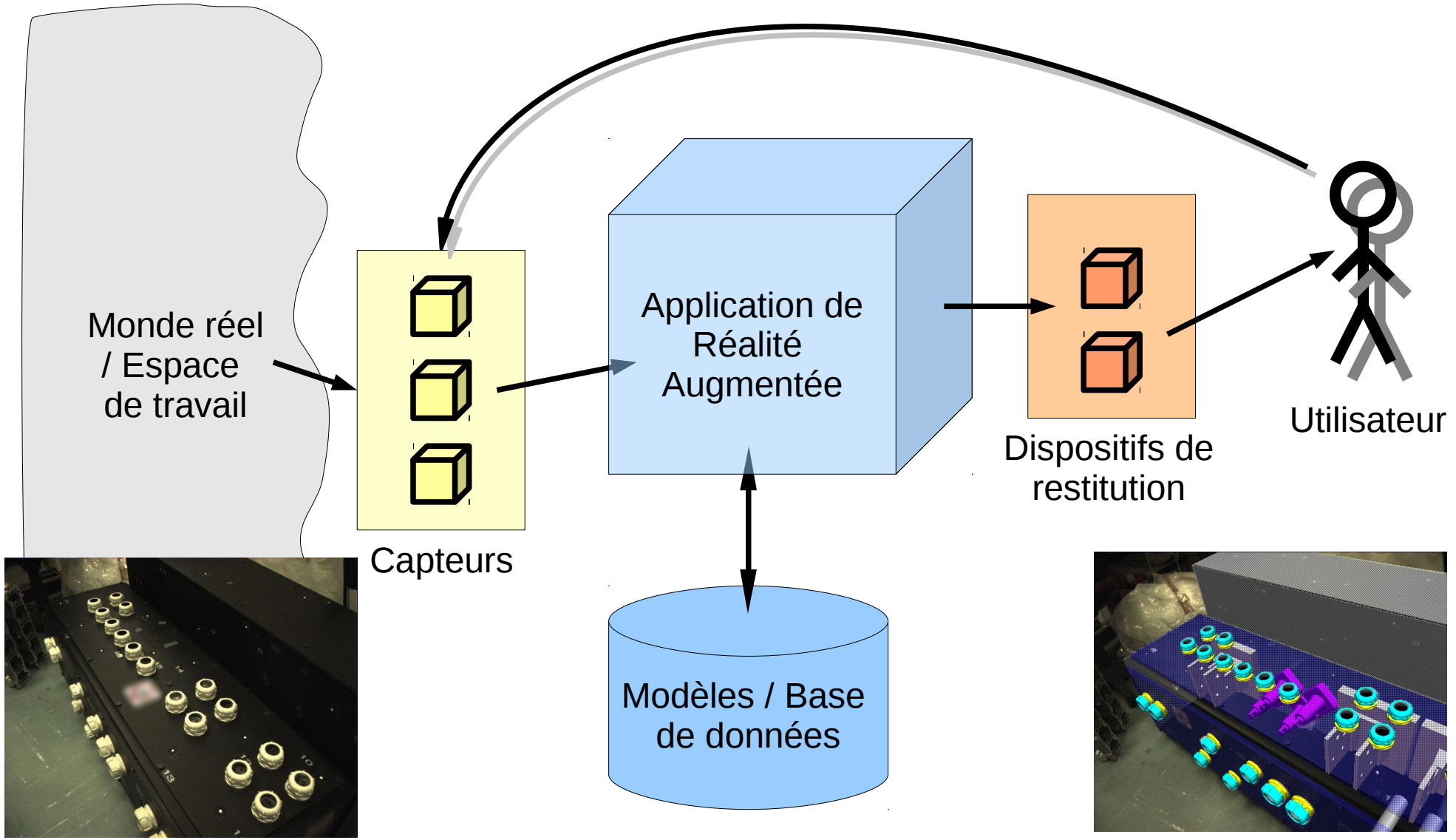
Composition de scènes de Réalité Augmentée

