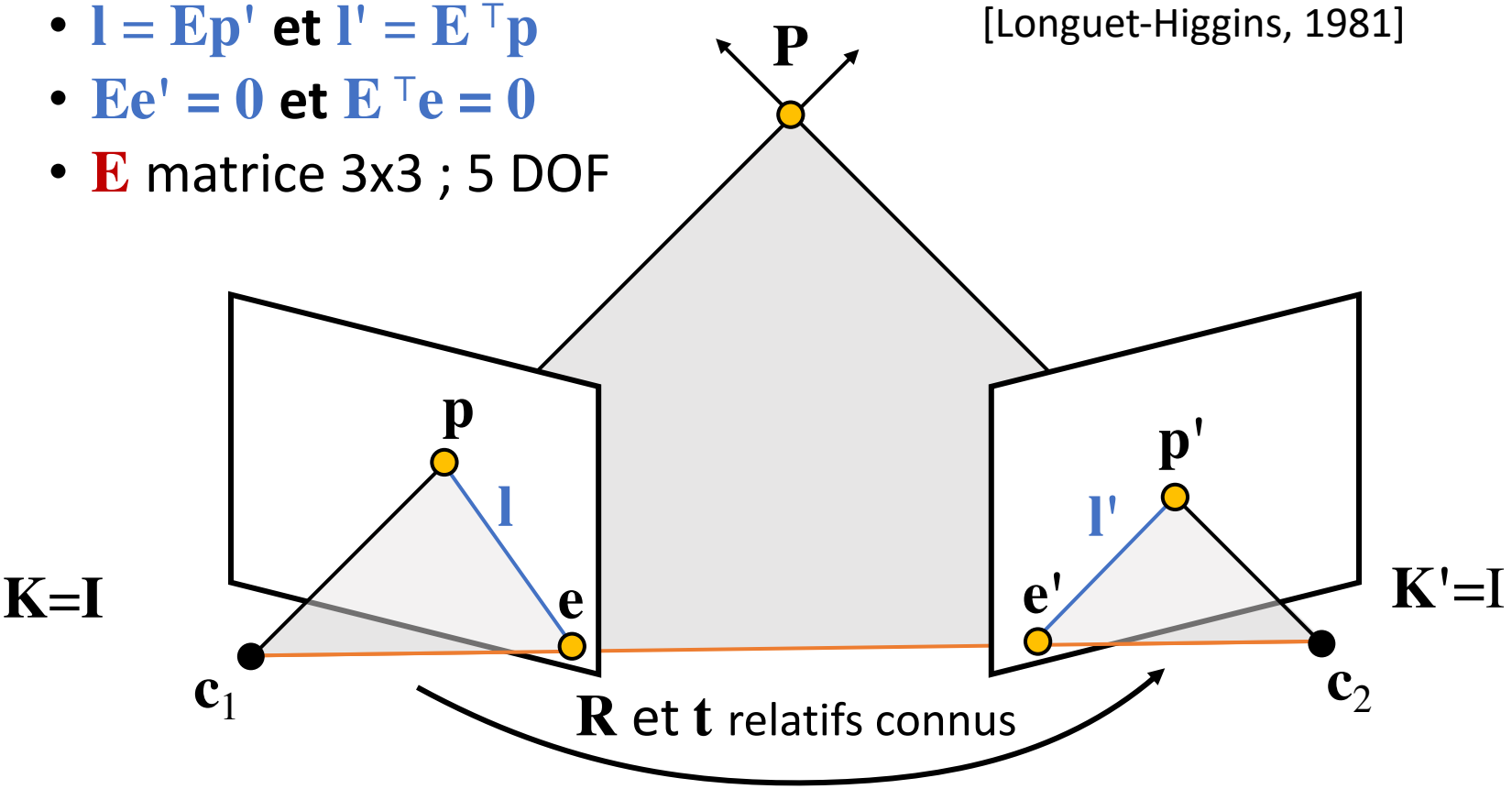
Contrainte épipolaire

$$\mathbf{p} \bullet (\mathbf{t} \times \mathbf{R}\mathbf{p}') = 0 \Leftrightarrow \mathbf{p}^{\top} \underbrace{[\mathbf{t}_{\times}]\,\mathbf{R}}_{= \mathbf{E}} \mathbf{p}' = 0$$

$= \mathbf{E}$: matrice essentielle

[Longuet-Higgins, 1981]

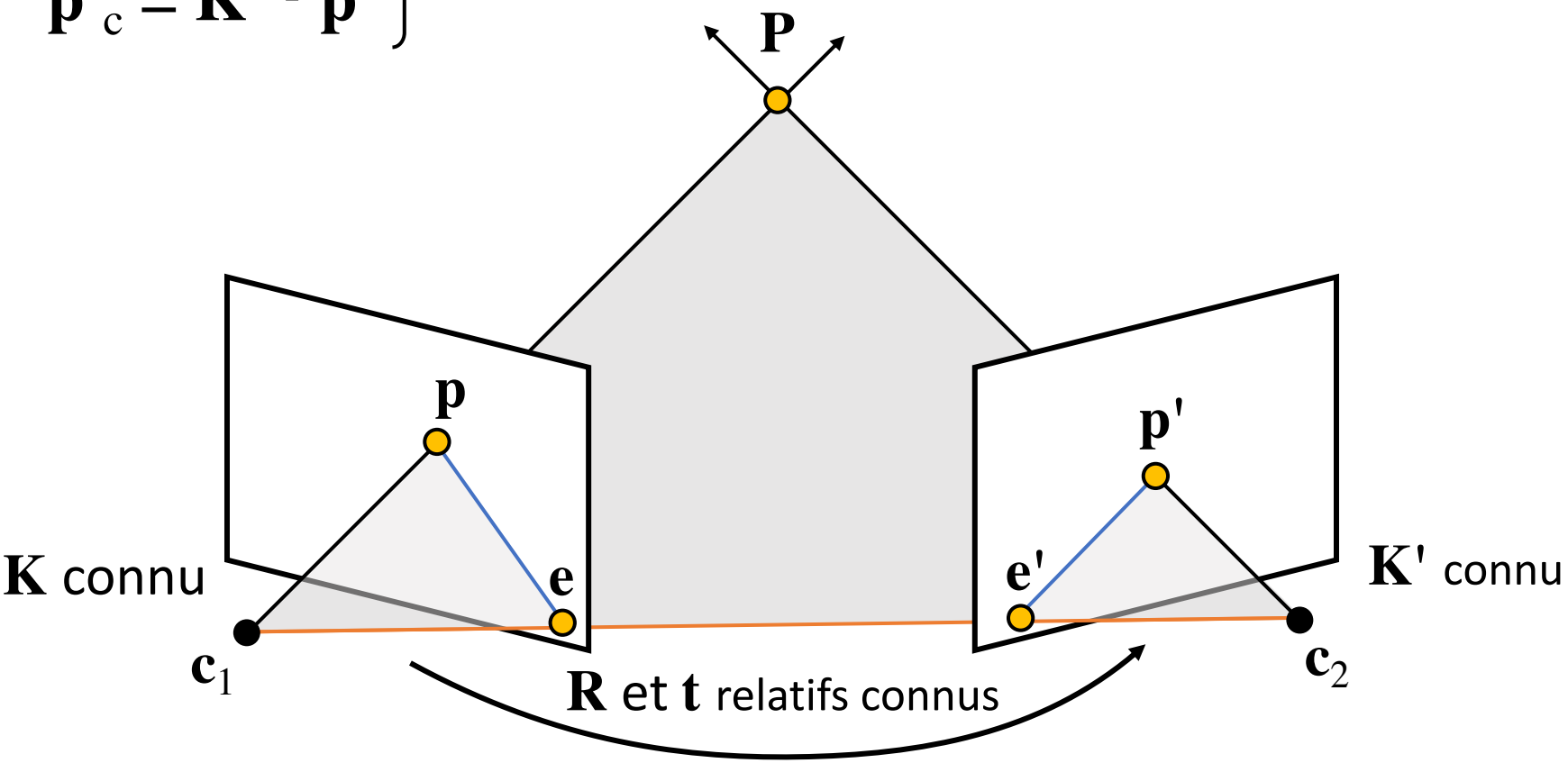
- $\mathbf{l} = \mathbf{E}\mathbf{p}'$ et $\mathbf{l}' = \mathbf{E}^{\top}\mathbf{p}$
- $\mathbf{E}\mathbf{e}' = 0$ et $\mathbf{E}^{\top}\mathbf{e} = 0$
- $\mathbf{E}$ matrice 3x3 ; 5 DOF



Contrainte épipolaire

$$\left.\begin{aligned} \mathbf{p}_c &= \mathbf{K}^{-1}\,\mathbf{p} \\ \mathbf{p}'_c &= \mathbf{K}'^{-1}\,\mathbf{p}' \end{aligned}\right\} \Longrightarrow \mathbf{p}_c{}^{\top}\,[\mathbf{t}_\times]\,\mathbf{R}\,\mathbf{p}'_c = 0$$

$$\Longrightarrow (\mathbf{K}^{-1}\,\mathbf{p})^{\top}\,[\mathbf{t}_\times]\,\mathbf{R}\,\mathbf{K}'^{-1}\,\mathbf{p}' = 0$$

