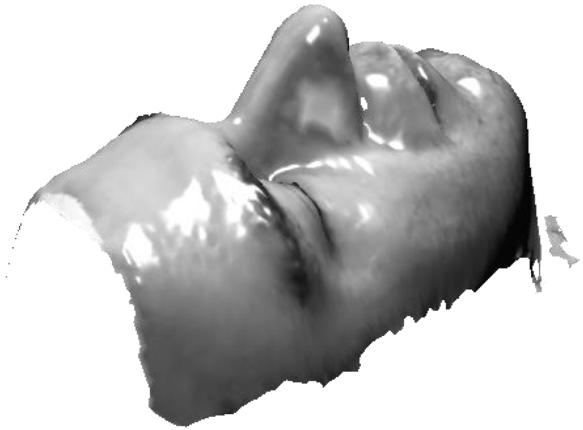
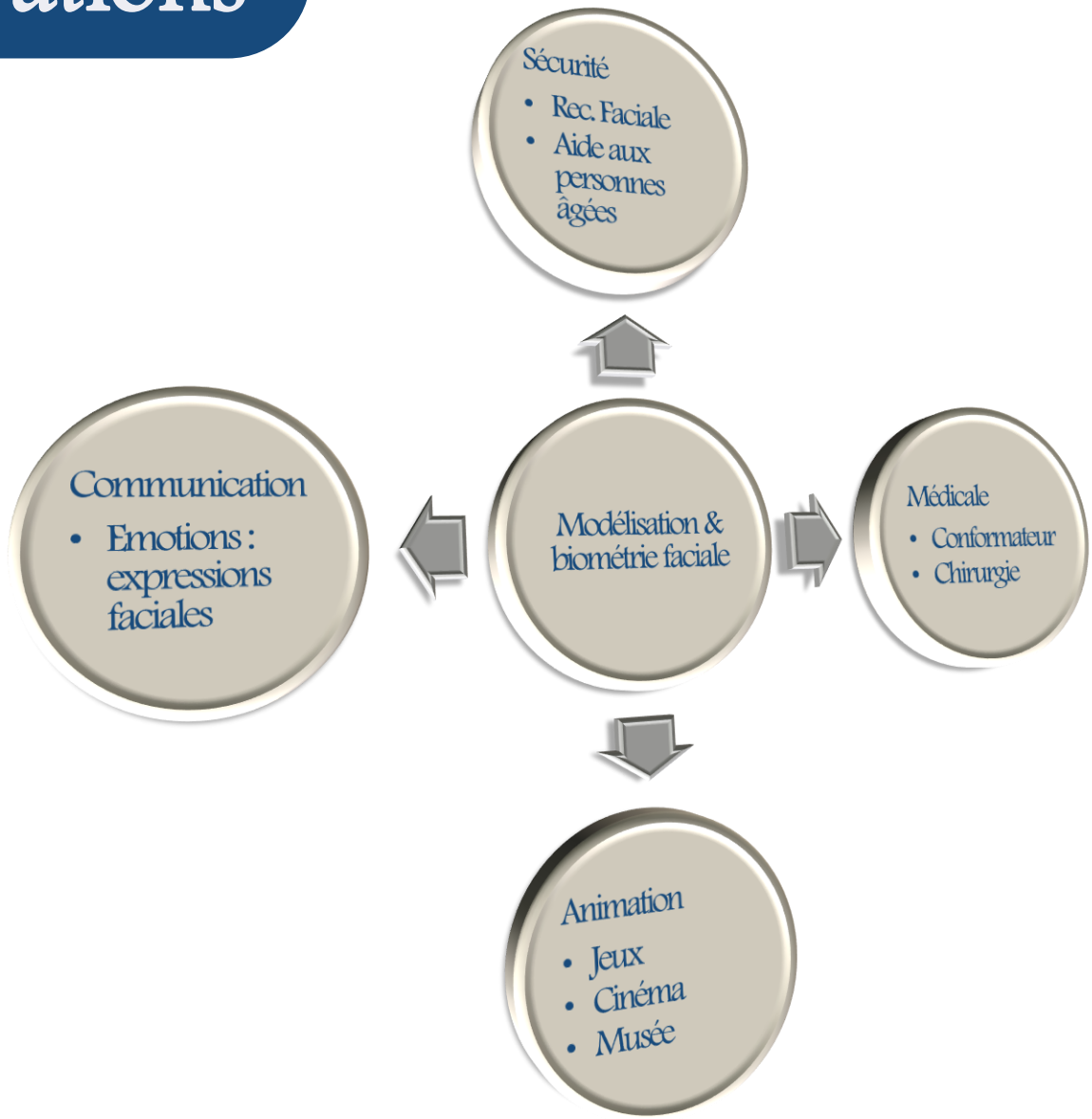
# Range Image Acquisition




**Laboratoire d'InfoRmatique en Image et Systèmes d'information**

LIRIS **UMR 5205** CNRS/INSA de Lyon/Université Claude Bernard Lyon 1/Université Lumière Lyon 2/Ecole Centrale de Lyon
Université Claude Bernard Lyon 1, 36 avenue Guy de Collongue - 69134 Ecully Cedex
http://liris.cnrs.fr

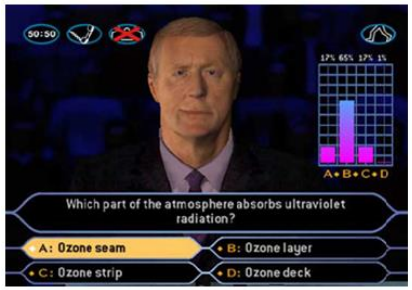
# Motivations

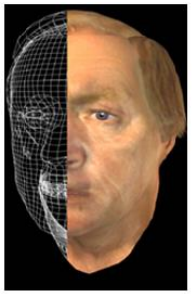


# Motivations

## Biométrie et animation faciale

- Reconnaissance de visages en 3D
- Jeux vidéos
- Cinéma 3D



## Des fins médicales

- Chirurgie esthétique
- Suivi postopératoire

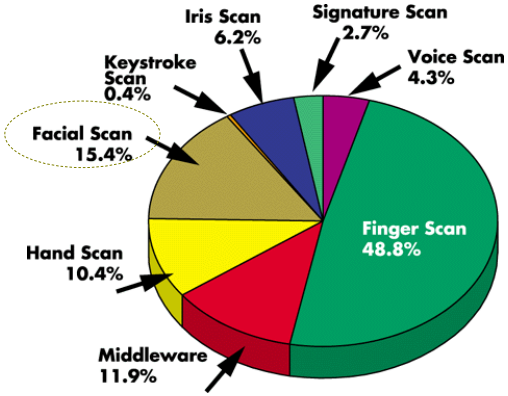
## Mode et beauté

- Coupe de cheveux
- Lunettes de soleil

## Des fins artistiques

# Scénario sécurité

- Lutte contre les fraudes, les vols, les crimes, le terrorisme,…
- Besoin de la sécurité
  - services public : aérogare, transport
  - domaine privé : secteur bancaire
  - secteur professionnel : lieux de travail
  - etc.
- Besoin d'identifier ou d'authentifier
  - ➦ Biométrie
- Pourquoi la modalité faciale ?
  - \+ acquisition peu intrusive et sans contact
  - \+ plus acceptable par les humains
  - \- taux de reconnaissance peu satisfaisant




*Source: International Biometric Group, New York, NY; 1.212.809.9491*

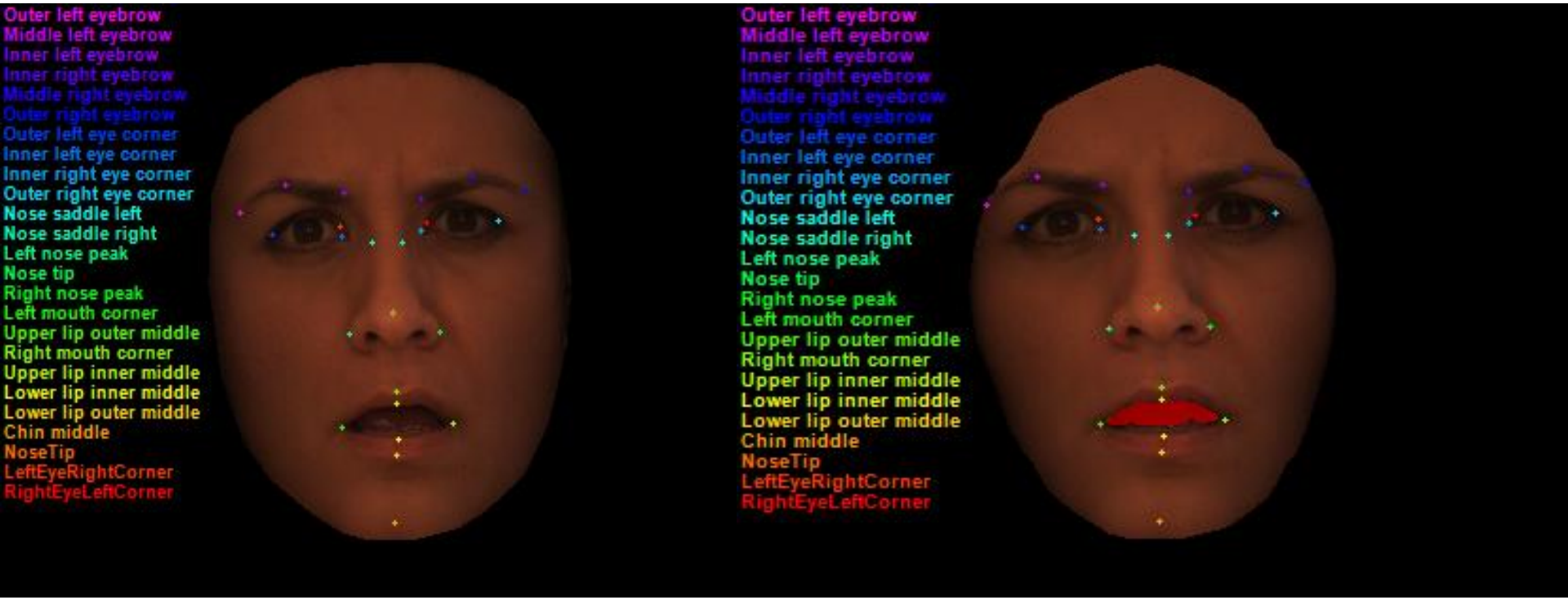
*La ministre de l'intérieur annonce une augmentation de 300% des caméra de vidéosurveillance entre 2007-2010*

# Scénario médical

- Fabrication de prothèse et de conformateur

- Chirurgie esthétique/réparatrice
  - Simulation des résultats, suivi médical, diagnostique
  - Pose d'implants
  - Maxillo-faciale
  - Greffe de visage

# Les expressions faciales



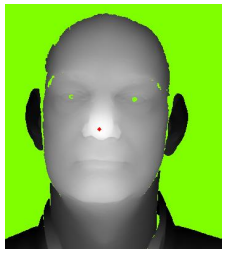
# Synthèse d'images

# Représentation et transformations géométriques

## Représentation des données 3D

- Nuage de points
- Maillage

- Image de profondeur (de disparité ou 2.5 D)



## Transformations sur ces données

- Translation, rotation, changement d'échelle…
- Projections :
  - Perspective, parallèle…
- Notation unifiée ?