J-Y. Didier

Système de RA



Composition de scènes de
Réalité Augmentée

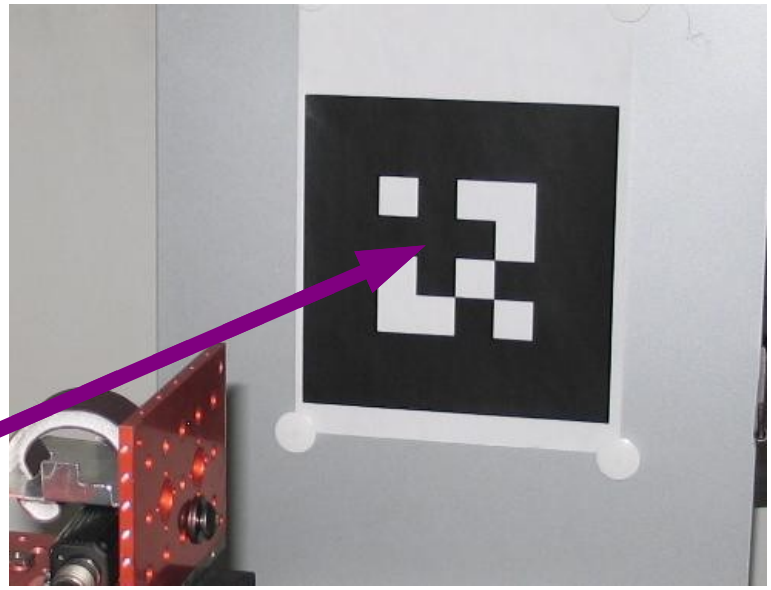
Contexte / Principe général

- RA en vision indirecte :
  - Une caméra filme la scène ;
  - La scène augmentée restitue l'image de la caméra.

- Recalage = points de vue réel et virtuel alignés.
  - Comment construire la scène ?
  - Comment lier caméras réelle et virtuelles ?
  - Comment régler les problèmes liés aux occultations ?
  - Help ! Ca ne marche toujours pas !

Composition simple

- Scène virtuelle



Arrière-plan : image acquise

Premier-plan : entités 3D

Point de vue

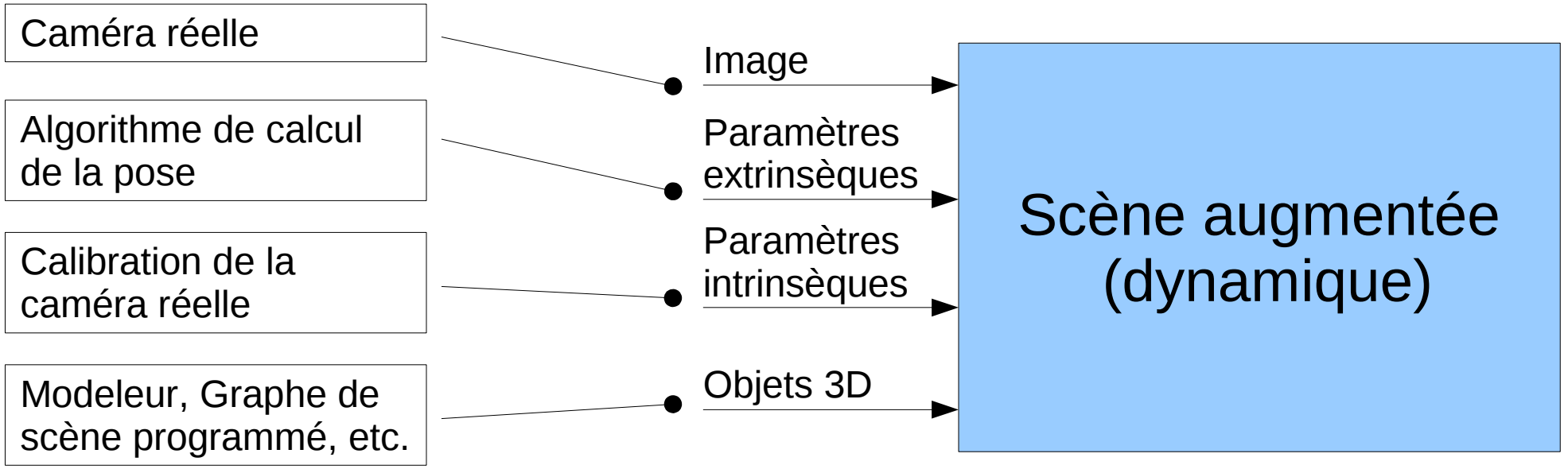
Ce qui est vu :