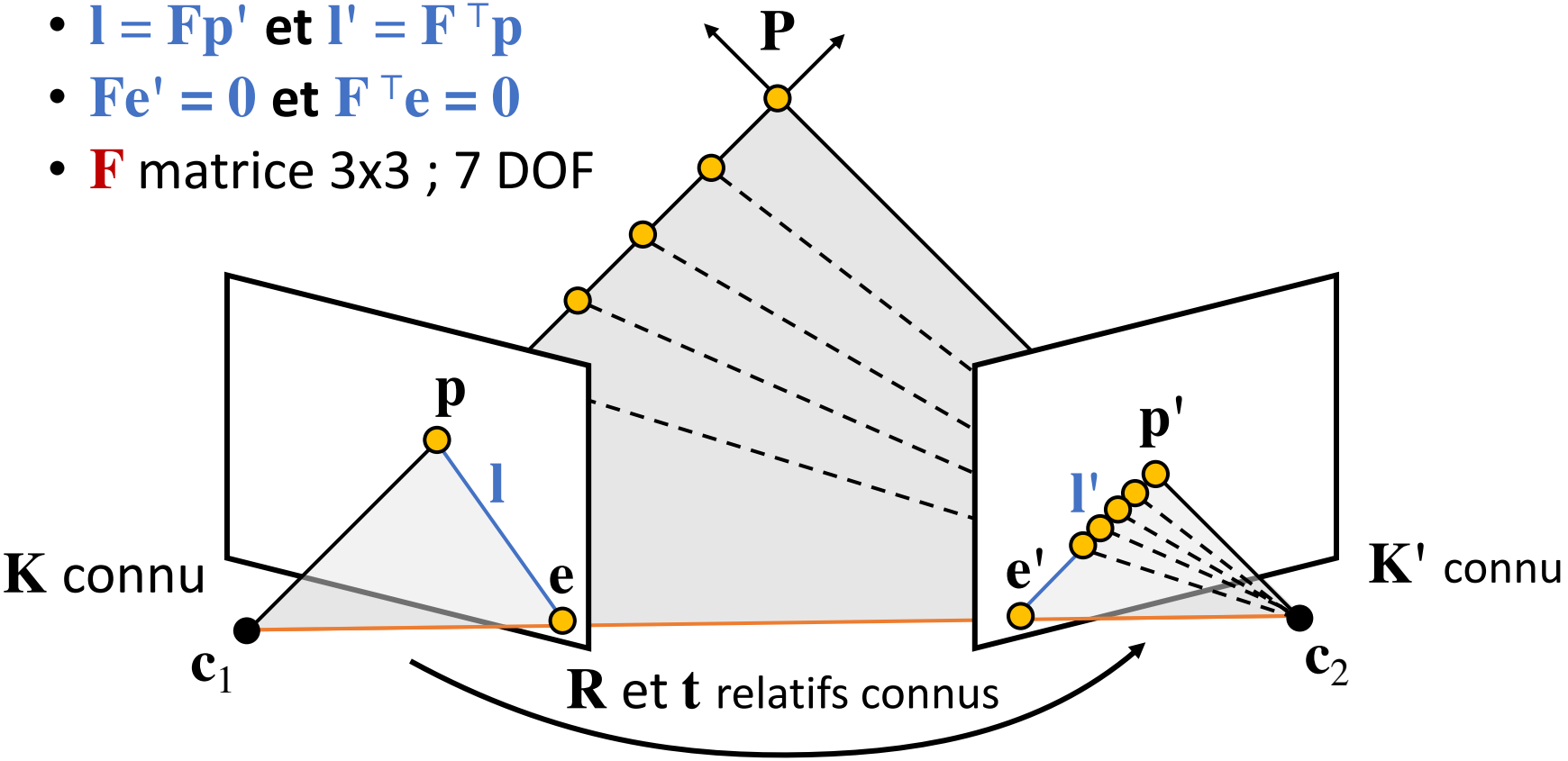


Contrainte épipolaire

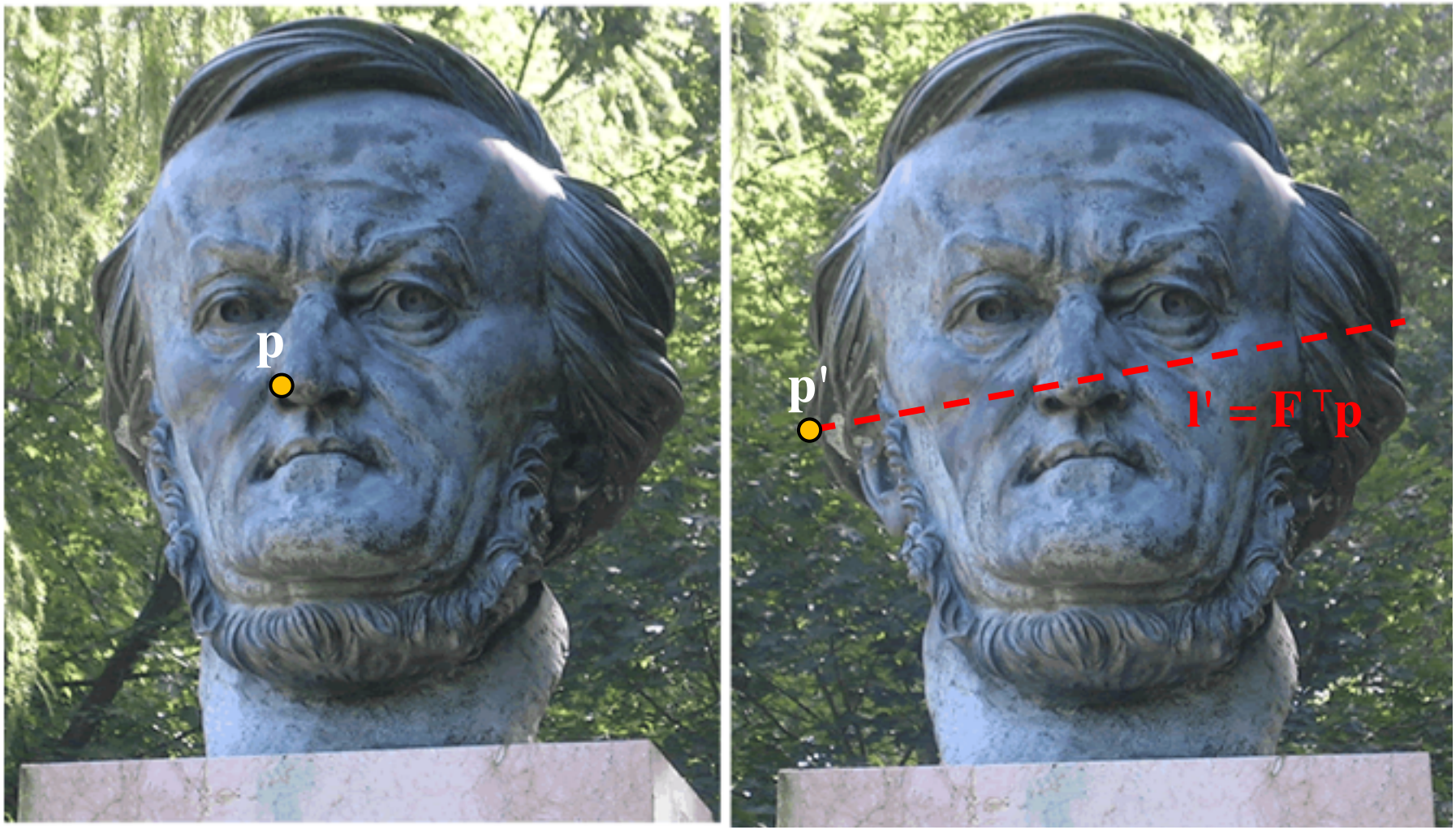
$\mathbf{p}^\top \mathbf{F}\, \mathbf{p}' = 0$ avec $\mathbf{F} = \mathbf{K}^{-\top} [\mathbf{t}_\times] \mathbf{R}\, \mathbf{K}'^{-1}$

matrice fondamentale [Faugeras and Luong, 1992]

- $\mathbf{l} = \mathbf{F}\mathbf{p}'$ et $\mathbf{l}' = \mathbf{F}^\top \mathbf{p}$
- $\mathbf{F}\mathbf{e}' = 0$ et $\mathbf{F}^\top \mathbf{e} = 0$
- $\mathbf{F}$ matrice 3x3 ; 7 DOF



Contrainte épipolaire



Estimer la matrice fondamentale

The Eight-Point Algorithm [Longuet-Higgins, 1981]

$$\mathbf{p}^\top \mathbf{F}\, \mathbf{p}' = 0 \qquad \mathbf{p} = \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \quad \mathbf{p}' = \begin{pmatrix} u' \\ v' \\ 1 \end{pmatrix}$$

$$\Longrightarrow \begin{pmatrix} u & v & 1 \end{pmatrix} \begin{pmatrix} F_{11} & F_{12} & F_{13} \\ F_{21} & F_{22} & F_{23} \\ F_{31} & F_{32} & F_{33} \end{pmatrix} \begin{pmatrix} u' \\ v' \\ 1 \end{pmatrix} = 0$$

$$\Longrightarrow \begin{pmatrix} uu' & uv' & u & vu' & vv' & v & u' & v' & 1 \end{pmatrix} \begin{pmatrix} F_{11} \\ F_{12} \\ F_{13} \\ F_{21} \\ F_{22} \\ F_{23} \\ F_{31} \\ F_{32} \\ F_{33} \end{pmatrix} = 0$$

Estimer la matrice fondamentale

Avec (au moins) 8 points **en correspondance**

$$\begin{pmatrix} u_1u'_1 & u_1v'_1 & u_1 & v_1u'_1 & v_1v'_1 & v_1 & u'_1 & v'_1 & 1 \\ u_2u'_2 & u_2v'_2 & u_2 & v_2u'_2 & v_2v'_2 & v_2 & u'_2 & v'_2 & 1 \\ u_3u'_3 & u_3v'_3 & u_3 & v_3u'_3 & v_3v'_3 & v_3 & u'_3 & v'_3 & 1 \\ u_4u'_4 & u_4v'_4 & u_4 & v_4u'_4 & v_4v'_4 & v_4 & u'_4 & v'_4 & 1 \\ u_5u'_5 & u_5v'_5 & u_5 & v_5u'_5 & v_5v'_5 & v_5 & u'_5 & v'_5 & 1 \\ u_6u'_6 & u_6v'_6 & u_6 & v_6u'_6 & v_6v'_6 & v_6 & u'_6 & v'_6 & 1 \\ u_7u'_7 & u_7v'_7 & u_7 & v_7u'_7 & v_7v'_7 & v_7 & u'_7 & v'_7 & 1 \\ u_8u'_8 & u_8v'_8 & u_8 & v_8u'_8 & v_8v'_8 & v_8 & u'_8 & v'_8 & 1 \end{pmatrix} \begin{pmatrix} F_{11} \\ F_{12} \\ F_{13} \\ F_{21} \\ F_{22} \\ F_{23} \\ F_{31} \\ F_{32} \\ F_{33} \end{pmatrix} = 0$$

$$\Leftrightarrow \mathbf{Wf} = 0$$

$\Longrightarrow$ Résolution aux moindres carrés par SVD

Estimer la matrice fondamentale

$\mathbf{W}$ mal conditionnée : $u, v \gg w = 1$

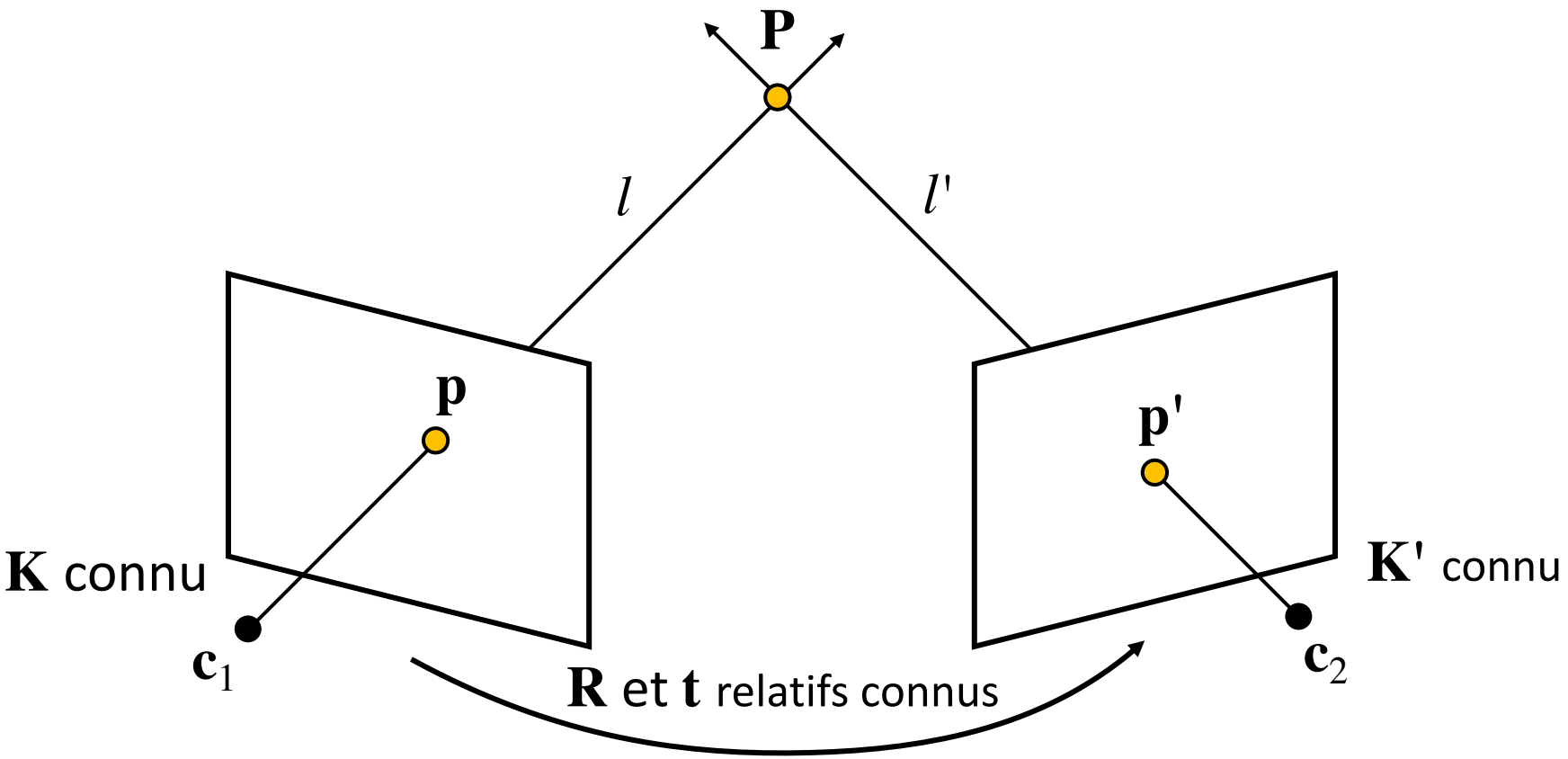
$\Longrightarrow$ ***The Normalized Eight-Point Algorithm*** [Hartley, 1997]

1. Trouver la transformation $\mathbf{T}$ (resp. $\mathbf{T'}$) telle que :
   - Origine = centroïde des points
   - Distance moyenne des points à l'origine ≈ 2px
2. Normaliser les points : $q = \mathbf{T}p$ et $q' = \mathbf{T'}p'$
3. Estimer $\mathbf{F}_q$ avec l'algo. précédent
4. Dénormaliser : $\mathbf{F} = \mathbf{T}^\top \mathbf{F}_q \mathbf{T'}$

Reconstruction des positions 3D

Triangulation : $\mathbf{P}$ à l'intersection de l et l'

…mais en pratique, pas d'intersection exacte !