# Transformations 2 dimensions

- On commence en 2D
  - Plus facile à représenter
- Chaque point est transformé :
  - $x' = f(x,y)$
  - $y' = g(x,y)$
- Comment représenter la transformation ?

# Translation

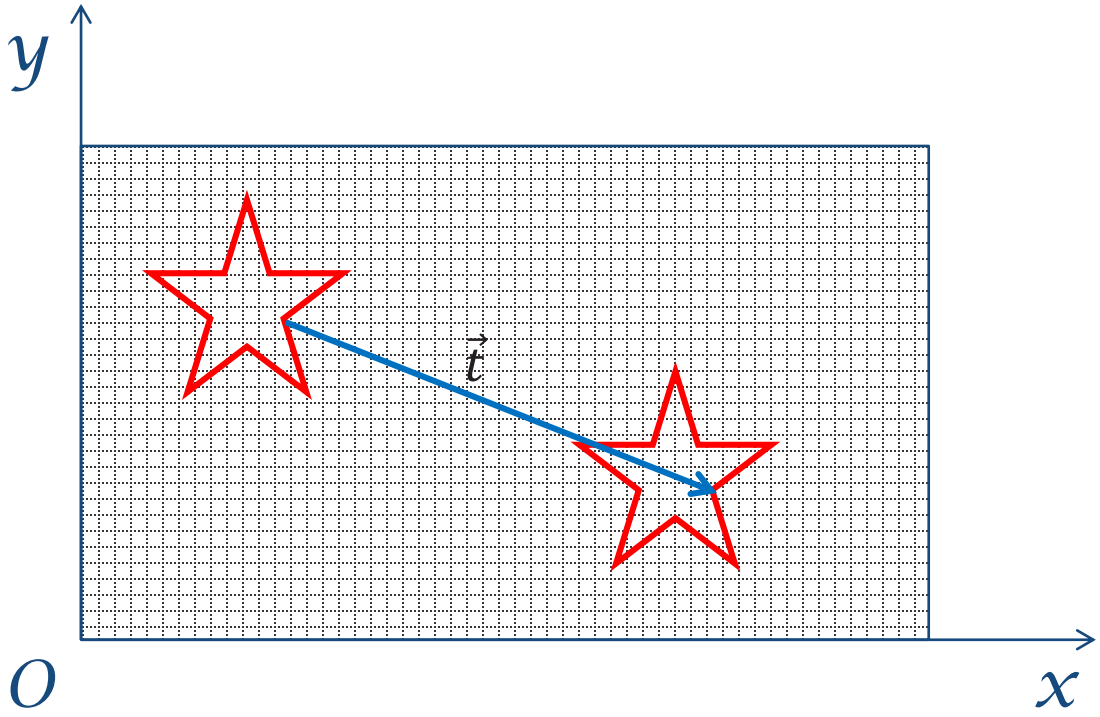
$$x' = x + t\ \ x$$
$$y' = y + t\ \ y$$

$$p = {}^{t}[x\,, y]$$

$$t = {}^{t}[u\,, v]$$

$$p' = {}^{t}[x'\,, y']$$

$$p' = p + t$$




*Addition par le vecteur de translation*

# Changement d'échelle

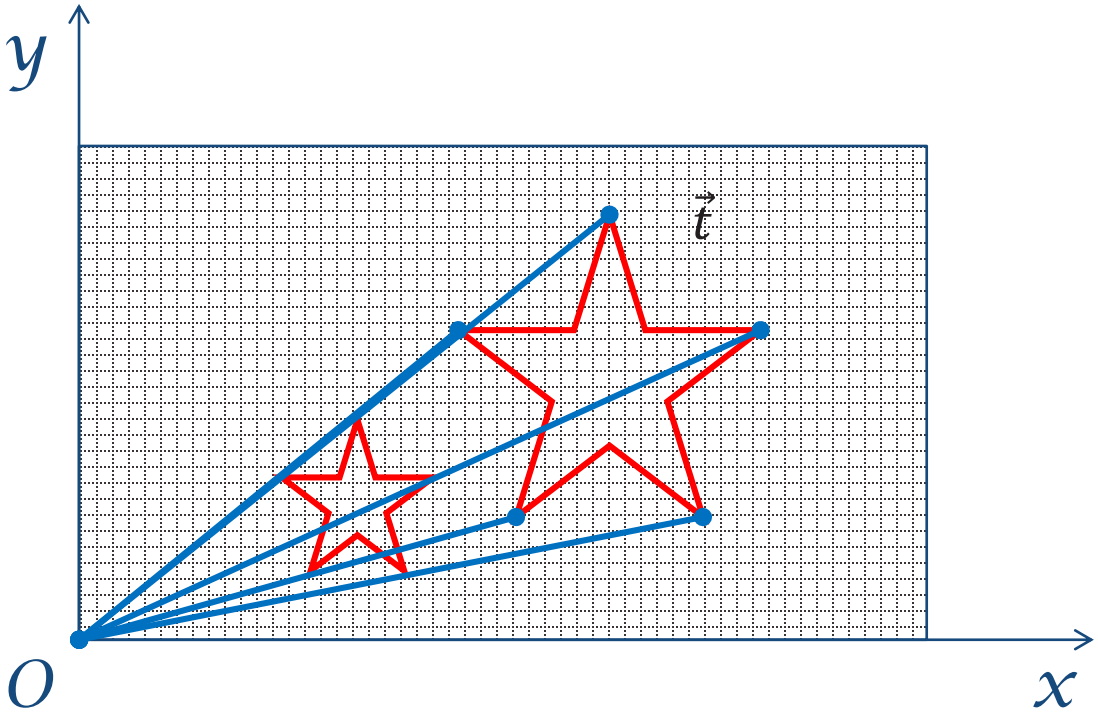
$$x' = s_x\, x$$
$$y' = s_y\, y$$

$$p = {}^t[x\,, y]$$

$$S = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix}$$

$$p' = {}^t[x'\,, y']$$

$$p' = S\, p$$




*Multiplication par le facteur de changement d'échelle*

# Rotation

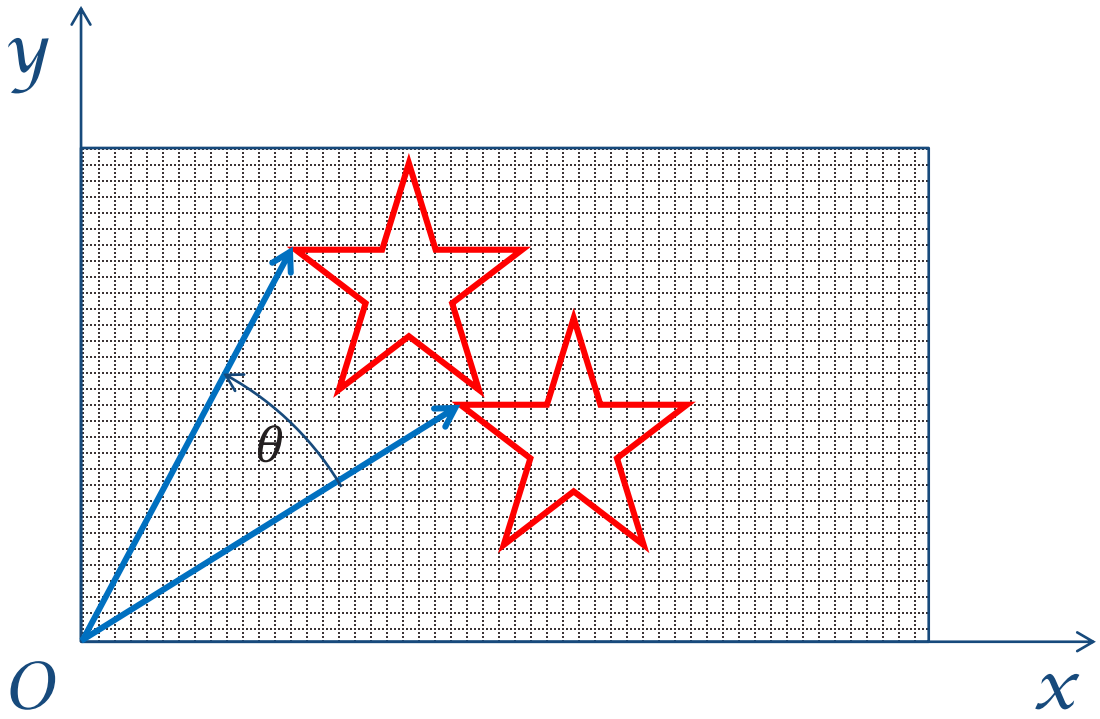
$$x' = \cos\theta\, x - \sin\theta\ \ y$$
$$y' = \sin\theta\, x + \cos\theta\, y$$

$$p = {}^t[x\,, y]$$

$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

$$p' = {}^t[x'\,, y']$$

$$p' = R\, p$$




*Multiplication par la matrice de rotation*

# Coordonnées homogènes

- Outil géométrique puissant :
  - Utilisé vision par ordinateur, synthèse d'image, géométrie projective
- Plan projectif
  - Ajout d'une troisième coordonnée, $w$
- Un point 2D devient un vecteur à 3 coordonnées

$$p = {}^t[x, y, w]$$

- Egalité entre deux points $p$ et $p'$

$$p' \frac{1}{w'} = p \frac{1}{w}$$

$${}^t\left[\frac{x'}{w'}, \frac{y'}{w'}, 1\right] = {}^t\left[\frac{x}{w}, \frac{y}{w}, 1\right]$$