Caméra réelle : définition

- Qu'est ce qu'une caméra réelle ?
  - Un capteur placé à un endroit et orienté d'une certaine manière dans l'espace de travail ;
    - La caméra peut-être définie par un repère local ;
    - S'il existe un repère associé à l'espace de travail, la relation liant les 2 repères (cad orientation + position) constitue les **paramètres extrinsèques**.
  - Elle crée une image 2D à partir de l'espace de travail ;
    - Les paramètres d'un modèle de cette projection sont appelés **paramètres intrinsèques.**

Aspect système

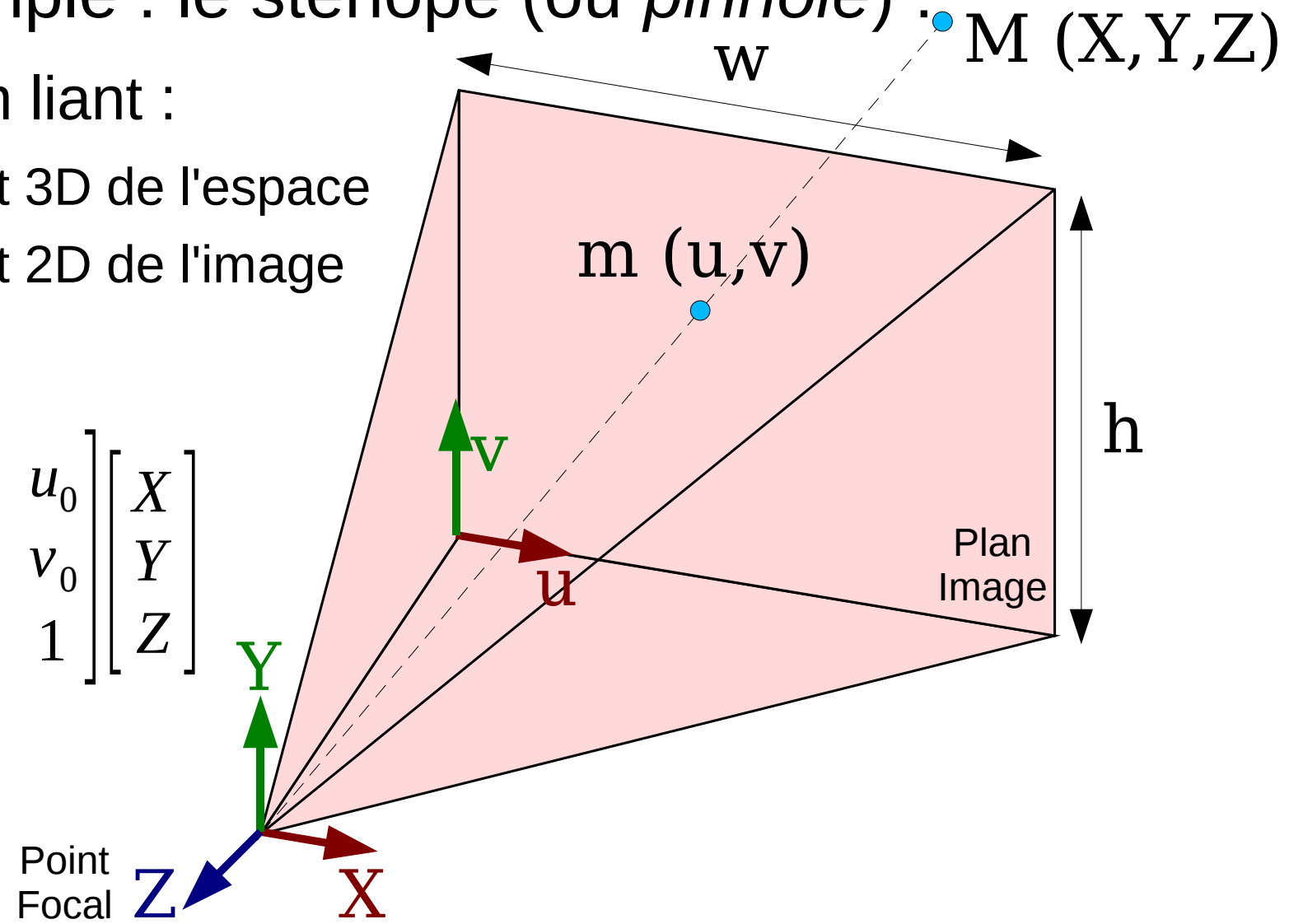
- Comment peut-être vue la brique ?
- D'où proviennent les données ?