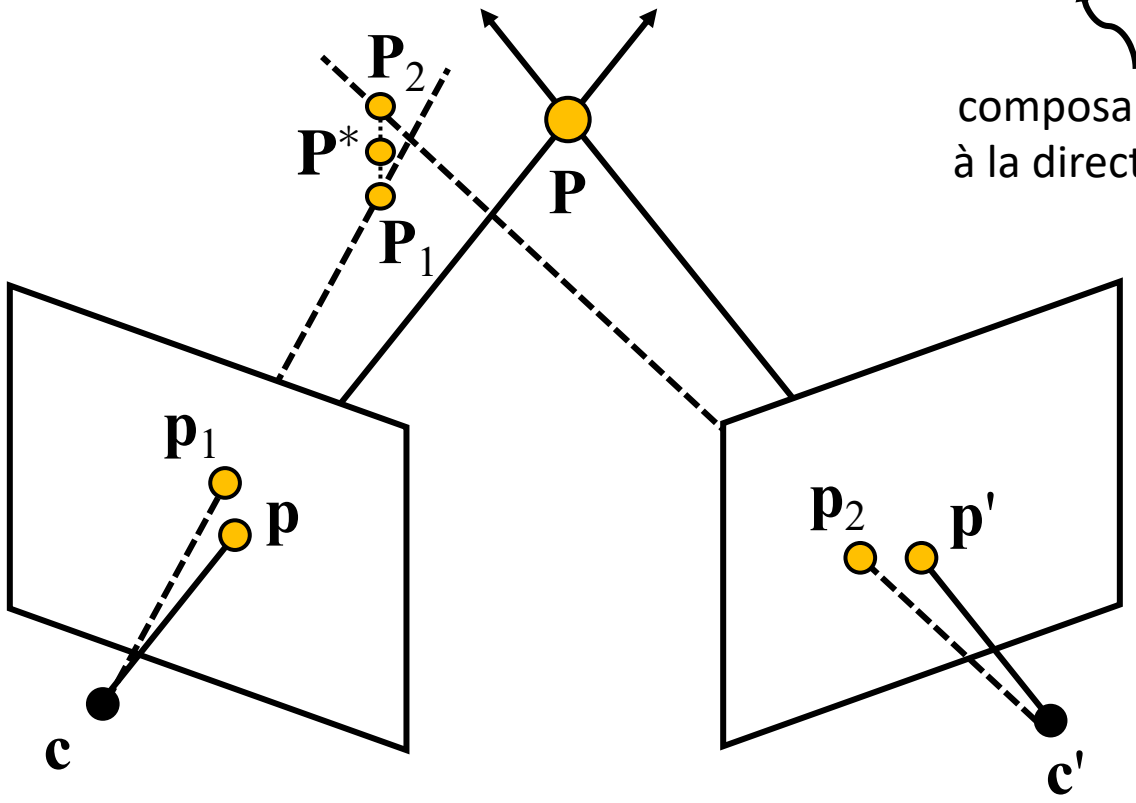


Triangulation (stéréo)

En pratique, **trouver $\mathbf{P}^*$ au plus proche de $\mathbf{P}_1$ et $\mathbf{P}_2$**

$\Longrightarrow$ minimisant : $||(\mathbf{P}^* - \mathbf{c})_\perp||^2 + ||(\mathbf{P}^* - \mathbf{c}')_\perp||^2$

composante normale
à la direction du rayon



Reconstruction des positions 3D

Par définition, $\mathbf{p} \times (\mathbf{MP}) = 0$ et $\mathbf{p}' \times (\mathbf{M}'\mathbf{P}) = 0$

$\Longrightarrow$ 4 contraintes d'égalité :

$$\mathbf{AP} = 0 \text{ avec } \mathbf{A} = \begin{pmatrix} u\mathbf{M}_3 - \mathbf{M}_1 \\ v\mathbf{M}_3 - \mathbf{M}_2 \\ u'\mathbf{M}'_3 - \mathbf{M}'_1 \\ v'\mathbf{M}'_3 - \mathbf{M}'_2 \end{pmatrix} \text{ et } \mathbf{M} = (\mathbf{M}_1 \,|\, \mathbf{M}_2 \,|\, \mathbf{M}_3)$$

$\Longrightarrow$ **système linéaire** (SVD) similaire à DLT

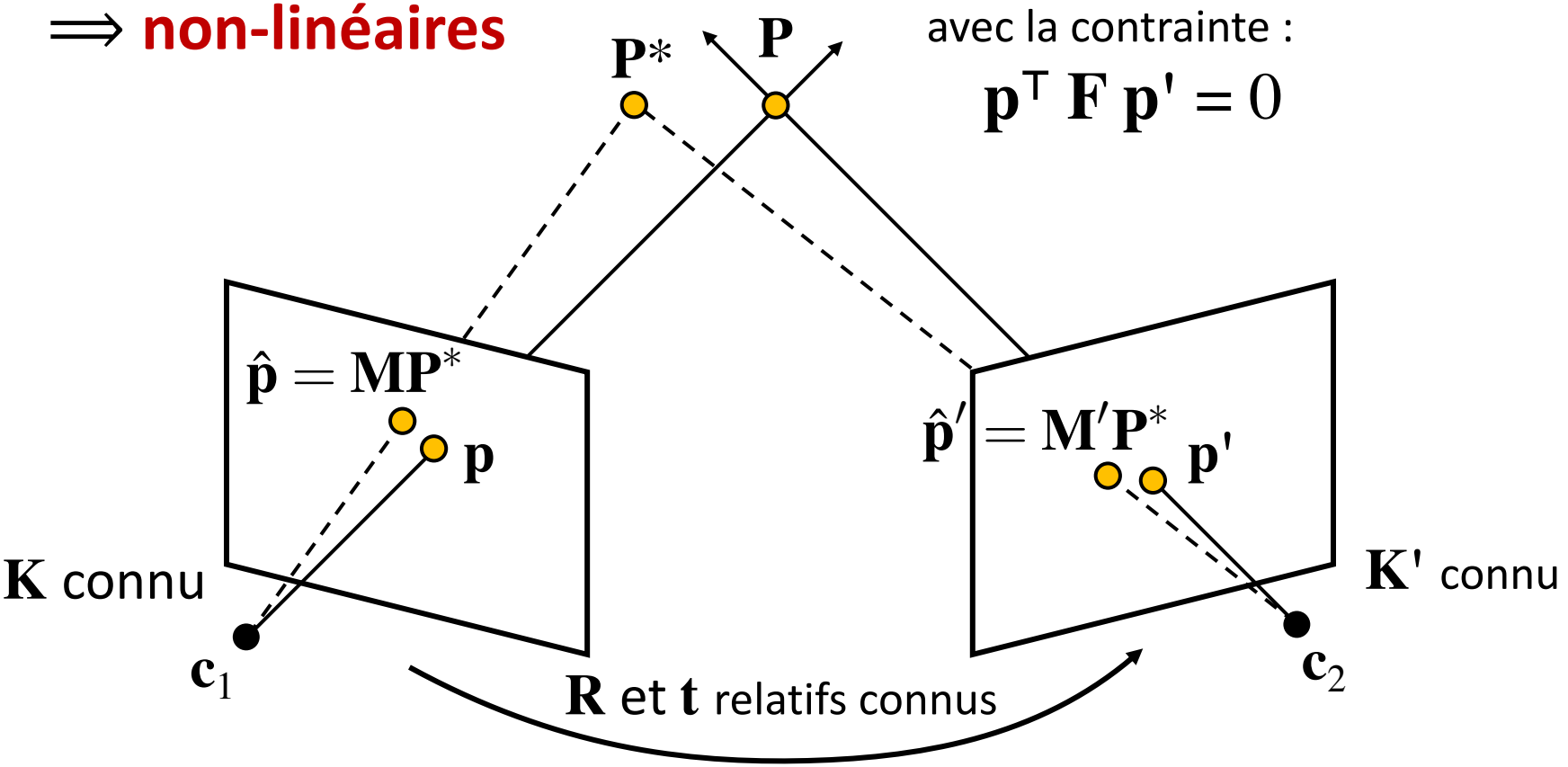
Reconstruction des positions 3D

Optimisation : trouver $\mathbf{P}^*$ minimisant
l'erreur de reprojection : $||\mathbf{p} - \mathbf{M}\mathbf{P}^*||^2 + ||\mathbf{p}' - \mathbf{M}'\mathbf{P}^*||^2$

$\Longrightarrow$ **non-linéaires**

avec la contrainte :

$$\mathbf{p}^\top \mathbf{F}\, \mathbf{p}' = 0$$

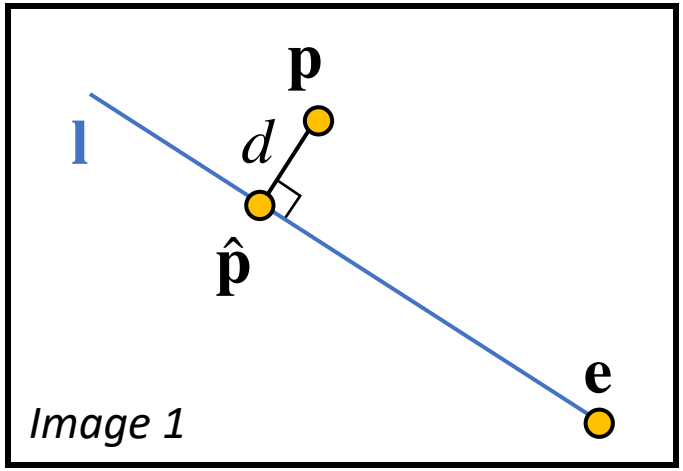


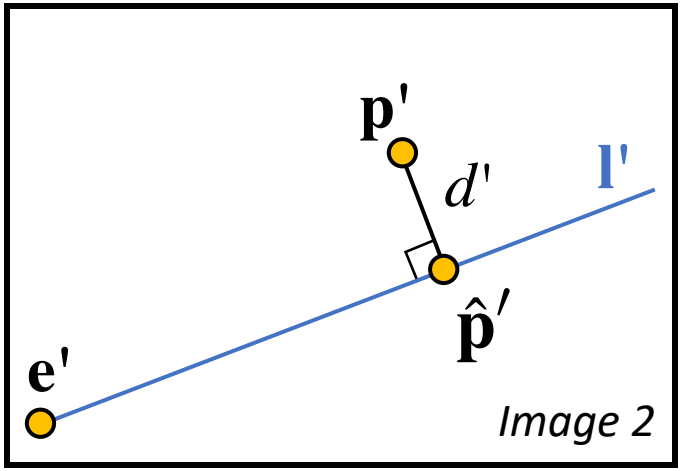
Reconstruction des positions 3D

Contrainte épipolaire :
projeté de $\mathbf{P^*}$ sur $\mathbf{l}$ (resp. $\mathbf{l'}$) dans l'image 1 (resp. 2)