- $w = 0$ : points « à l'infini »
  - Très utile pour les projections, et pour certaines splines

# En 3 dimensions

- Un point 3D devient un vecteur à 4 coordonnées

$$P = {}^t[X, Y\ Z, W]$$

- Egalité entre deux points $P$ et $P'$

$$P' \frac{1}{W'} = P \frac{1}{W}$$

$${}^t\left[\frac{X'}{W'}, \frac{Y'}{W'}, \frac{Z'}{W'}, 1\right] = {}^t\left[\frac{X}{W}, \frac{Y}{W}, \frac{Z}{W}, 1\right]$$

$$\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

- $w = 0$ : points « à l'infini »
- Toutes les transformations sont exprimées sous forme de matrices 4x4

# Translation en c. homogènes

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

$$\begin{cases} \frac{x'}{w'} = \frac{x}{w} + t_x \\ \frac{y'}{w'} = \frac{y}{w} + t_y \end{cases}$$

$$\begin{cases} x' = x + wt_x \\ y' = y + wt_y \\ w' = w \end{cases}$$

# Changement d'échelle

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

$$\begin{cases} x' = s_x\, x \\ y' = s_y\, y \\ w' = w \end{cases}$$

$$\begin{cases} \dfrac{x'}{w'} = s_x \dfrac{x}{w} \\ \dfrac{y'}{w'} = s_y \dfrac{y}{w} \end{cases}$$

# Rotation

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

$$\begin{cases} \dfrac{x'}{w'} = \cos\theta \ \dfrac{x}{w} - \sin\theta \ \dfrac{y}{w} \\ \dfrac{y'}{w'} = \sin\theta \ \dfrac{x}{w} + \cos\theta \ \dfrac{y}{w} \end{cases}$$

$$\begin{cases} x' = \cos\theta \ x - \sin\theta \ y \\ y' = \sin\theta \ x + \cos\theta \ y \\ w' = w \end{cases}$$

# Composition des transformations

- Par multiplication matricielle
  - Une rotation suivie d'une translation :

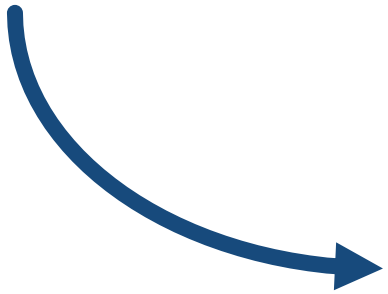
$$M = R\ T$$

- Rotation autour d'un point Q
  - Translater $Q$ à l'origine $(TQ)$,
  - Rotation autour de l'origine $(RQ)$
  - Translater en retour vers $Q$ $(-\ TQ)$

$$p' = \left(-T_Q\right) R_\Theta\ T_Q\ p$$

# Translation en 3D

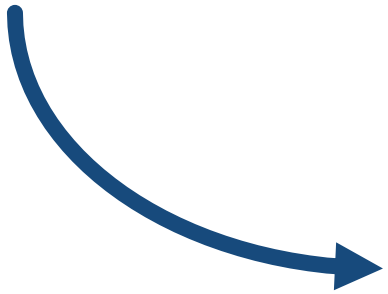
$$T(t_x, t_y, t_z) = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



$$\begin{cases} x' = x + w\, t_x \\ y' = y + w\, t_y \\ z' = z + w\, t_z \\ w' = w \end{cases}$$

# Changement d'échelle en 3D

$$S(s_x, s_y, s_z) = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



$$\begin{cases} x' = x + w\, t_x \\ y' = y + w\, t_y \\ z' = z + w\, t_z \\ w' = w \end{cases}$$

# Rotation en 3D

- Définie par un axe et un angle
- La matrice dépend de l'axe et de l'angle
- Expression directe possible, en partant de l'axe et de l'angle, et quelques produits vectoriels
  - Passage par les quaternions
- Fait par la librairie graphique :
  - glRotatef(angle, x, y, z)

# Quaternion

- Soit la matrice de rotation $R$. Il existe quatre nombres $\{q_1, q_2, q_3, q_4\}$ tels que $q_0^2+q_1^2+q_2^2+q_3^2=0$, et

$$R = \begin{pmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1 q_2 - q_0 q_3) & 2(q_1 q_3 - q_0 q_2) \\ 2(q_2 q_1 + q_0 q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2 q_3 + q_0 q_1) \\ 2(q_3 q_1 - q_0 q_2) & 2(q_3 q_2 + q_0 q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{pmatrix}$$

# Quaternion

La représentation quaternion $\{q_1, q_2, q_3, q_4\}$ d'une rotation autour d'un axe $l$ et d' un angle $\Omega$, $0 \leq \Omega \leq \pi$, est donnée par

$$q_0 = \varepsilon \cos\frac{\Omega}{2}, \qquad \begin{pmatrix} q_1 \\ q_2 \\ q_3 \end{pmatrix} = \varepsilon\, l \sin\frac{\Omega}{2}, \qquad \varepsilon = \pm 1$$

$$\Omega = 2\cos^{-1} q_0, \qquad l = \frac{\begin{pmatrix} q_1 \\ q_2 \\ q_3 \end{pmatrix}}{\left\| \begin{pmatrix} q_1 \\ q_2 \\ q_3 \end{pmatrix} \right\|}, \qquad si\ q_0 \geq 0,$$

.

$$\Omega = 2(\pi - \cos^{-1} q_0), \qquad l = -\frac{\begin{pmatrix} q_1 \\ q_2 \\ q_3 \end{pmatrix}}{\left\| \begin{pmatrix} q_1 \\ q_2 \\ q_3 \end{pmatrix} \right\|}, \qquad si\ q_0 < 0.$$

# Prérequis

- Soit la matrice de corrélation *K*. Soit *K'* la matrice symétrique de dimension quatre, définie à partir de *K* de la manière suivante

$$K' = \begin{pmatrix} K_{11}+K_{22}+K_{33} & K_{32}-K_{23} & K_{13}-K_{31} & K_{21}-K_{12} \\ K_{32}-K_{23} & K_{11}-K_{22}-K_{33} & K_{12}+K_{21} & K_{31}+K_{13} \\ K_{13}-K_{31} & K_{12}+K_{21} & -K_{11}+K_{22}-K_{33} & K_{23}+K_{32} \\ K_{21}-K_{12} & K_{31}+K_{13} & K_{23}+K_{32} & -K_{11}-K_{22}+K_{33} \end{pmatrix}$$

- Soit *q'* le vecteur propre unité de dimension quatre de la matrice *K'*, associé à la plus grande valeur propre
- Alors, *trace*($^tR\,K$) est maximisée par la matrice de rotation *R* définie par *q'*
- La solution est unique si la plus grande valeur propre est une racine simple du polynôme caractéristique

# Décomposition de la matrice épipolaire

- Calculer $h$, le vecteur propre unité de la matrice ${}^tM\,M$ pour la plus petite valeur propre
- Poser $K = \sim h \times M$, et par la méthode de représentation quaternion calculer la matrice de rotation $R$ qui maximise :

$$\text{trace}({}^tR\,K)$$

- Retourner les paramètres du mouvement ($l$, $\Omega$, $h$)
- Calcul du résidu de l'équation épipolaire

# Toutes les transformations 3D

- Toute transformation 3D s'exprime comme combinaison de translations, rotations, changement d'échelle
  - Et donc comme une matrice en coordonnées homogènes
- Fournies par la librairie graphique :
  - glTranslatef(x, y, z);
  - glRotatef(angle, x, y, z);
  - glScalef(x, y, z);

# Transformations 3D (suite)

- On peut faire ses transformations soi-même :
  - glLoadIdentity();
    - Remplace la matrice de transformation courante par la matrice identité
  - glLoadMatrixf(pm);
    - Remplace la matrice de transformation courante par la matrice pm exprimée en floats
  - glMultMatrixf(pm);
    - Multiplie la matrice de transformation courante par la matrice pm exprimée en floats
- Pile de transformations :
  - glPushMatrix();
    - Pousse la matrice courante dans le stack et la duplique. Après l'appel à cette fonction la matrice au dessus du stack est dupliquée.
  - glPopMatrix();
    - Déplie la matrice courante.

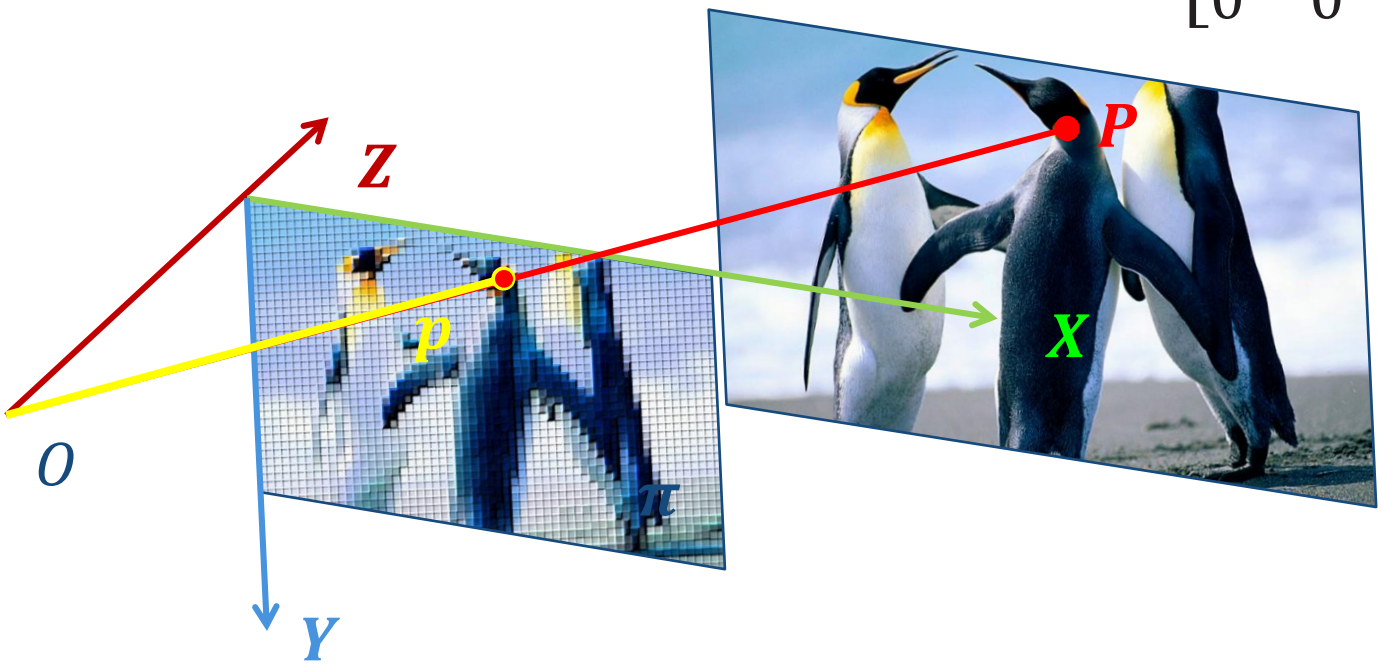
# Exemple

```
drawHighLevelObject(parameters) {
                                    glPushMatrix()
                                      glRotate(…)
                                     glTranslate(…)
                                      glScale(…)
                                    drawSimpleShape()
  gllPopMatrix()
}
drawModel()                                                          {
                                    glPushMatrix()
                                  drawHighLevelObject1(…)
                                    glTranslate(…)
                                  drawHighLevelObject2(…)
                                       [etc…]
  gllPopMatrix()
}
```