Modèle de caméra réelle

- Modèle simple : le sténopé (ou *pinhole*) :
  - Projection liant :
    - Un point 3D de l'espace
    - Un point 2D de l'image

$$\begin{bmatrix} s.u \\ s.v \\ s \end{bmatrix} = \begin{bmatrix} \alpha_u & s_{uv} & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

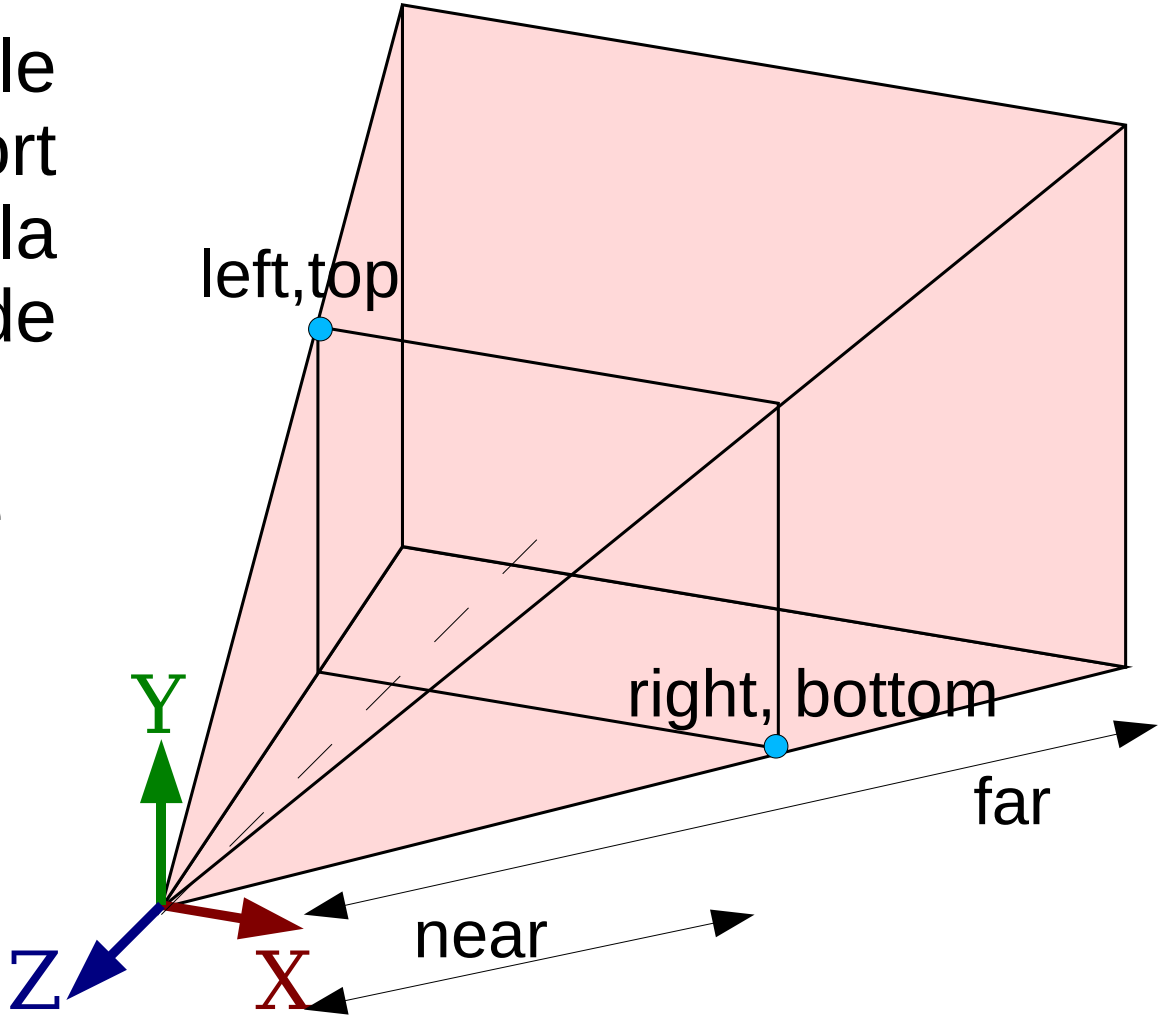


Modèle de caméra réelle

- Paramètres (obtenus par calibration) :
  - α_u : distance focale exprimée en largeurs de pixels
  - α_v : distance focale exprimée en hauteurs de pixels
  - u_0 : abscisse du point focal projeté sur le plan image
  - v_0 : ordonnée du point focal projeté sur le plan image
  - s_{uv} : traduit la non orthogonalité de la matrice physique
    - Souvent négligé (*s* = *slant* ou *skew*).
  - w : largeur en pixels de la matrice physique
  - h : hauteur en pixels de la matrice physique
- Tous sont considérés ici > 0 (sauf s_{uv}) !

Modèle de caméra virtuelle

- Définie suivant des plans de coupe (*clipping*) :
  - Définit la portion visible de l'espace (par rapport à la caméra) sous la forme d'une pyramide tronquée (*frustum*)
  - Ce modèle transforme cet espace en un espace normalisé (cube de côté 2).
  - Une notion de profondeur reste.



Relation entre caméra réelle et caméra virtuelle (frustum)