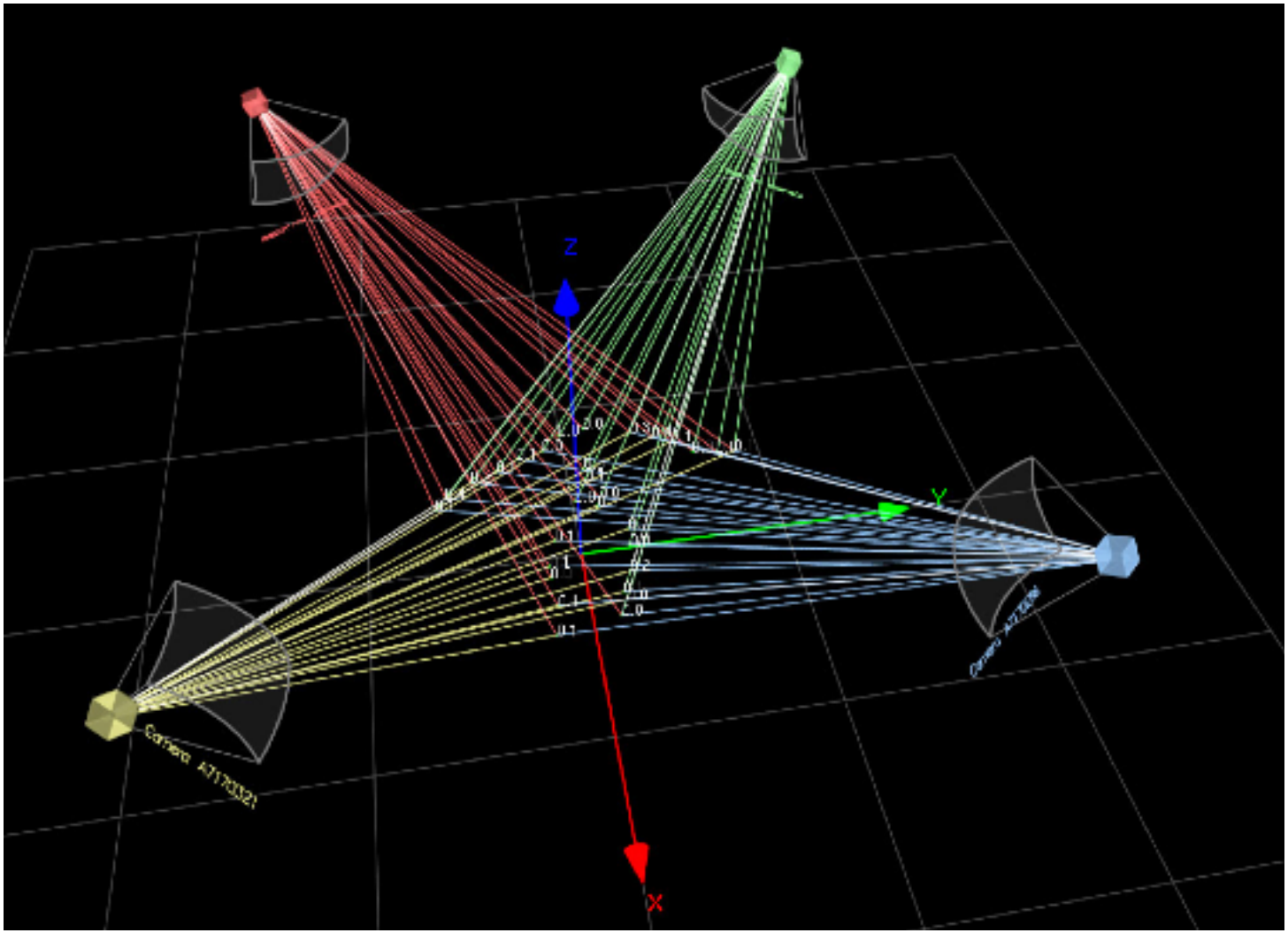
L'optimum minimise : $d^2 + d'^2$

$\Longrightarrow$ **solution exacte** d'une équation de degré 6





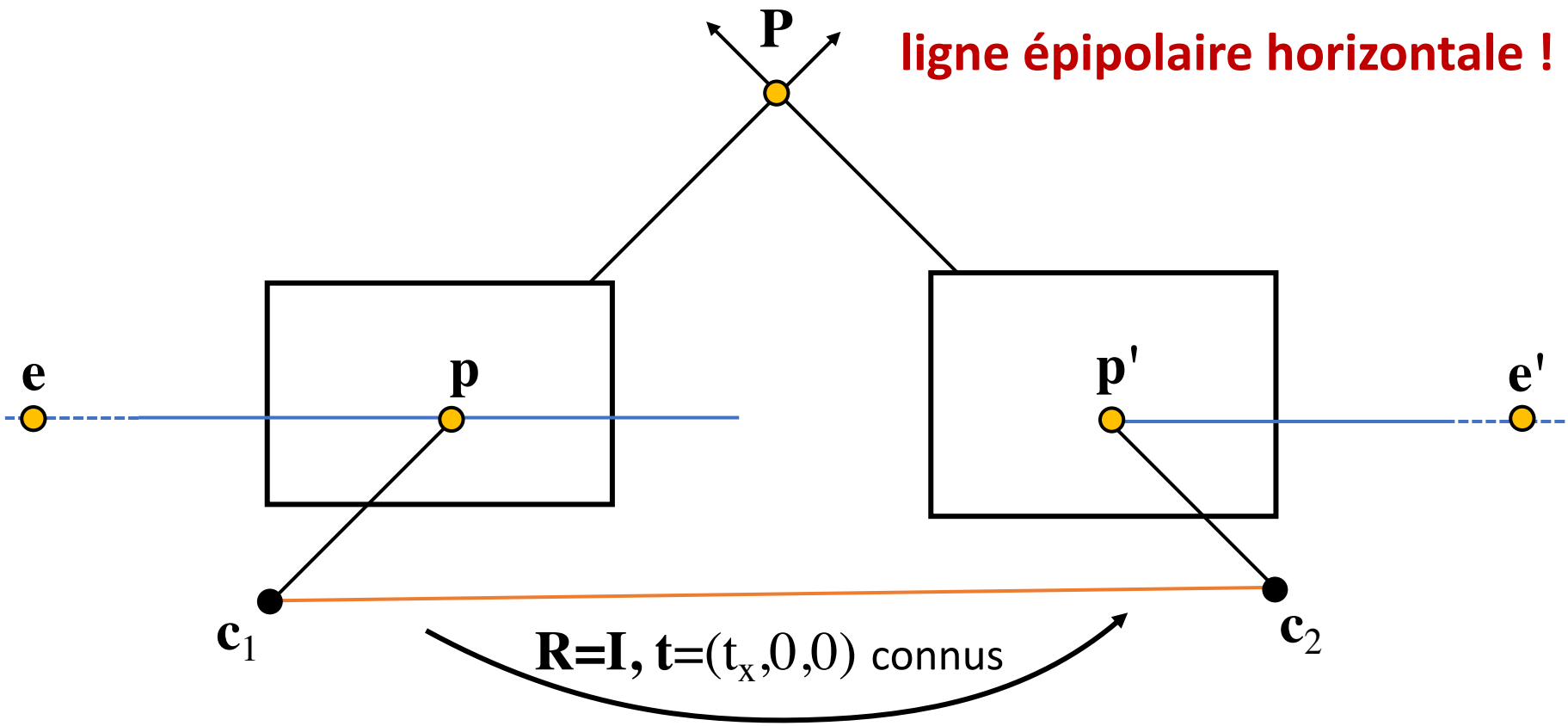
Contrainte épipolaire



Caméras parallèles

$$\mathbf{E} = [\mathbf{t}_\times]\mathbf{R} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -t_x \\ 0 & t_x & 0 \end{pmatrix} \Longrightarrow \mathbf{E}\mathbf{p}' = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -t_x \\ 0 & t_x & 0 \end{pmatrix} \begin{pmatrix} u' \\ v' \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ -t_x \\ t_x v' \end{pmatrix}$$

ligne épipolaire horizontale !

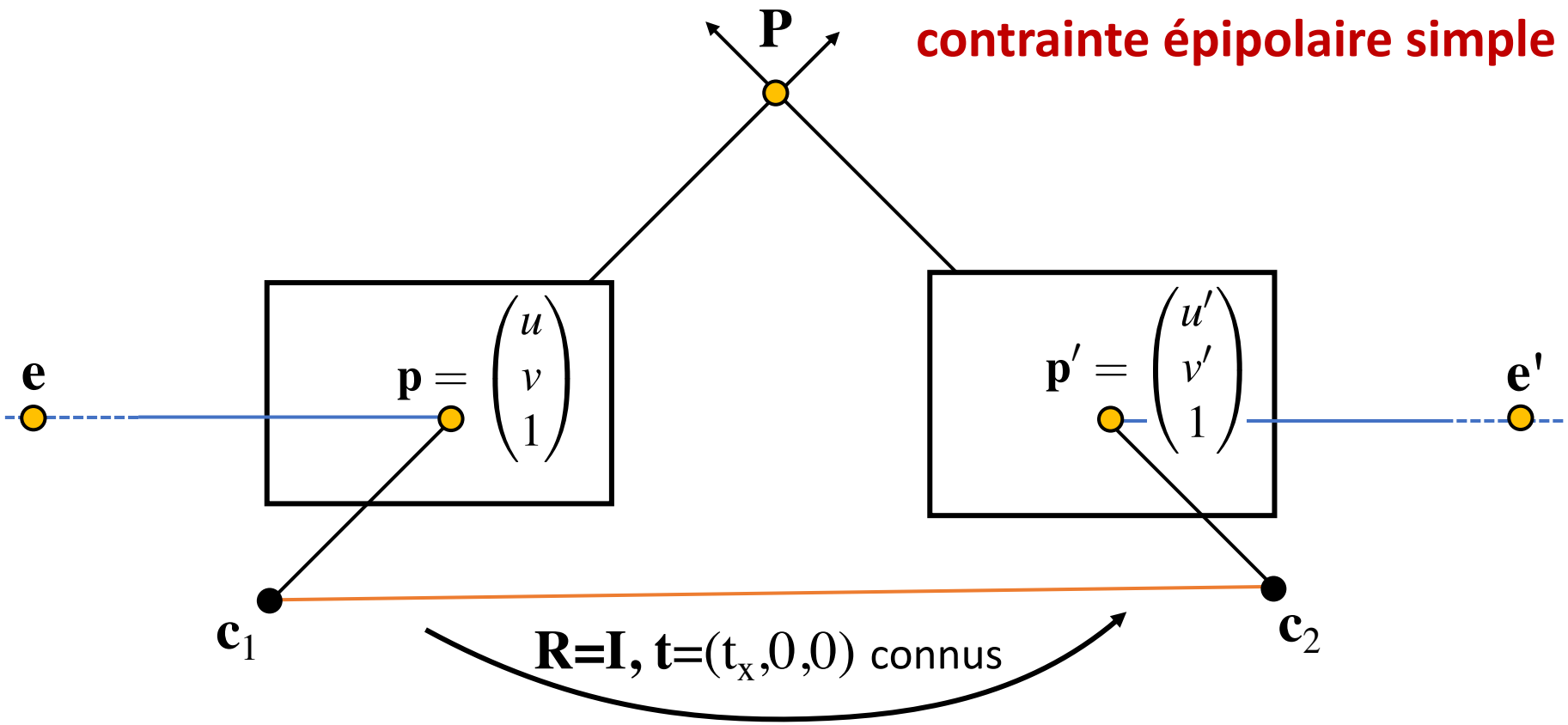


Caméras parallèles

$$\mathbf{p}^{\mathsf{T}}\,\mathbf{E}\,\mathbf{p}' = 0 \Longrightarrow \begin{pmatrix} u & v & 1 \end{pmatrix} \begin{pmatrix} 0 \\ -t_x \\ t_x v \end{pmatrix} = 0 \Longrightarrow t_x v = t_x v'$$