# Supplément : projection perspective

- Projection sur le plan z = 0, avec le centre de projection placé à z = – d

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{d} & 1 \end{bmatrix}$$



# Supplément : perspective (suite)

- Projection perspective s'appuie sur coordonnées homogènes
- La rétrécissement des objets utilise w

$$W' = \frac{Z}{d} + W$$

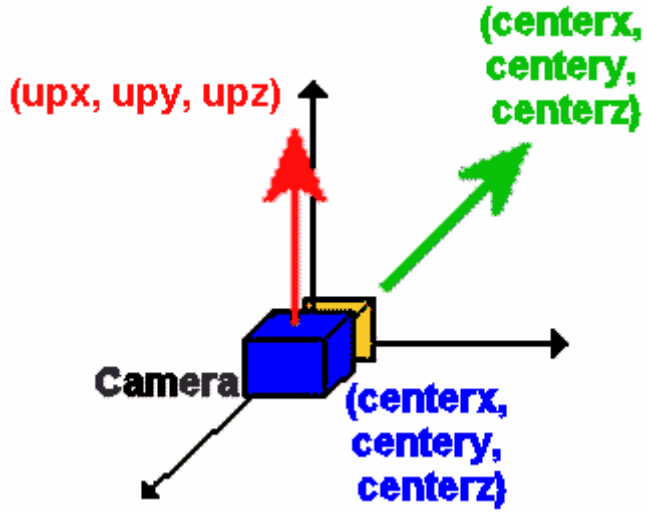
$$\frac{X'}{W'} = \frac{X}{\frac{Z}{d} + W}$$

$$\frac{Y'}{W'} = \frac{Y}{\frac{Z}{d} + W}$$

# Perspective en pratique

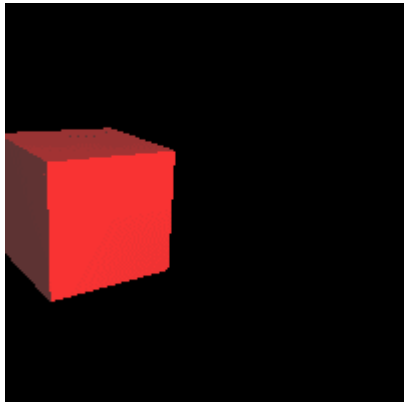
*void gluLookAt(GLdouble eyex, GLdouble eyey, GLdouble eyez, GLdouble centerx, GLdouble centery, GLdouble centerz, GLdouble upx, GLdouble upy, GLdouble upz);*

- Définit la position et l'orientation de la caméra.
- (eyex, eyey, eyez) spécifies la position de la caméra,
- (centerx, centery, centerz) spécifies la position vers laquelle la caméra pointe (aussi appelé la ligne de mire)
- (upx, upy, upz) spécifies un vecteur qui nous informe de la direction du haut de la caméra
- Cette commande est contenu dans la GLU (OpenGL Utility Library).
- Inclure le header (fichier d'en-tête) suivant : glu.h

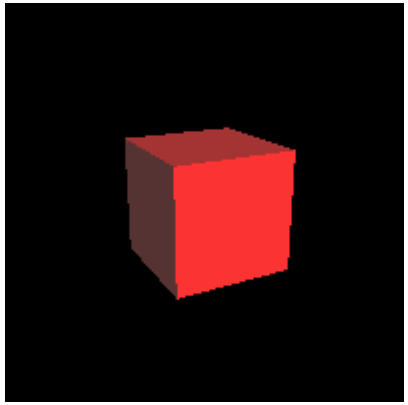


# Perspective : en pratique

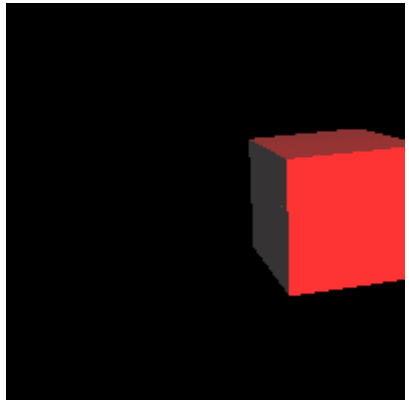
```
...
glLoadIdentity(); // Restaure la matrice de modélisation-visualisation
gluLookAt(0.0, 0.0, 4.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0); // Positionne et
   oriente la caméra Dessin d'un cube
...
```



*gluLookAt(0, 0, 4, 1, 0, 0, 0, 1, 0)*



*gluLookAt(0, 0, 4, 0, 0, 0, 0, 1, 0)*



*gluLookAt(0, 0, 4, -1, 0, 0, 0, 1, 0)*

# Perspective : en pratique

*void gluPerspective(GLdouble fovy, GLdouble aspect,*
*GLdouble zNear, GLdouble zFar);*

- Assure la projection en perspective.
- fovy spécifie l'angle du champs de vision, en degrés, dans la direction de y,
- aspect spécifie l'aspect ratio qui détermine le champs de vision dans la direction de x. aspect ratio est le rapport de x (largeur) sur y (hauteur) ,
- zNear spécifie la distance de l'observateur au plus proche plan (toujours positif),
- zFar spécifie la distance de l'observateur au plus loin plan (toujours positif),
- Inclure le header (fichier d'en-tête) suivant : glu.h.

$$f = \cot angent\left(\frac{fovy}{2}\right)$$

$$\begin{pmatrix} \frac{f}{aspect} & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & \frac{zFar + zNear}{zFar - zNear} & \frac{2 \times zFar \times zNear}{zFar - zNear} \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

# Géométrie épipolaire


Laboratoire d'InfoRmatique en Image et Systèmes d'information

LIRIS **UMR 5205** CNRS/INSA de Lyon/Université Claude Bernard Lyon 1/Université Lumière Lyon 2/Ecole Centrale de Lyon
Université Claude Bernard Lyon 1, 36 avenue Guy de Collongue - 69134 Ecully Cedex
http://liris.cnrs.fr

# Questions

***(i)*** ***Géométrie d'appariement :*** *Soit un point p dans une première image, comment il contraint la position de son correspondant p' dans une seconde image ?*

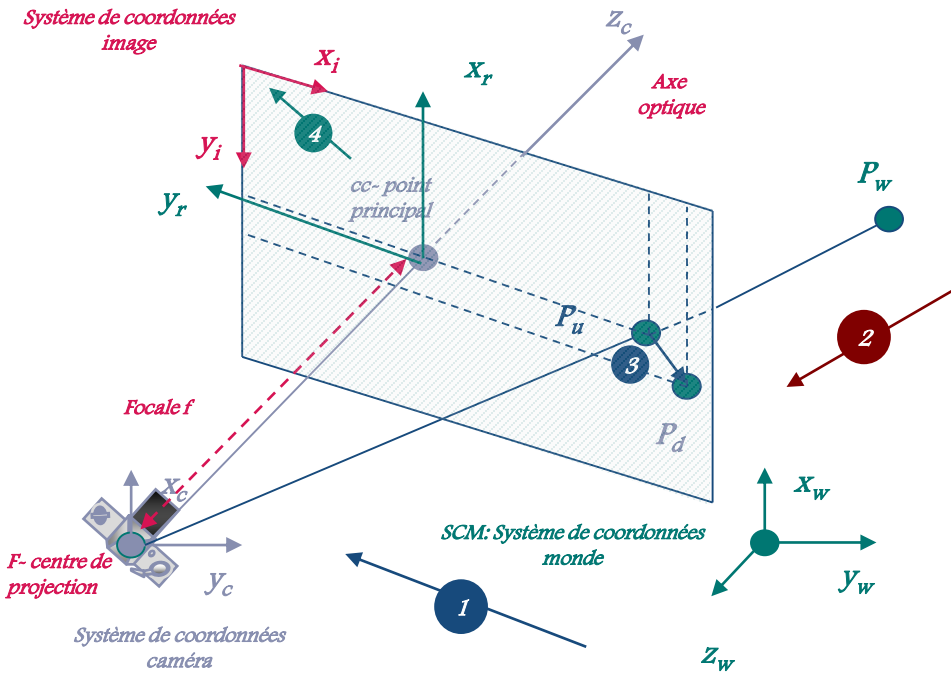
***(ii)*** ***Géométrie de caméra (mouvement) :*** *Etant donné un ensemble de points appariés* $\{p_i \leftrightarrow p_i' \}, i = 1, \dots n$, *quelle est la relation entre les deux caméras et la transformation géométrique entre les deux images ?*

***(iii)*** ***Géométrie de la scène (structure) :*** *Etant donnés un ensemble de points appariés* $\{p_i \leftrightarrow p_i' \}, i = 1, \dots n$, *et caméras* $C_1$ *et* $C_2$, *quelles sont les positions des points* $P_i$ *dans l'espace ?*