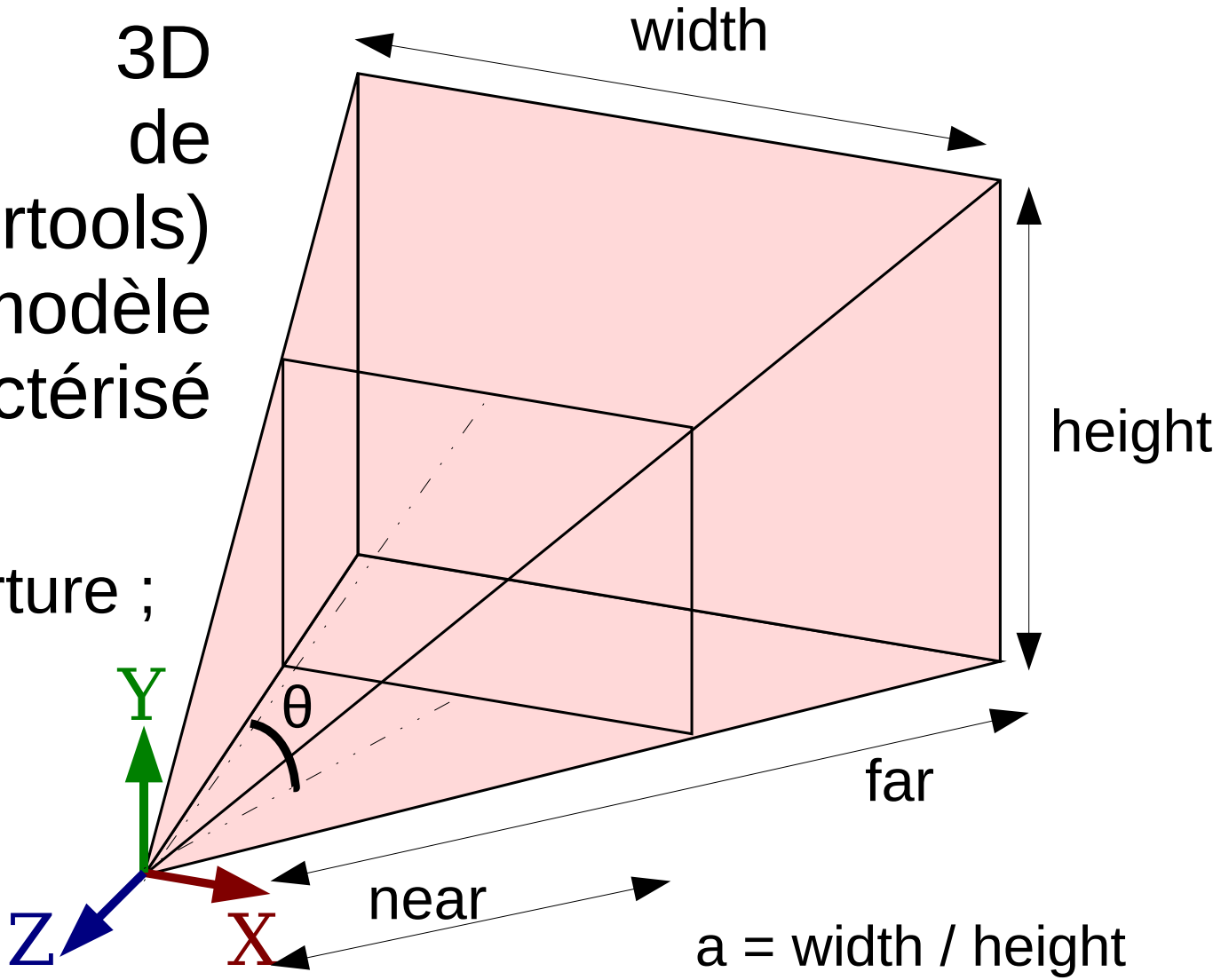
- Thales be my guide !
  - *near* et *far* doivent être fixés :
    - Soit de manière arbitraire ;
    - Soit en calculant la boîte englobante des objets 3D de la scène si cela est possible.

$$left = -\frac{near}{\alpha_u}.u_0 \qquad bottom = -\frac{near}{\alpha_v}.v_0$$

$$right = \frac{near}{\alpha_u}.\left(w - u_0\right) \qquad top = \frac{near}{\alpha_v}.\left(h - v_0\right)$$

Lorsque les frustums ne sont pas disponibles

- Certaines API 3D n'utilisent pas de frustums (ex : Virtools) mais un modèle simplifié caractérisé par :
  - θ : angle d'ouverture ;
  - *a* : aspect ratio ;
  - *near* et *far.*



Sans les frustums

- Recouper l'image pour la rendre symétrique par rapport à la projection du centre optique de la caméra sur le plan image !
  - *near* et *far* sont à fixer.