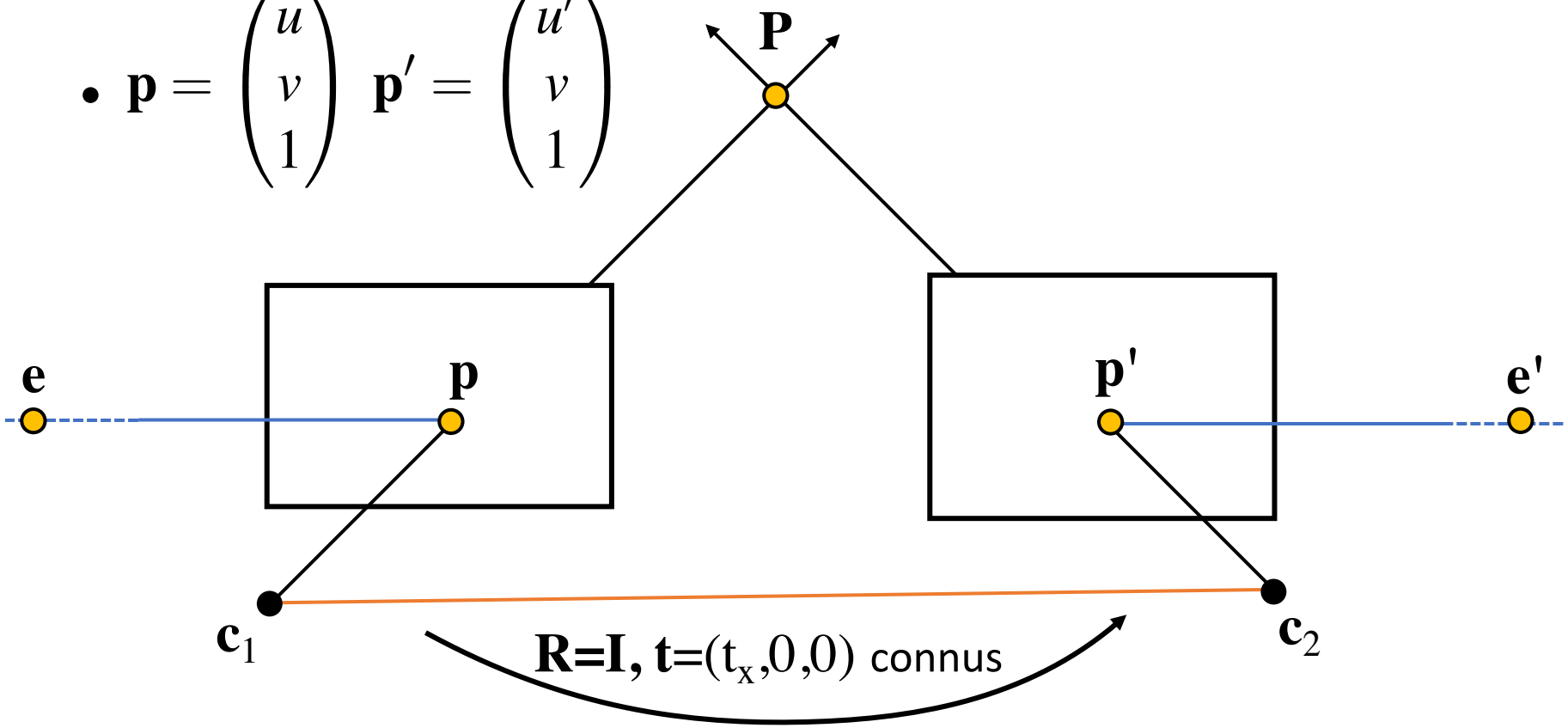
$$\Longrightarrow v = v'$$

contrainte épipolaire simple



Caméras parallèles

- Lignes épipolaires **horizontales**
- Épipoles **à l'infini**
- $\mathbf{p} = \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \quad \mathbf{p}' = \begin{pmatrix} u' \\ v \\ 1 \end{pmatrix}$



Rectification d’image

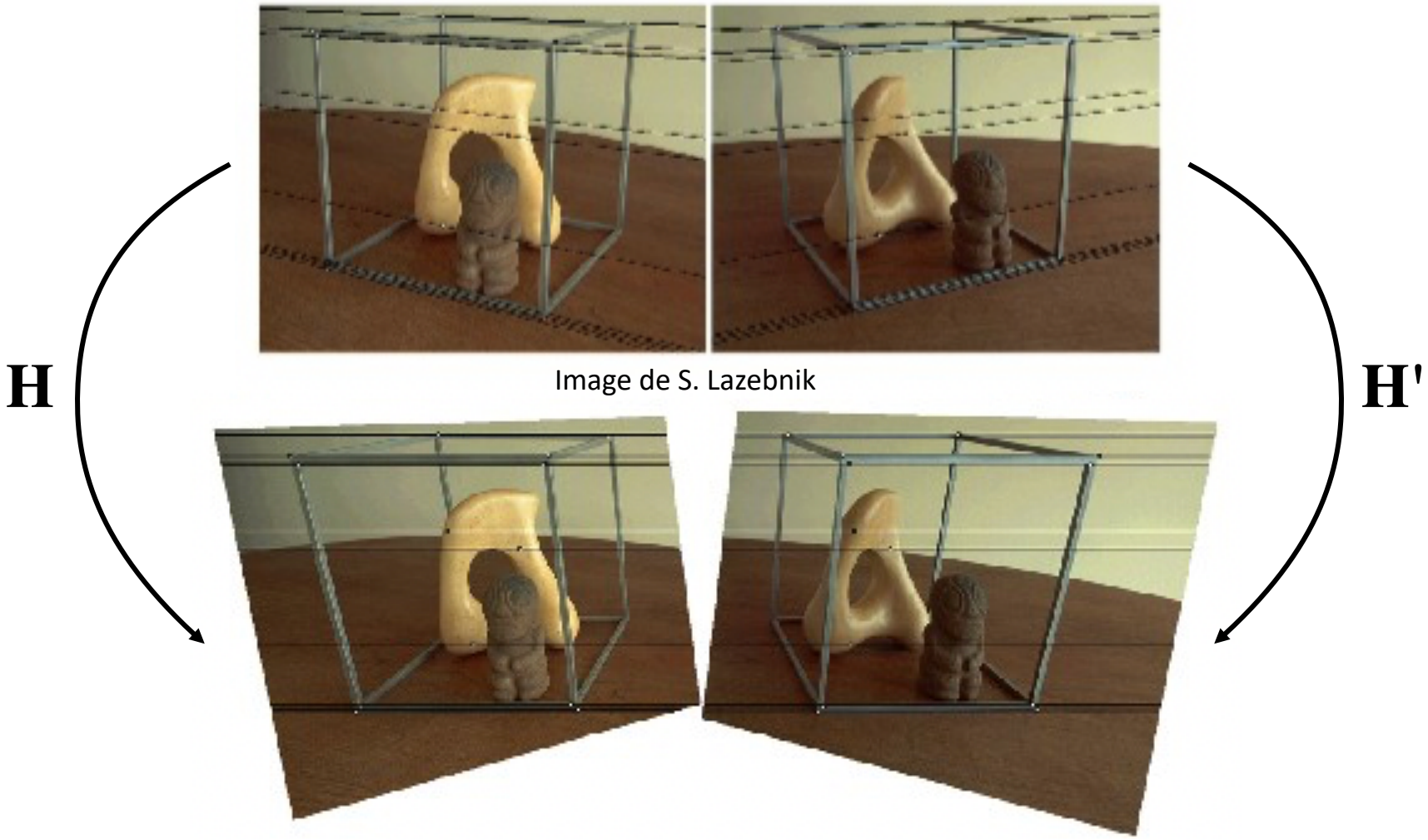


Image de S. Lazebnik

Homographies $\mathbf{H}$ et $\mathbf{H}'$

1. Trouver $\mathbf{H}'$ tel que $\mathbf{e}' = (f,0,1)^\mathsf{T}$ soit envoyé à l'infini en $(1,0,0)^\mathsf{T} \Longrightarrow$ 4 degrés de libertés

   $\Longrightarrow$ minimiser les distorsions (aussi rigide que possible)

2. Trouver $\mathbf{H}$ de la forme $\mathbf{H}_\mathrm{A}\mathbf{H}'\mathbf{R}$ avec $\mathbf{H}_A = \begin{pmatrix} a & b & c \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ transformation affine

   où $\mathbf{F} = [\mathbf{e}'_\times]\,\mathbf{R}$ minimisant :

   $$\sum_i ||\mathbf{H}_A\mathbf{H}'\mathbf{M}\mathbf{x}_i - \mathbf{H}'\mathbf{x}'_i||^2$$