# Transformations Monde/image

Une caméra est modélisée par plusieurs paramètres, matrice de projection M :

- La Translation T du centre optique (de projection F) par rapport à l'origine de SCM
- La Rotation R du plan image
- Focale f, Point principal $(x'_c, y'_c)$, taille d'un pixel $(s_x, s_y)$
- Les paramètres "extrinsèques," et "intrinsèque"




$$X = \begin{bmatrix} sx \\ sy \\ s \end{bmatrix} = \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = MX$$

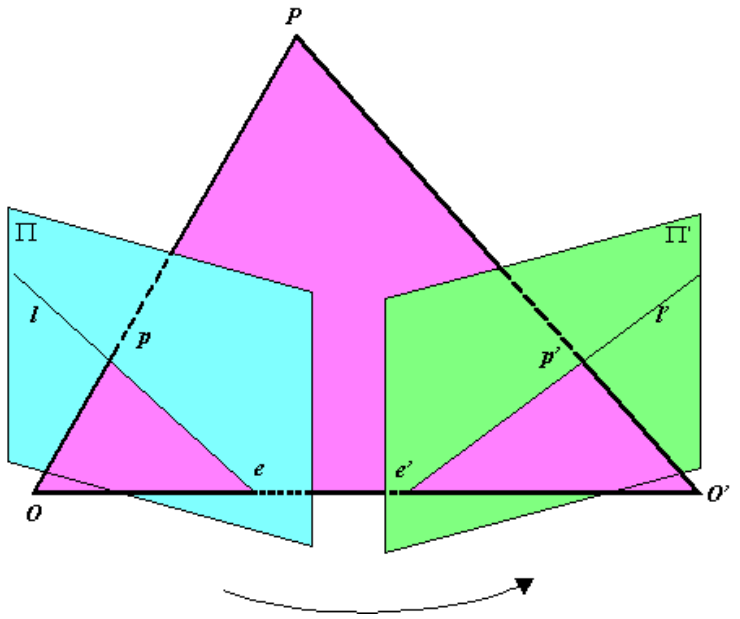
$$M = \underbrace{\begin{bmatrix} -fs_x & 0 & x'_c \\ 0 & -fs_y & y'_c \\ 0 & 0 & 1 \end{bmatrix}}_{4\ \text{intrinsics}} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{2\ \text{projection}} \underbrace{\begin{bmatrix} R_{3x3} & 0_{3x1} \\ 0_{1x3} & 1 \end{bmatrix}}_{1\ \text{rotation}} \underbrace{\begin{bmatrix} I_{3x3} & T_{3x1} \\ 0_{1x3} & 1 \end{bmatrix}}_{1\ \text{translation}}$$

# Le modèle projectif et la stéréovision

$$\underbrace{\begin{bmatrix} -fs_x & 0 & x'_c \\ 0 & -fs_y & y'_c \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{M}\underbrace{\begin{bmatrix} R_{3x3} & 0_{3x1} \\ 0_{1x3} & 1 \end{bmatrix}\begin{bmatrix} I_{3x3} & T_{3x1} \\ 0_{1x3} & 1 \end{bmatrix}}_{E} = {}^{t}M' E$$

*Intrinsics* X *Projection* — *Extrinsic*

- *Les matrices fondamentales et essentielles : F, E*
- $F = {}^{t}M' E M$
- Centre de projection *O* et *O'*
- Le point *P* de la scène
- Les plans de projection : deux images $\pi$ et $\pi'$
- Les points *p* et *p'* obtenus par les transformations projectives
- ${}^{t}p' F p = 0$, ${}^{t}p' {}^{t}M' E M p = 0$
- *E* contient le vecteur de translation h et la matrice de rotation *R*
- ${}^{t}p' F = \vec{p}$
- *Connaissance de E , M et M' → Etalonnage fort*
- *Connaissance F→ Etalonnage faible*




*F*

# Géométrie épipolaire

P : point 3D de l'espace

O, O' : centres optiques

Π, Π' : plans d'images

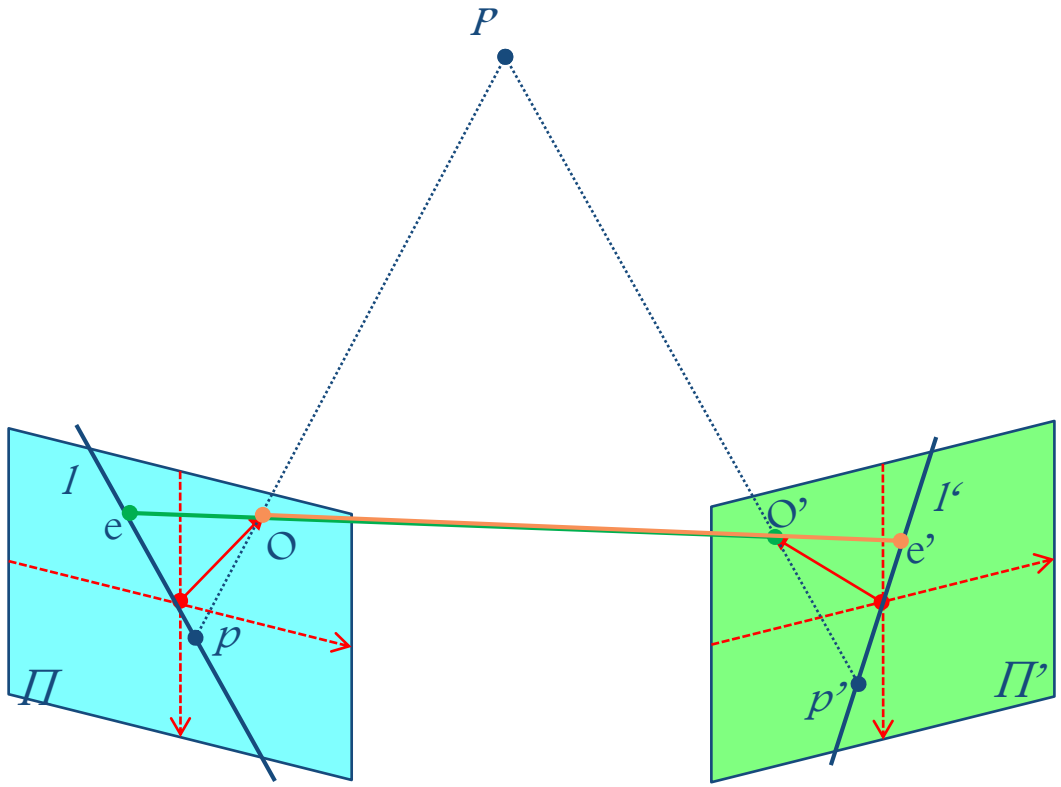
p, p' : projetés de P

e, e' : les épipoles

- e = image de O' dans Π
- e' = image de O dans Π'

l, l' : droites épipolaires

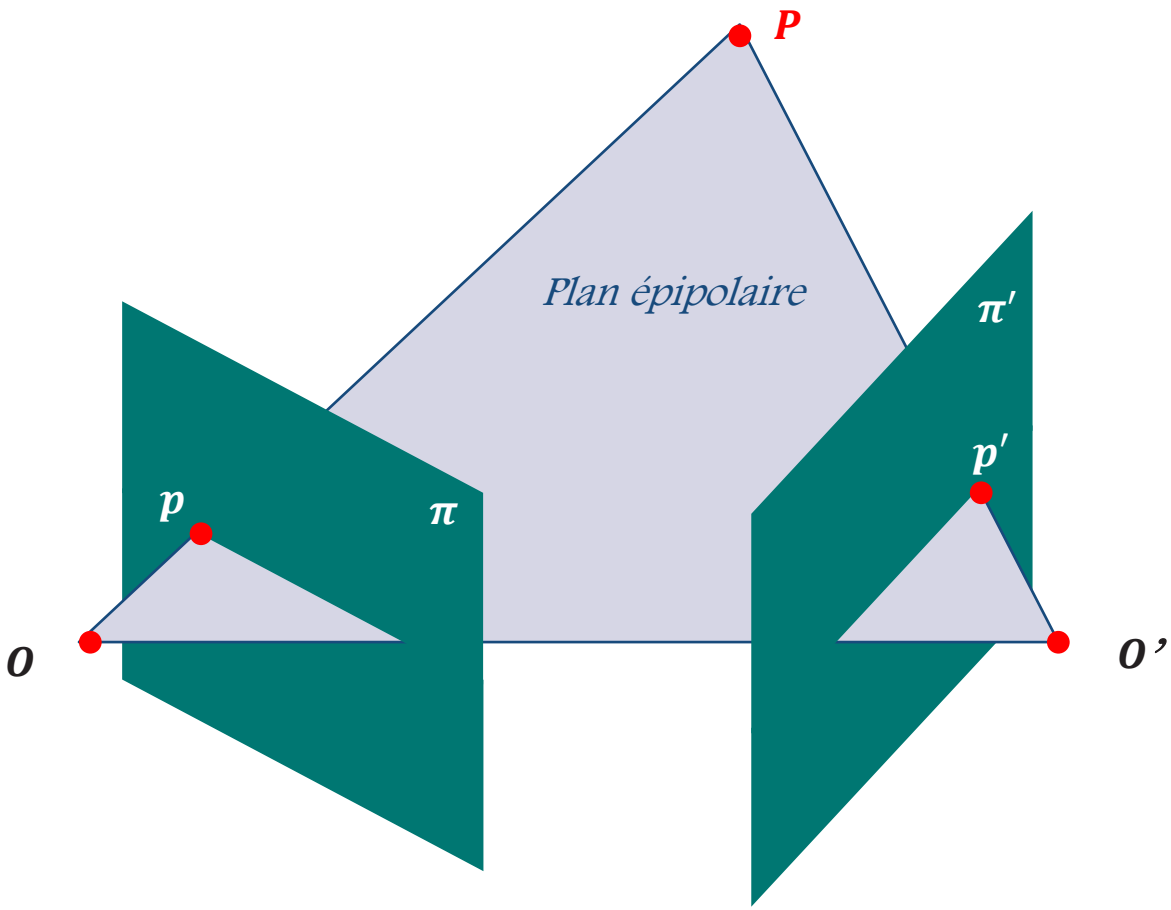
- l = intersection du plan OPO' avec l'image Π
- l' = intersection du plan OPO' avec l'image Π'




*L'image p' d'un point p de l'image gauche est la droite l' dans l'image droite*

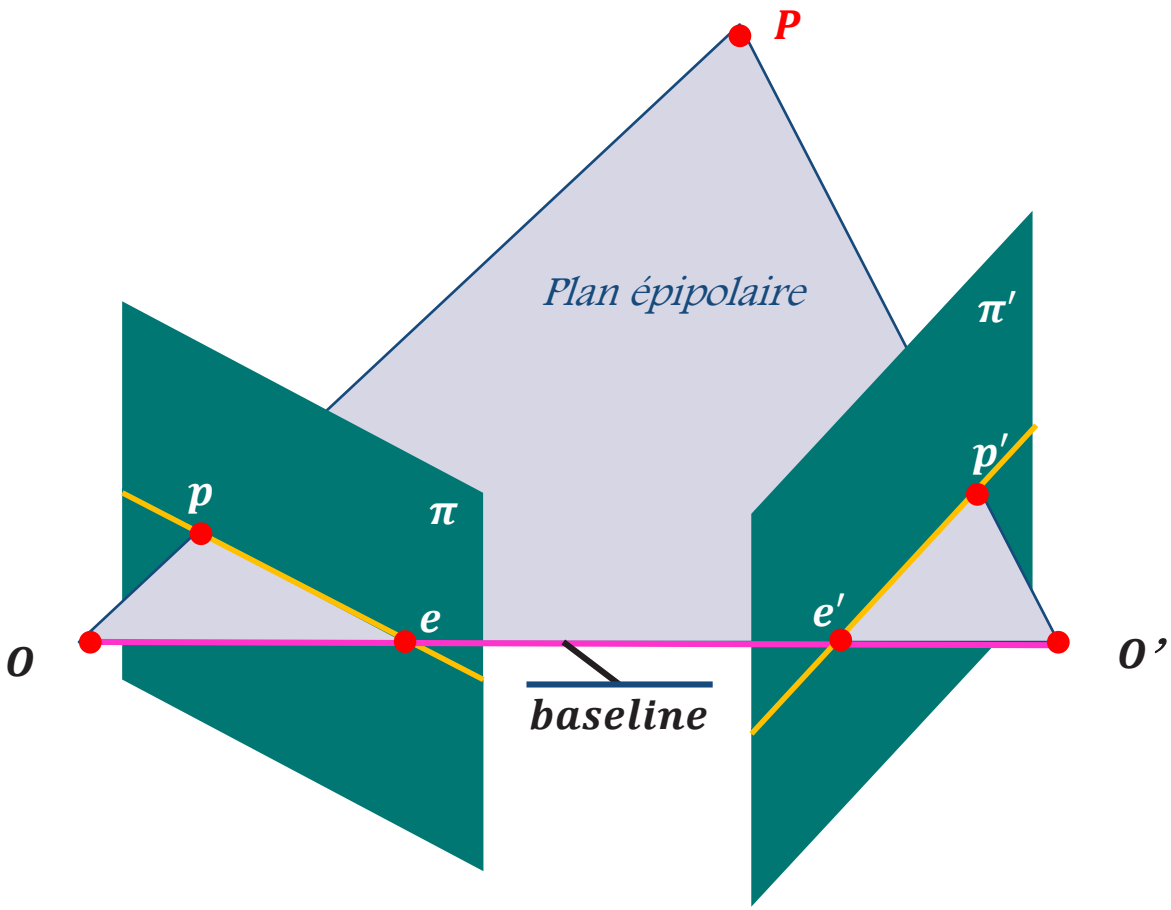
→ *nécessité d'appariement : où est trouve le point p' sur la droite l'*

# La géométrie épipolaire



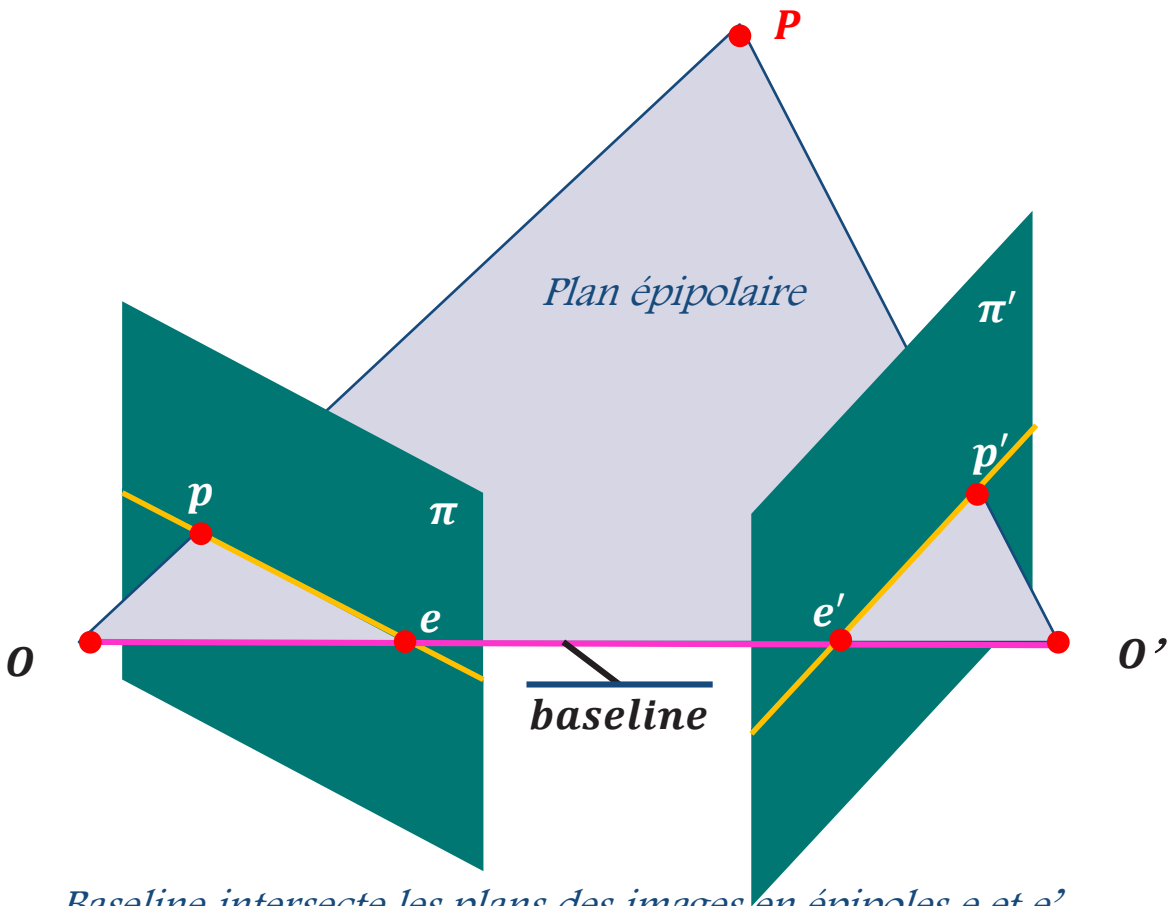

*Les caméras,O et O’, avec P et ses projetés p et p’ sont coplanaires.*

# La géométrie épipolaire



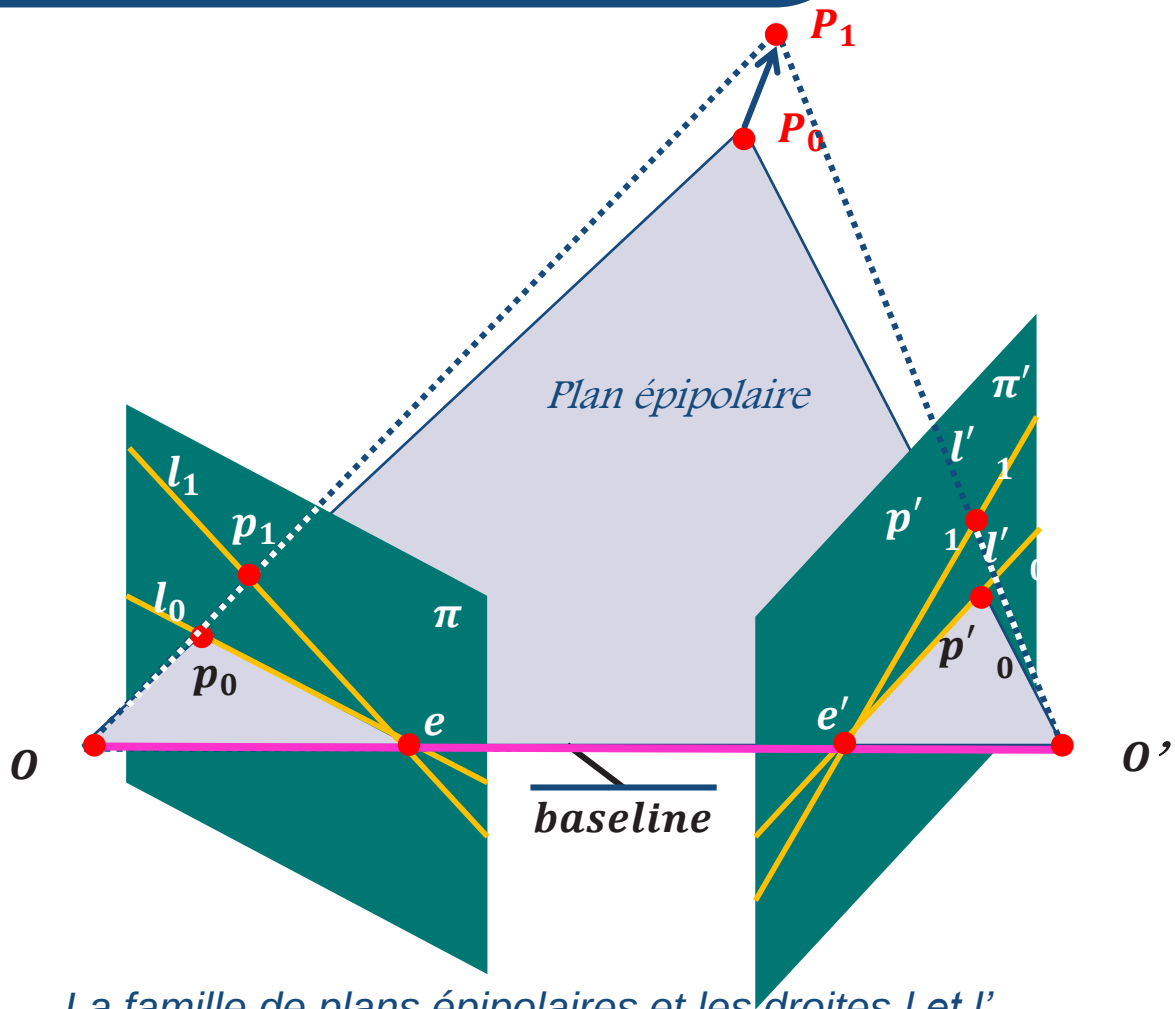

*Les caméras,O et O’, avec P et ses projetés p et p’ sont coplanaires.*
*Tous les points du plan épipolaires sont projetés sur l et l’.*