   $\Longrightarrow$ linéaire, moindres carrées

Rectification d'image

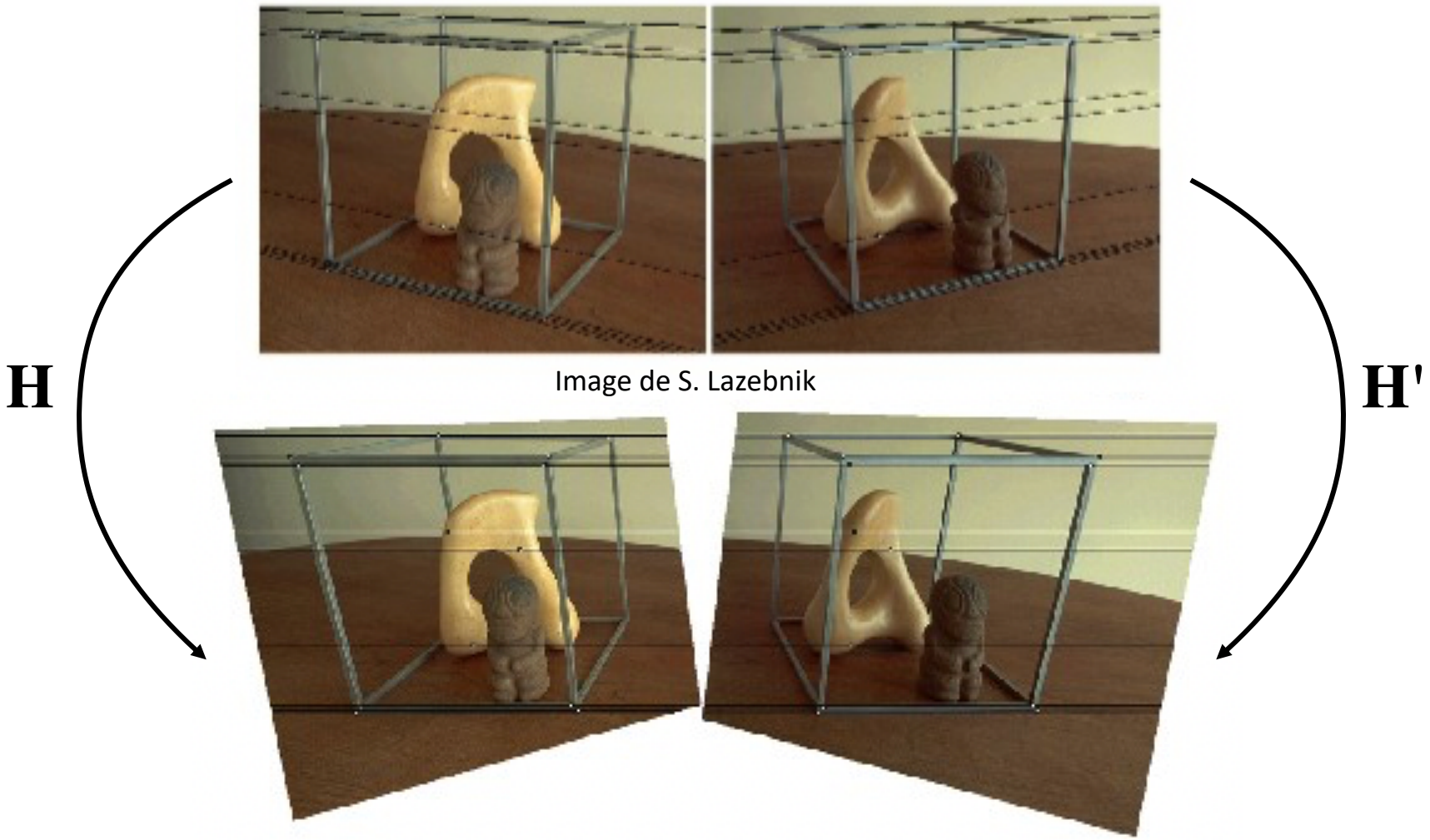


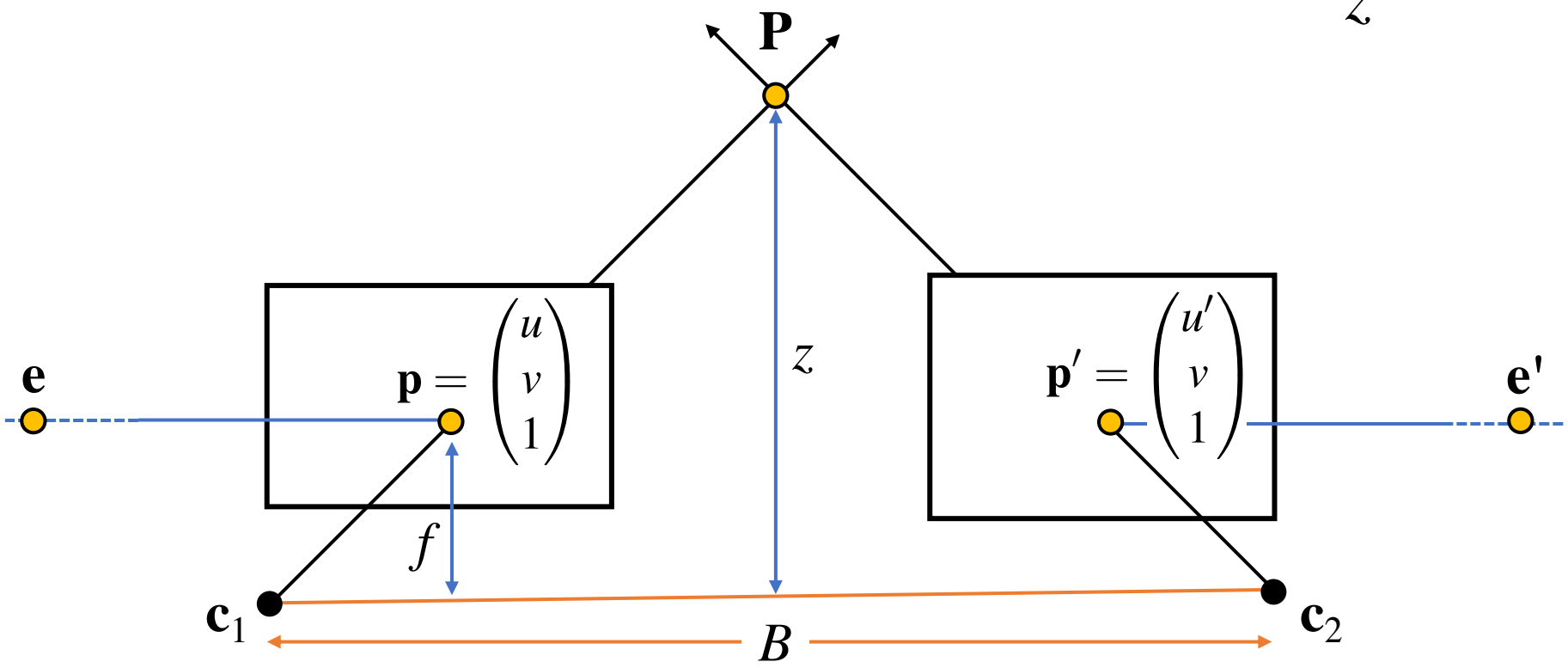
Image de S. Lazebnik

$\Longrightarrow$ simplifie la triangulation et la mise en correspondance

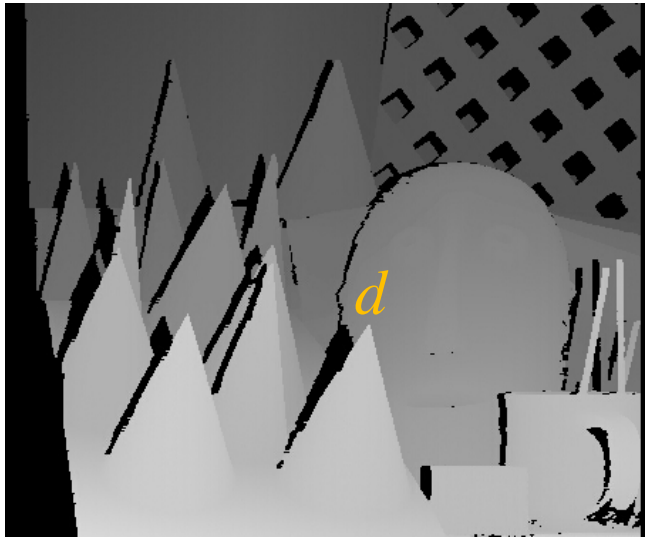
Estimation de la profondeur

Disparité inversement proportionnelle à z :

$$d = u - u' = f\frac{B}{z}$$

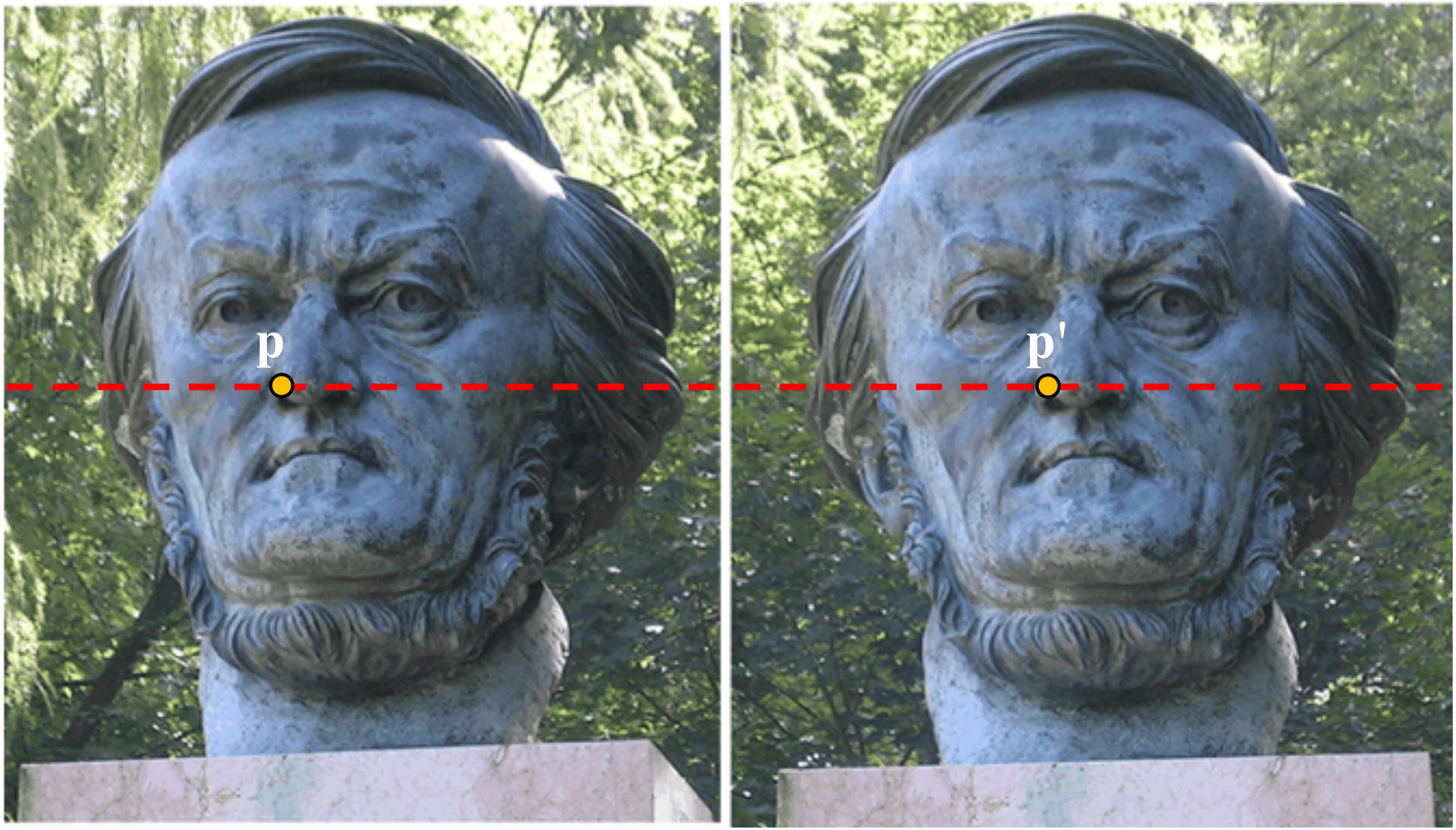


Carte de disparité / profondeur



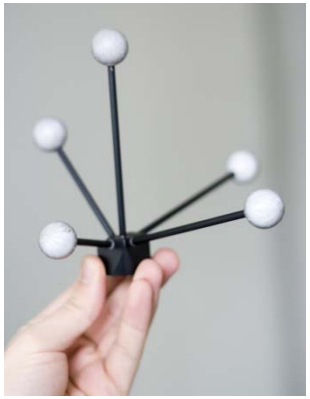
$$d = u - u' = f\frac{B}{z}$$

Mise en correspondance



Chaîne de traitement

1. **Calibration** des caméras
2. **Détection** des blobs
3. **Mise en correspondance** entre les vues
4. **Reconstruction** des positions 3D
5. **Ajustement** d'une constellation rigide
6. **Estimation de la pose** de a constellation (position et orientation)

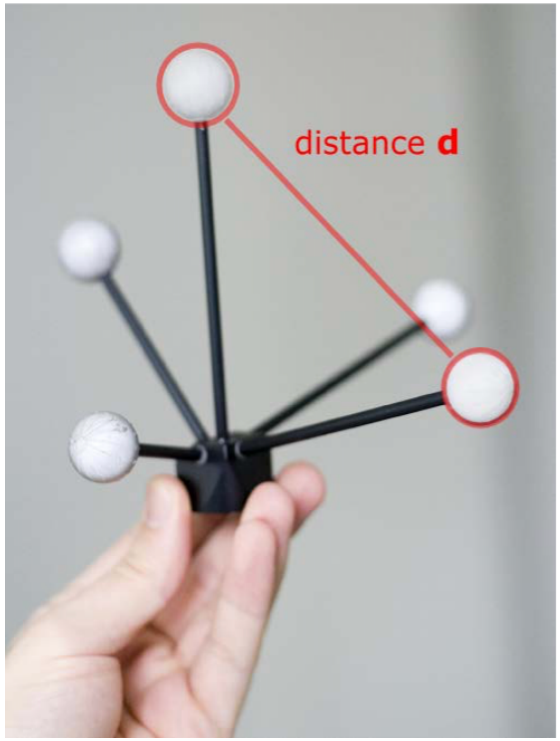


Ajustement d'un modèle rigide

1. Pré-calcul des **distances inter-marqueurs** en 3D de chaque constellation

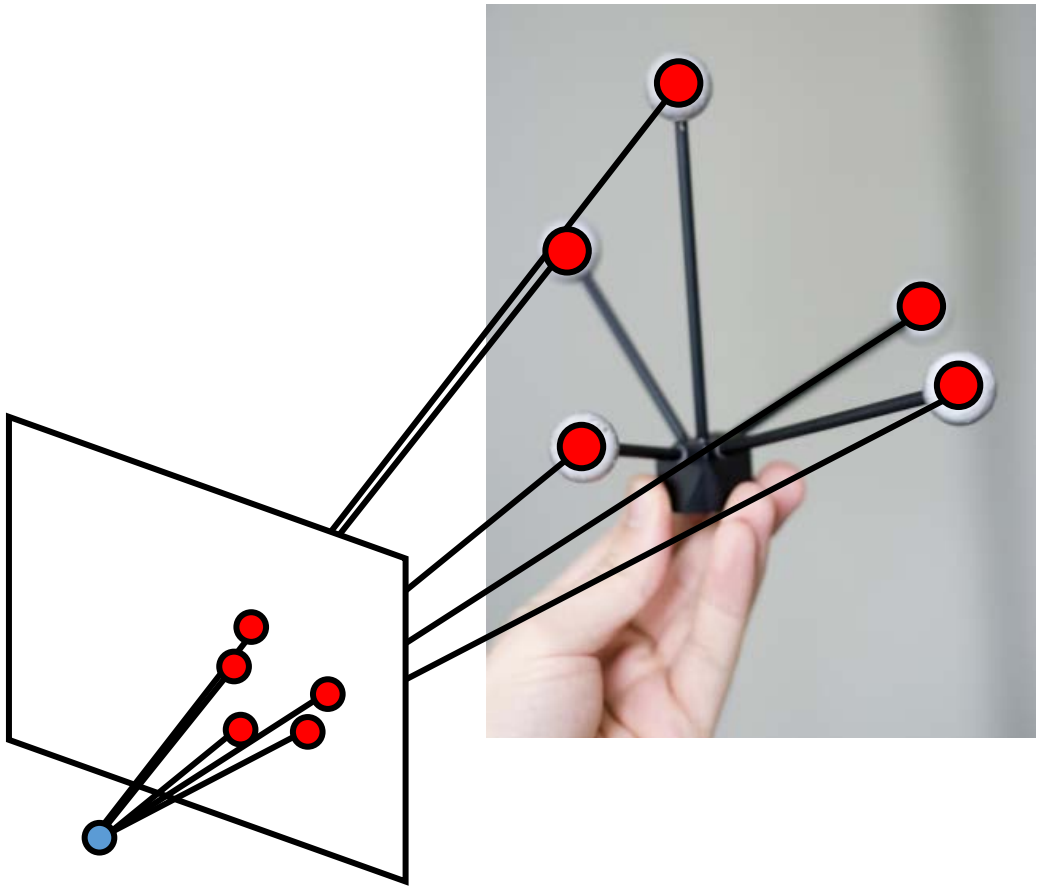
$$\Longrightarrow \frac{N!}{2(N-2)!} \text{ distances pour } N \text{ marqueurs}$$

2. À chaque nouvelle image :
   - calcul des distances entre chaque pair de marqueurs
   - sélection de la constellation la plus probable