# La géométrie épipolaire




*Baseline intersecte les plans des images en épipoles e et e’.*
*Tous plancontenant baseline est un plan épipolaire.*
*Tous les points d’un plan épipolaires sont projetés sur l et l’.*

# La géométrie épipolaire




*La famille de plans épipolaires et les droites l et l' s'intersectent en e et e'.*

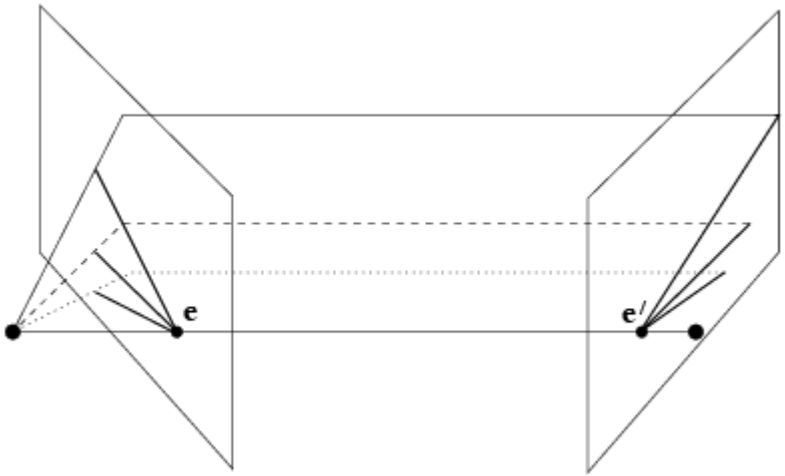
# La géométrie épipolaire

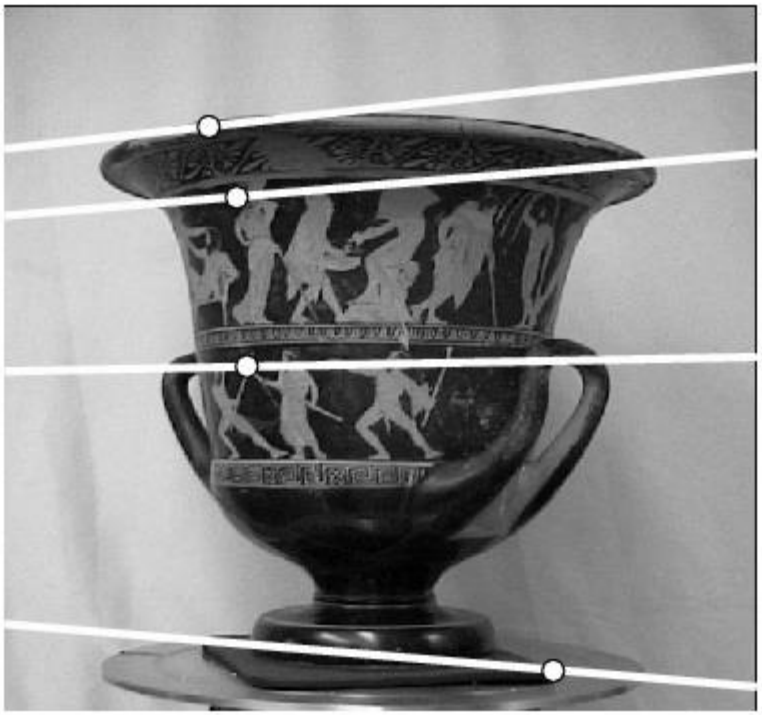
*Epipoles e et e'*
*= intersection de la baseline avec les plans des images*
*= projection de chaque centre optique sur le plan d'image conjugué*
*= point de fuite de la caméra dans la direction du mouvement*

*Un plan épipolaire = plan contenant la baseline (famille 1-D)*

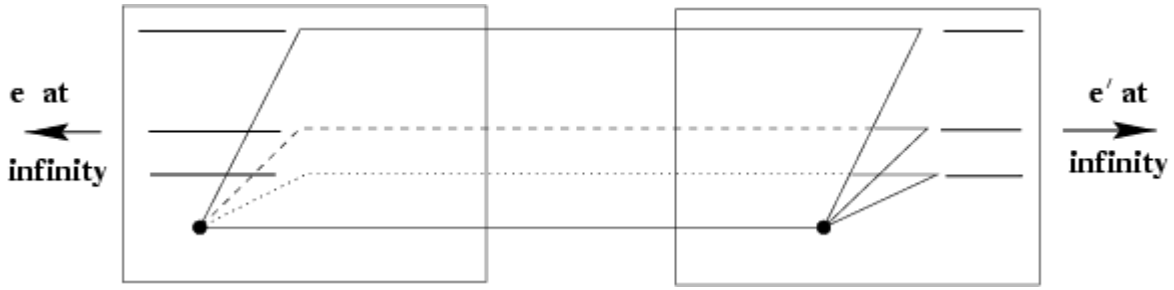
*Une ligne épipolaire = intersection des plans épipolaires avec image de manière conjuguée*

# Exemple de caméras convergentes

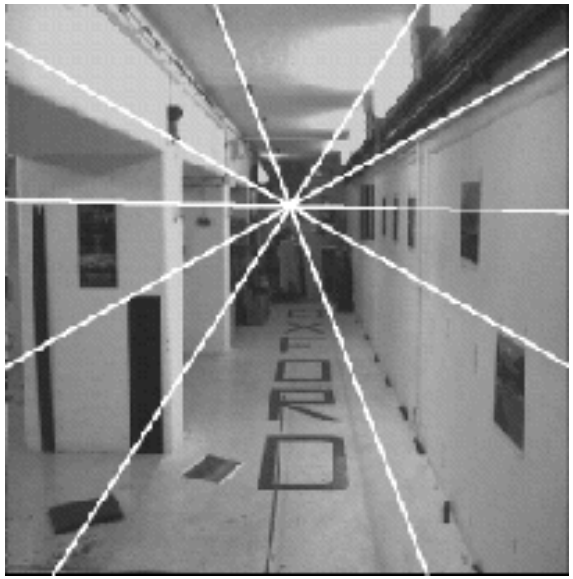


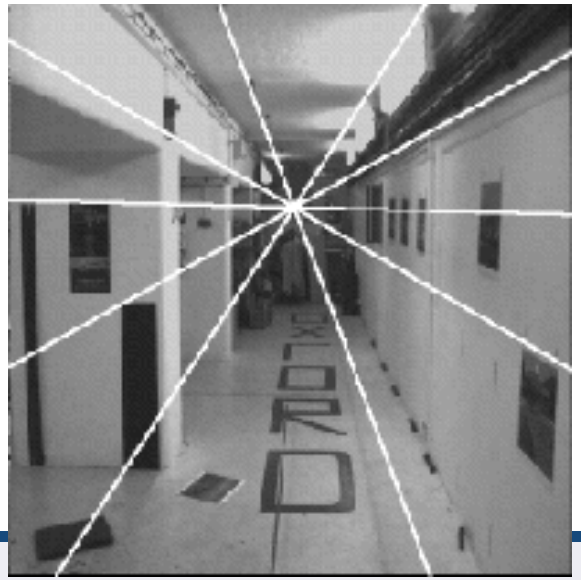



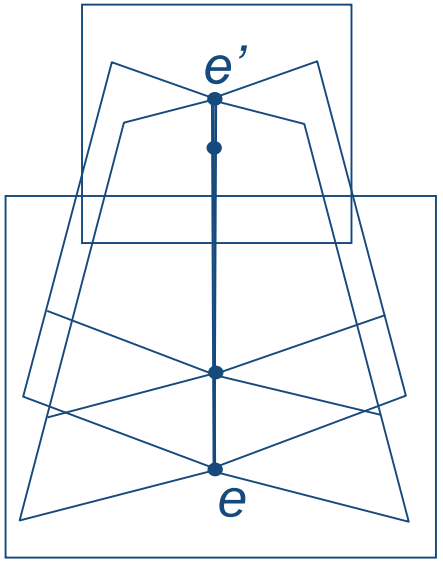
# Exemple de mouvement // à la caméra




# Exemple de mouvement en profondeur







# Définition de produit vectoriel

$$a \times b = \begin{bmatrix} a_2 b_3 - a_3 b_2 \\ a_3 b_1 - a_1 b_3 \\ a_1 b_2 - a_2 b_1 \end{bmatrix} = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} b = \quad [a_\times] b$$

$$a \cdot (a \times b) = 0$$
$$b \cdot (a \times b) = 0$$

# Propriétés des matrices fondamentale et essentielle

- Matrice 3 x 3
- Transposition : si F est la matrice essentielle de la paire de caméras (C, C'), ${}^{t}F$ est la matrice essentielle de (C',C).

- Lignes épipolaires : $p$ et $p'$ deux points dans les plans projectifs, alors $Fp$ est une droite projective dans l'image droite.

$$l'=Fp, \qquad l={}^{t}Fp'$$

- Epipoles : pour tout p la droite épipolaire $l'=F\,p$ contient l'épipole $e'$. Donc $({}^{t}e'F)\,p=0$ pour tout $p$.