Estimation de la pose

$\Longrightarrow$ estimer $\mathbf{R}$ et $\mathbf{t}$ par alignement 3D



Systèmes optiques

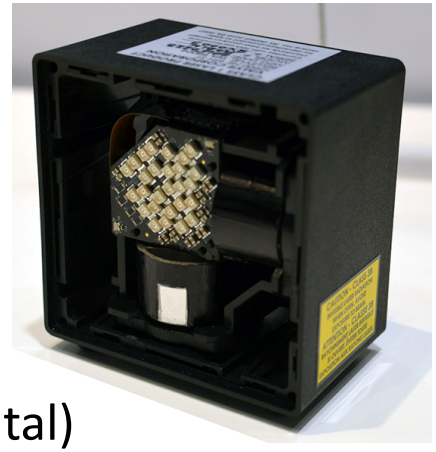
Avantages :

- ✓ liberté de mouvement, grandes zones de suivi
- ✓ précision sous-millimétrique
- ✓ marqueurs actifs : pas d'ambiguïté entre les marqueurs
- ✓ marqueurs passifs : ajout de marqueurs faciles

Inconvénients :

- ✗ occlusions
- ✗ cher (~10k€)
- ✗ calibration
- ✗ marqueurs actifs : nombre de marqueurs limité
- ✗ marqueurs passifs : ambiguïté entre les marqueurs

HTC Vive Lighthouse

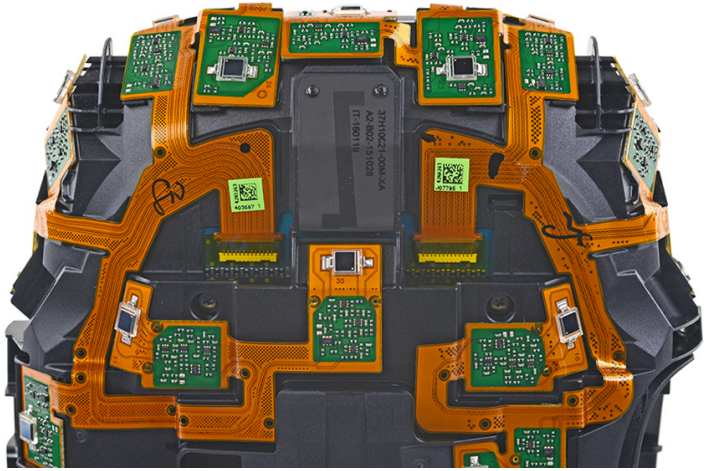


2 stations de base :

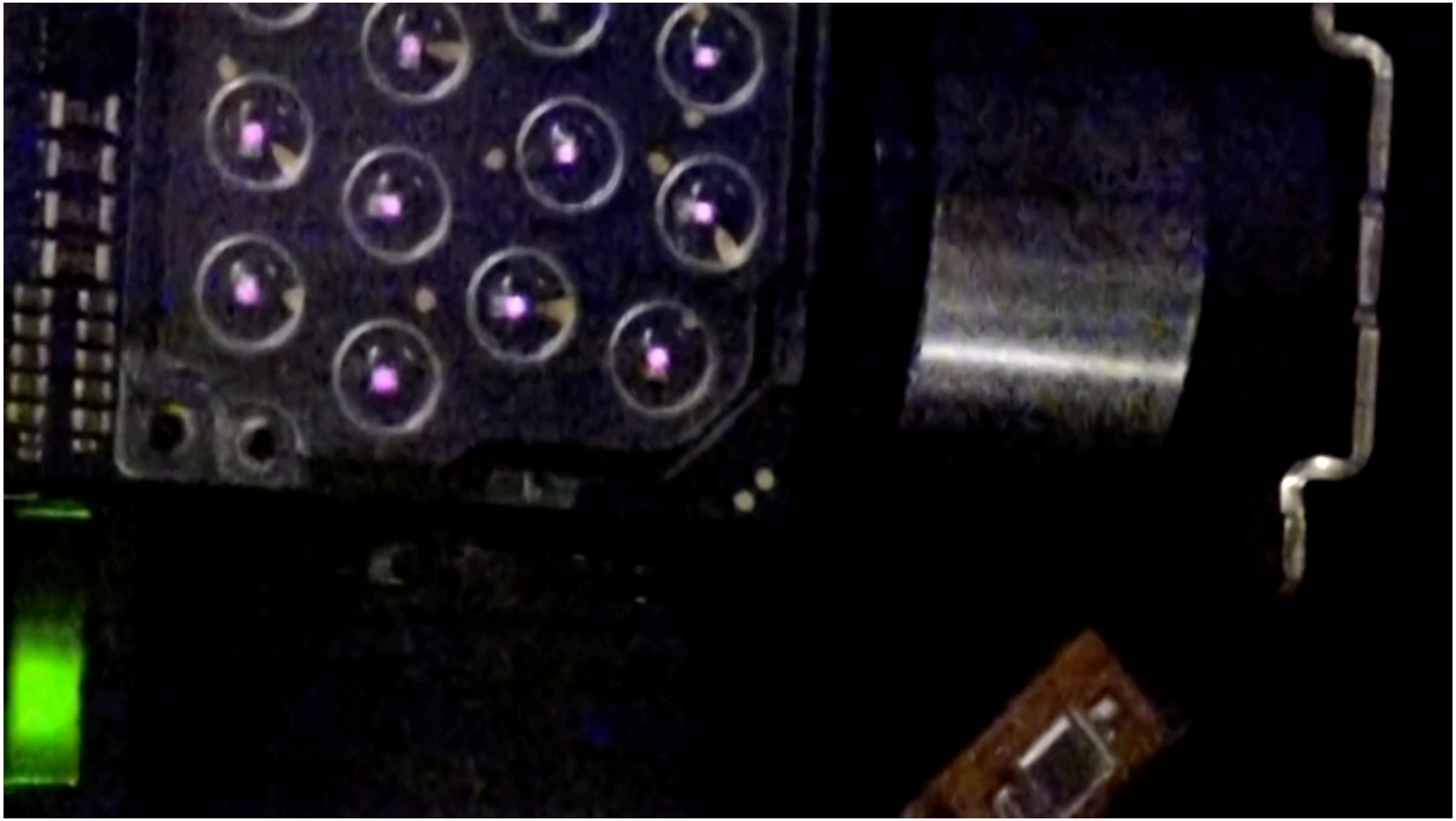
- 2 lignes laser IR par station (vertical et horizontal) en rotation à 60 Hz (déphasées)
- 1 flash IR de synchronisation à 120Hz

Photodiodes sur le casque et les manettes

mesure du temps à la réception
(time of arrival / flight)

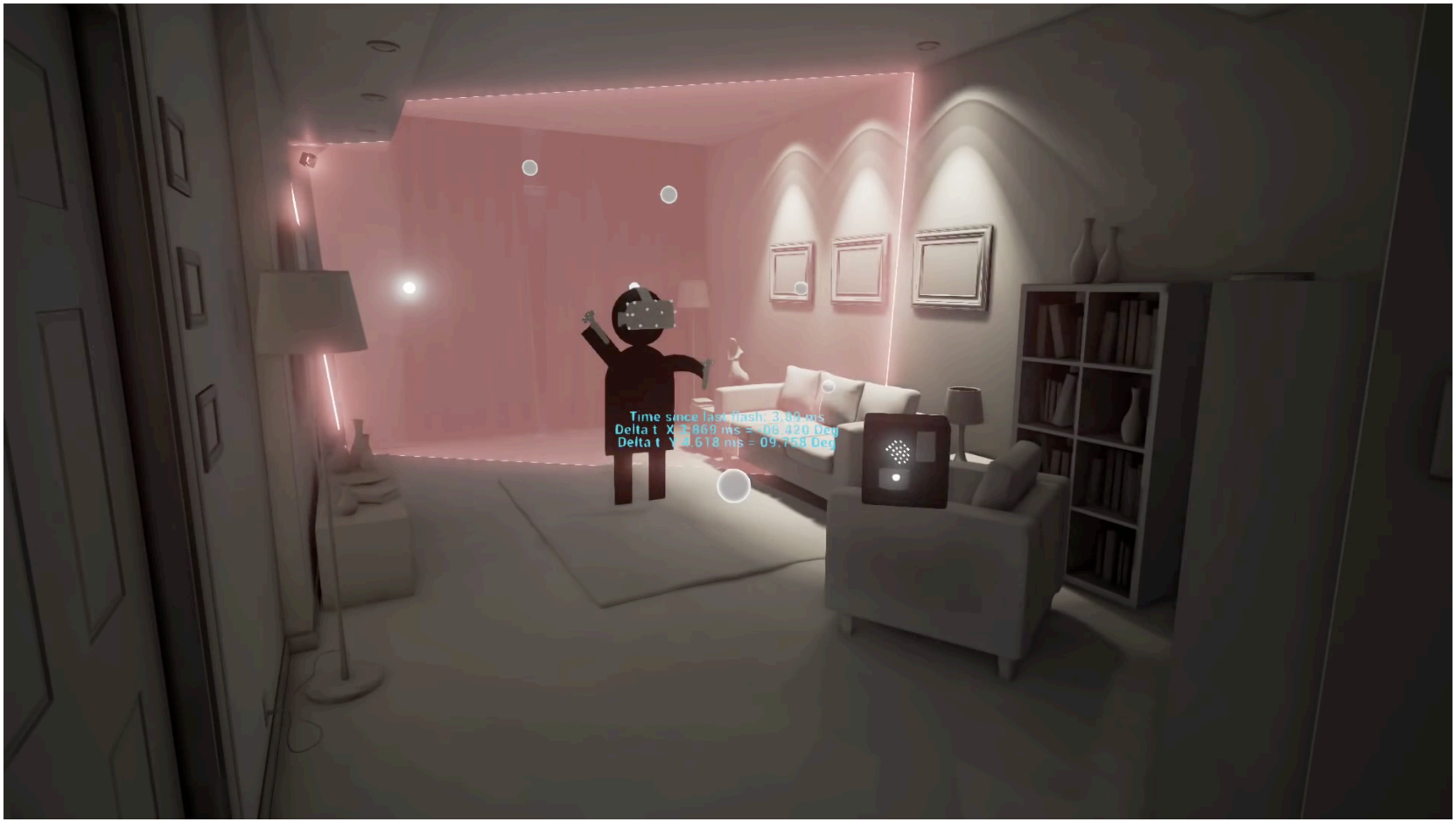


HTC Vive Lighthouse



https://www.youtube.com/watch?v=avBt_P0wg_Y

HTC Vive Lighthouse



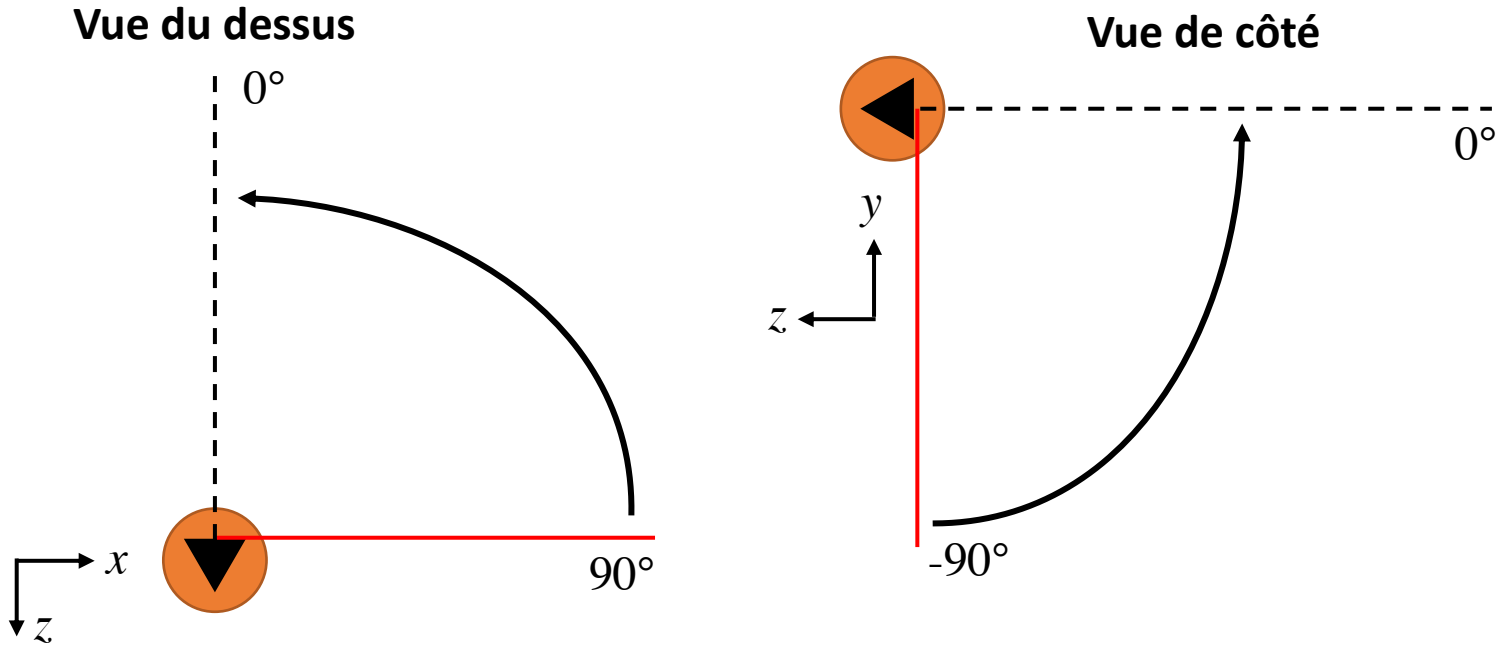
https://www.youtube.com/watch?v=J54dotTt7k0