  Donc ${}^{t}e'F=0$ et $Fe=0$

# Matrice fondamentale

- Contient les paramètres intrinseques et extrainseques
- F est de rang 2
- F a 7 degrées de liberté
- F a neuf éléments à une échèlle près

$$\det(F) = 0$$

# Matrice essentielle

- Contient uniquement les paramètres extrinseques
- E est de rang 2
- Ces deux valeurs singulières sont égales
- A 5 degrés de liberté, 3 pour la rotation et 2 pour la translation

# Ambiguité de taille

$$P' = R\,P + t$$

$$p = \frac{P}{{}^{t}\hat{Z}\,P} \qquad p' = \frac{RP+t}{{}^{t}\hat{Z}\,(RP+t)}$$

*Les profondeurs Z et Z' et t sont obtenus à un facteur d'échèlle près*
*Seulement la direction de la translation peut être calculée*

# Les moindres carrés

$$Min \sum_{i=1}^{n} \left( p'_i \;\; F \; p_i \right)^2$$

$$|F|^2 = 1$$

*Un système homogène A f=0*
*La solution au sens moindres carrés est la plus petites valeur singulière de A:*
*La dernière colonne de V telle que A=U D $^t$V par SVD*

# Estimation de la matrice fondamentale à partir de points appariés

$$ {}^{t}p'_i \; F \; p_i = 0 $$

*Pour* $p = (x, y, 1)$ *et* $p' = (x', y', 1)$ *nous avons*

$$ x'xf_{11} + x'yf_{12} + x'f_{13} + y'xf_{21} + y'yf_{22} + y'f_{23} + xf_{31} + yf_{32} + f_{33} = 0 $$

*Pour n paires de points appariés nous avons*

$$ Af = \begin{bmatrix} x'_1 x_1 & x'_1 y_1 & x'_1 & y'_1 x_1 & y'_1 y_1 & y'_1 & x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x'_n x_n & x'_n y_n & x'_n & y'_n x_n & y'_n y_n & y'_n & x_n & y_n & 1 \end{bmatrix} f = 0 $$

# Algorithme de 8 points : 8 points algorithm

*Un système d'équations homogène* $Af = 0$

*f peut être déterminée à un facteur près, possède donc 8 inconnus donc son estimation nécessite 8 points appariés*

***8 points algorithm***

*La solution au sens des moindres carrés est le vecteur singulier correspondant à la plus petite valeur singulière de A*

*La dernière colonne de* $V$ *pour la SVD* $A = U\ D\ {}^tV$

# La solution par approche non linéaire

$$\min \sum_{i=0}^{n} \left( \left\| p_i' F \; p_i \right\|^2 + \left\| p_i \quad F \; p_i' \right\|^2 \right)$$

# Localiser les épipoles

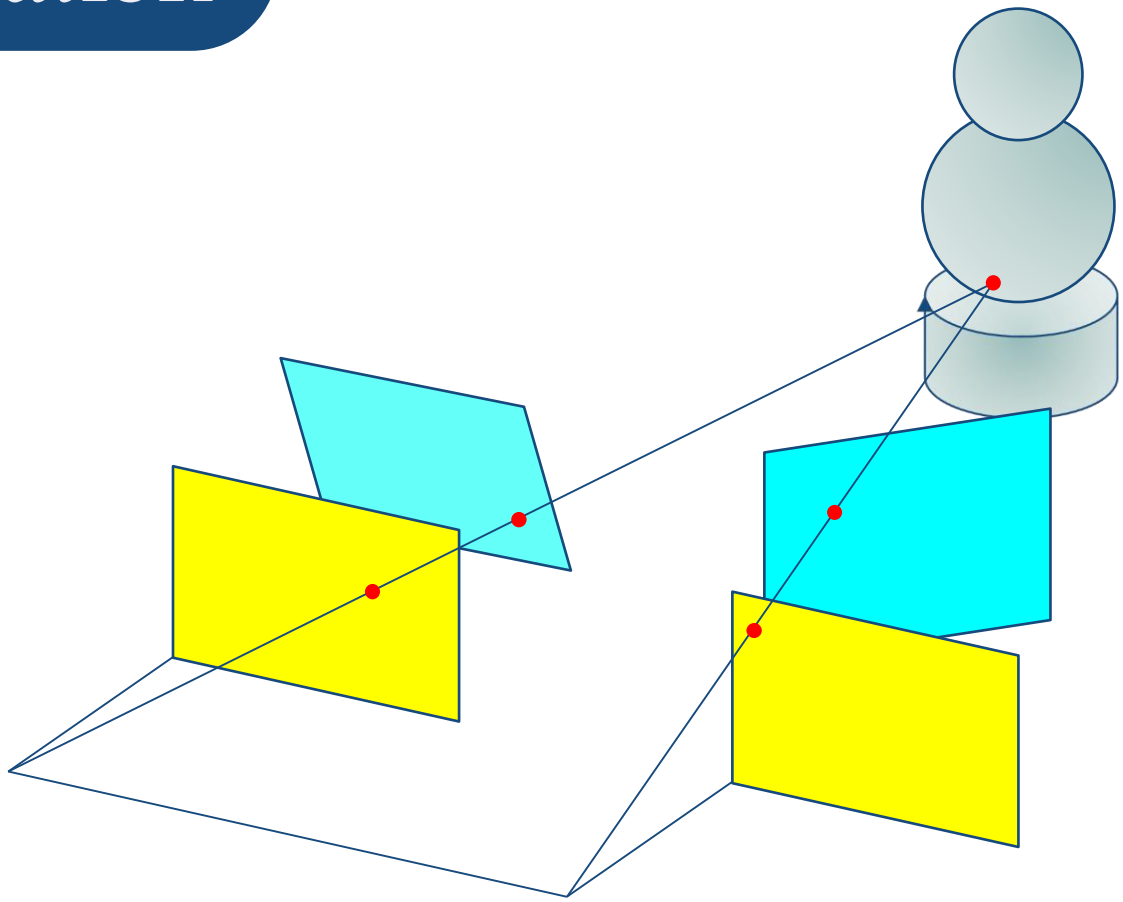
$$^{t}p' \; F \; e = 0$$

$$F \; e \; = \; 0$$

*e est le noyau de F et e′ est le noyau de* $^{t}F$

$$SVD \;\; de \, F = U \;\; D \;\; ^{t}V$$

# Réctification



*Reprojection de l'image sur un plan paralèlle à la droite passant par les centres optiques*

# Réctification

- Tourner la caméra gauche de manière à ce que l'épipole se situent à l'infini sur l'axe x
- Idem pour la caméra droite
- Annuler la rotation
- Ajuster l'échèlle

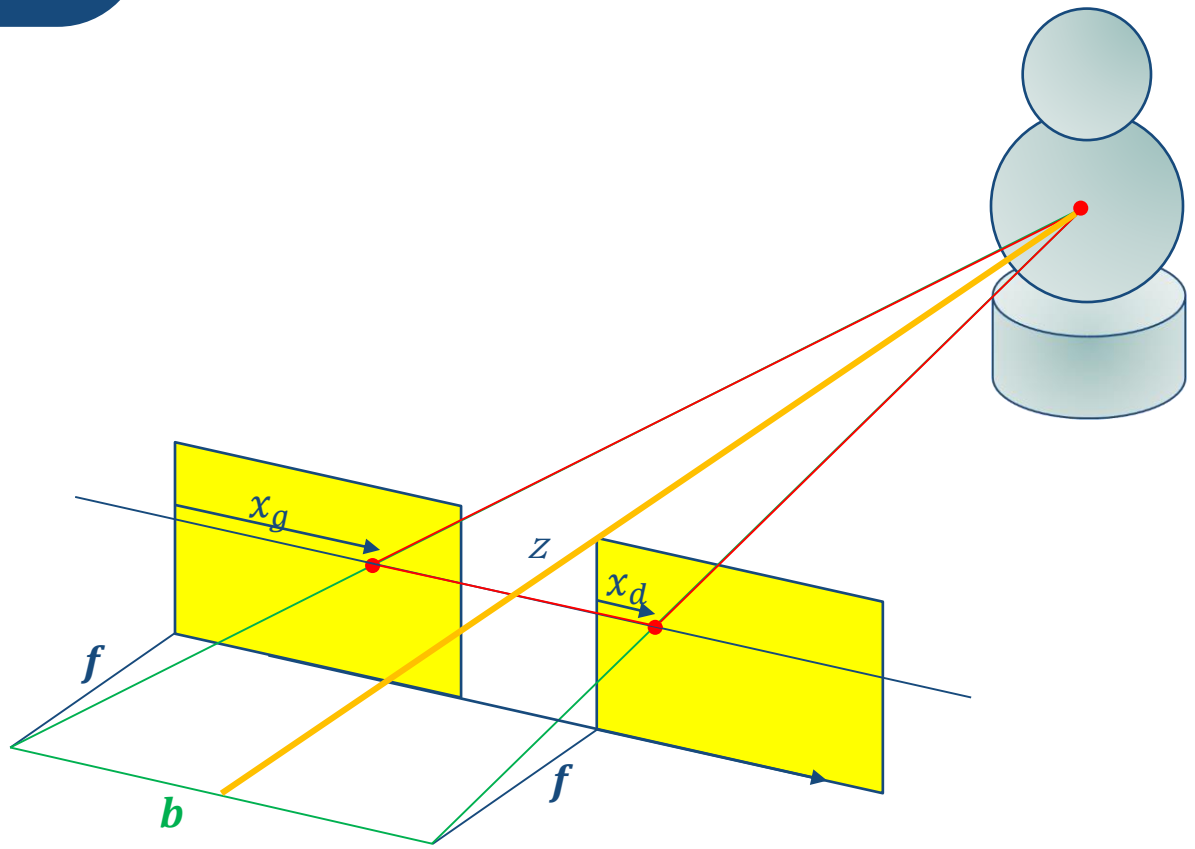
# Reconstruction 3D

- Stéréo : connaissant la géométrie entre les caméras (paramètres extrinseques) et la géométrie des caméras (paramètres insrinseques), appariés des points des images gauches et droites et reconstruire un modèle 3D

- Structure from motion (cameras étalonnées) : trouver les appariements, puis trouver les paramètres extrinseques, puis reconstruire

- Caméras non-étalonnées : trouver les appariements, trouver la matrice de projection (à un facteur près), puis reconstruire à facteur près

# Disparité




$$\frac{b}{Z} = \frac{(b + x_d) - x_g}{Z - f} \Rightarrow d = x_g - x_d = \frac{f.b}{Z}$$