Calcul des positions

Au flash, laser 1 à 90° horizontalement
laser 2 à -90° verticalement

Rotation de 360° en 1/60ème de seconde

Photodiode mesure Δt depuis le flash



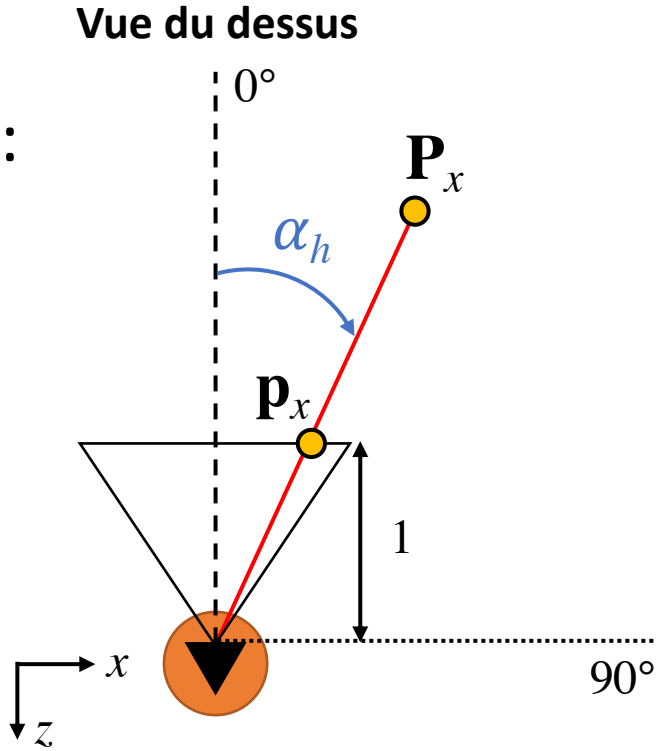
Calcul des positions

Conversion du temps Δt à la réception en angle α :

$$\alpha_h[^\circ] = 90[^\circ] - \frac{\Delta t_h[s]}{\frac{1/60}{360}\left[\frac{s}{^\circ}\right]} \qquad \alpha_v[^\circ] = 90[^\circ] + \frac{\Delta t_v[s]}{\frac{1/60}{360}\left[\frac{s}{^\circ}\right]}$$

puis en position relative $\mathbf{p}$
sur le plan à distance unitaire :

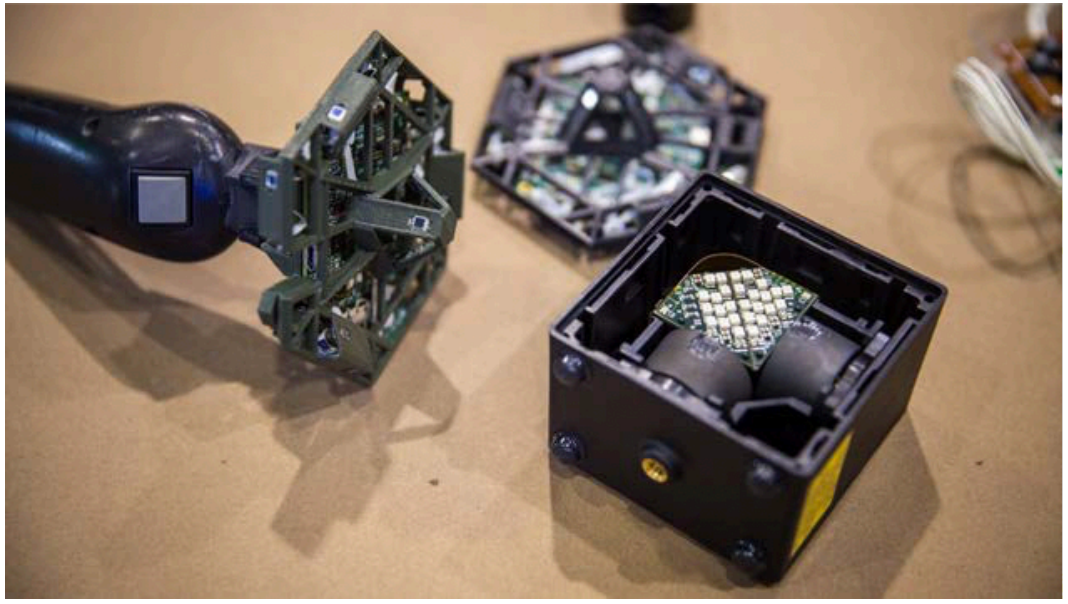
$$\mathbf{p}_{x/y} = \tan\left(\frac{\alpha_{h/v}[^\circ]}{360[^\circ]} 2\pi[\text{rad}]\right)$$



Steam VR Lighthouse

Suivi optique + IMU

- Très faible latence : 1ms max. pour le casque,
  2.7ms pour les manettes
- Très grande précision : 0.3mm en latéral, 2.1mm en z



http://doc-ok.org/?p=1478

Inside-Out tracking

Capteur embarqué sur l'objet suivi

Essentiellement des **systèmes optiques**

- basés marqueurs
- utilisant des caractéristiques naturelles de l'image (coins, contours…)
- avec capteur de profondeur

https://xinreality.com/wiki/Inside-out_tracking

https://goo.gl/dFH1Js

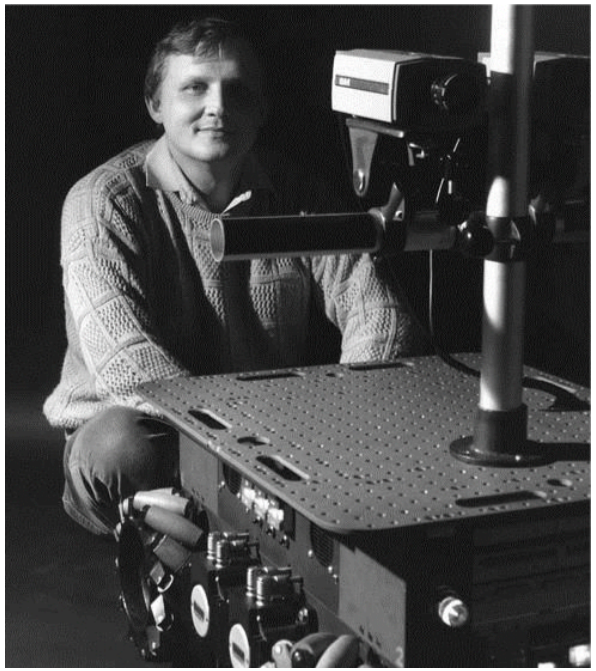
Inside-Out tracking

Nécessite de cartographier l'environnement

Visual Odometry (VO)

⊃ *Simultaneous Localization And Mapping* (SLAM)

⊃ *Structure from Motion* (SfM)



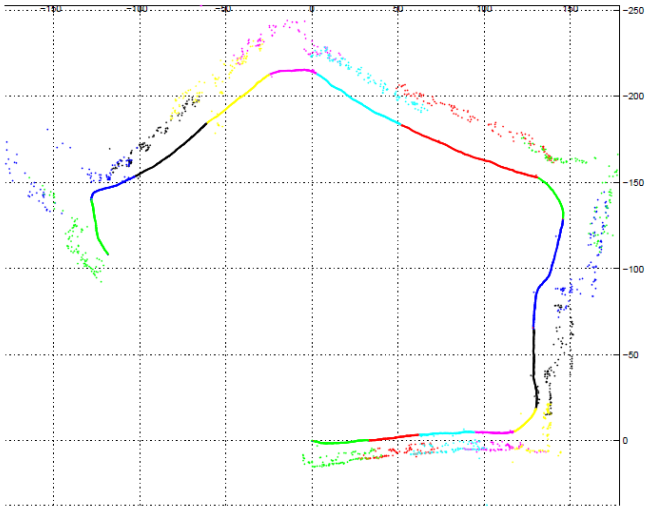
Hans Moraveck
(NASA/JPL)

Visual Odometry

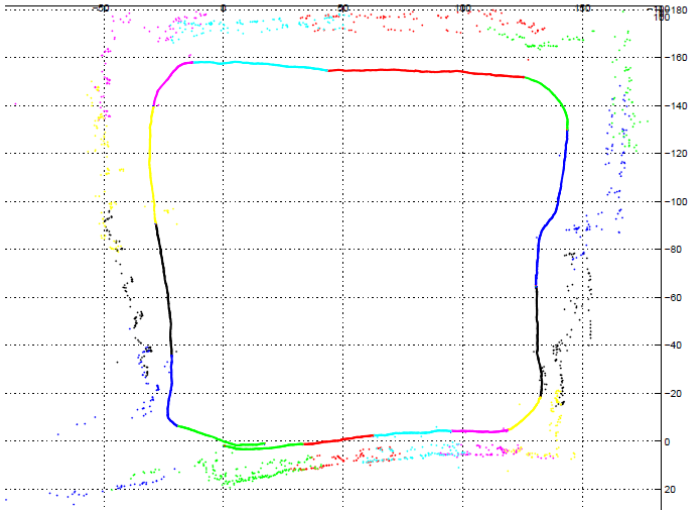
Calcul incrémental de la trajectoire de la caméra (pose après pose)

- accès séquentiel aux images
- pas d'optimisation globale ⟹ risque de dérive
- en temps-réel

Visual Odometry

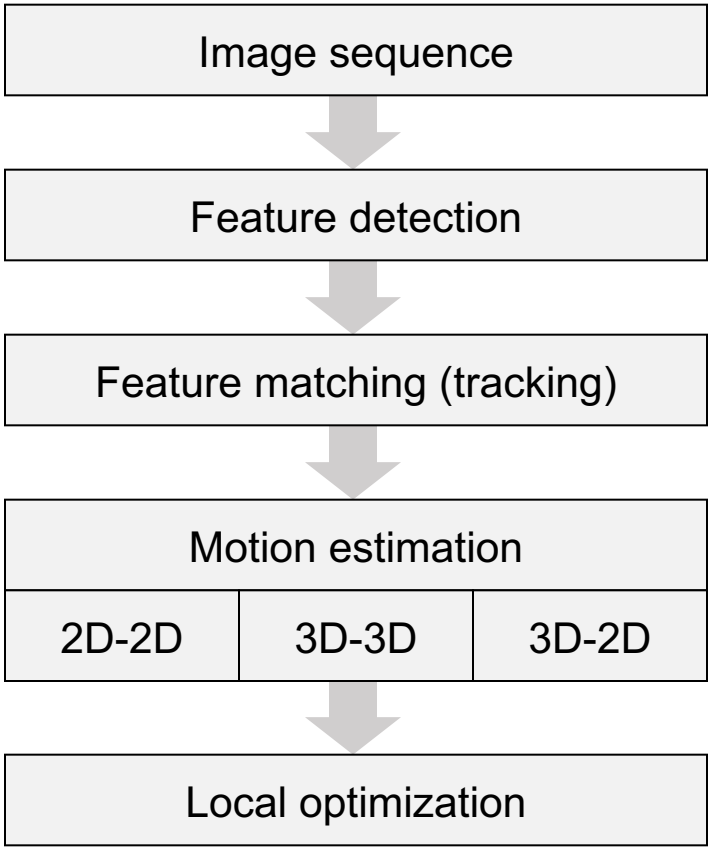


Visual SLAM



[Clemente et al. 2007]

Visual Odometry



Scaramuzza, D. &, Fraundorfer, F., Visual Odometry
http://rpg.ifi.uzh.ch/docs/VO_Part_I_Scaramuzza.pdf
http://rpg.ifi.uzh.ch/docs/VO_Part_II_Scaramuzza.pdf