$$a = \frac{min(u_0, w - u_0)}{min(v_0, h - v_0)}$$

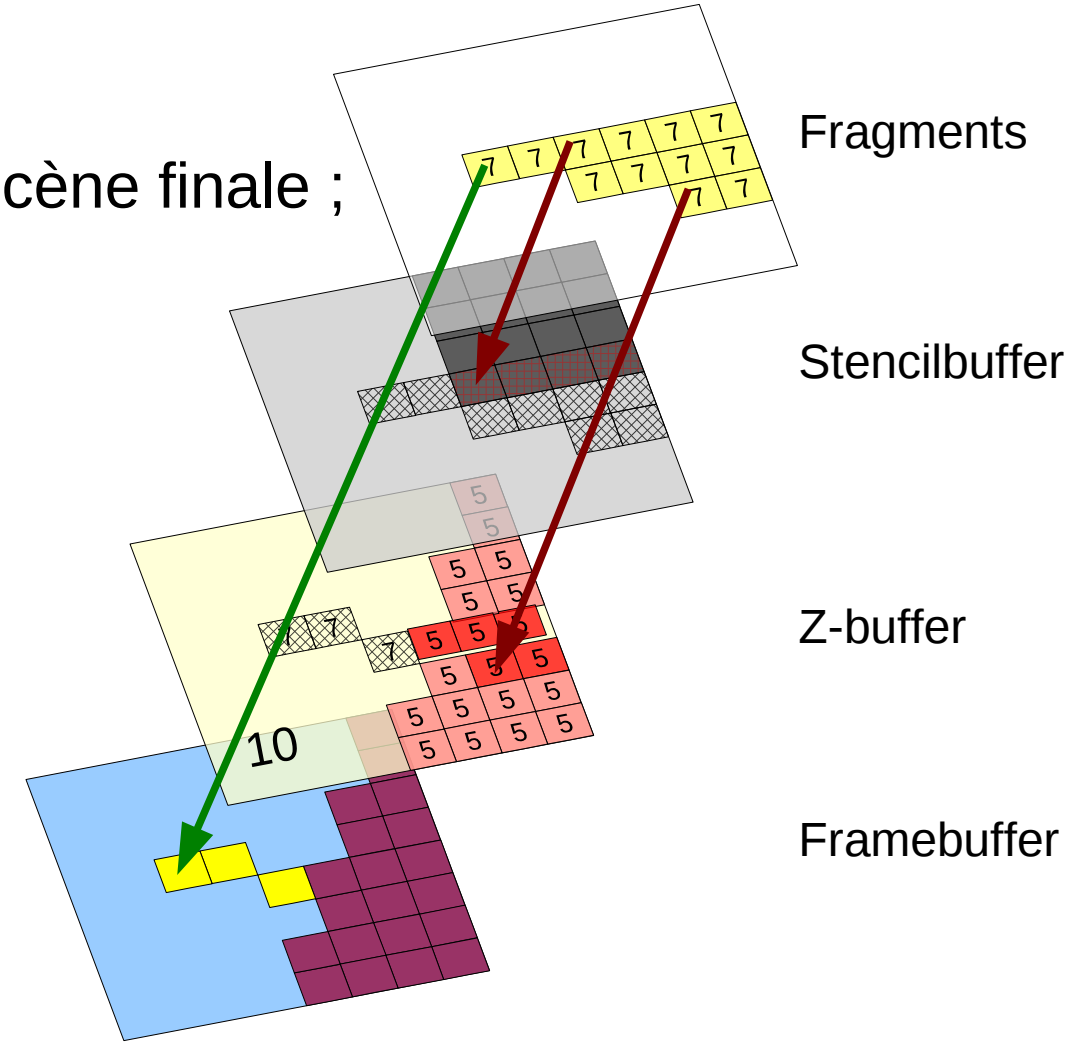
$$\theta = 2 . atan\left(\frac{near}{\alpha_v} . min(v_{0,} h - v_0) \right)$$

buffers et carte graphique 3D

- Une scène 3D : des polygones + des textures ;
- Le travail de la carte graphique :
  - Décomposer polygones et textures en fragments ;
  - L'idée : 1 fragment = 1 pixel dans l'image finale :
    - Paramètres associés : *u, v, couleur, profondeur*
  - Ces fragments passent au travers de différents tamis !
  - Chaque tamis est programmable ;
  - Taille du tamis = Taille de l'image finale ;
  - 1 tamis est 1 *buffer.*

buffers et carte graphique 3D

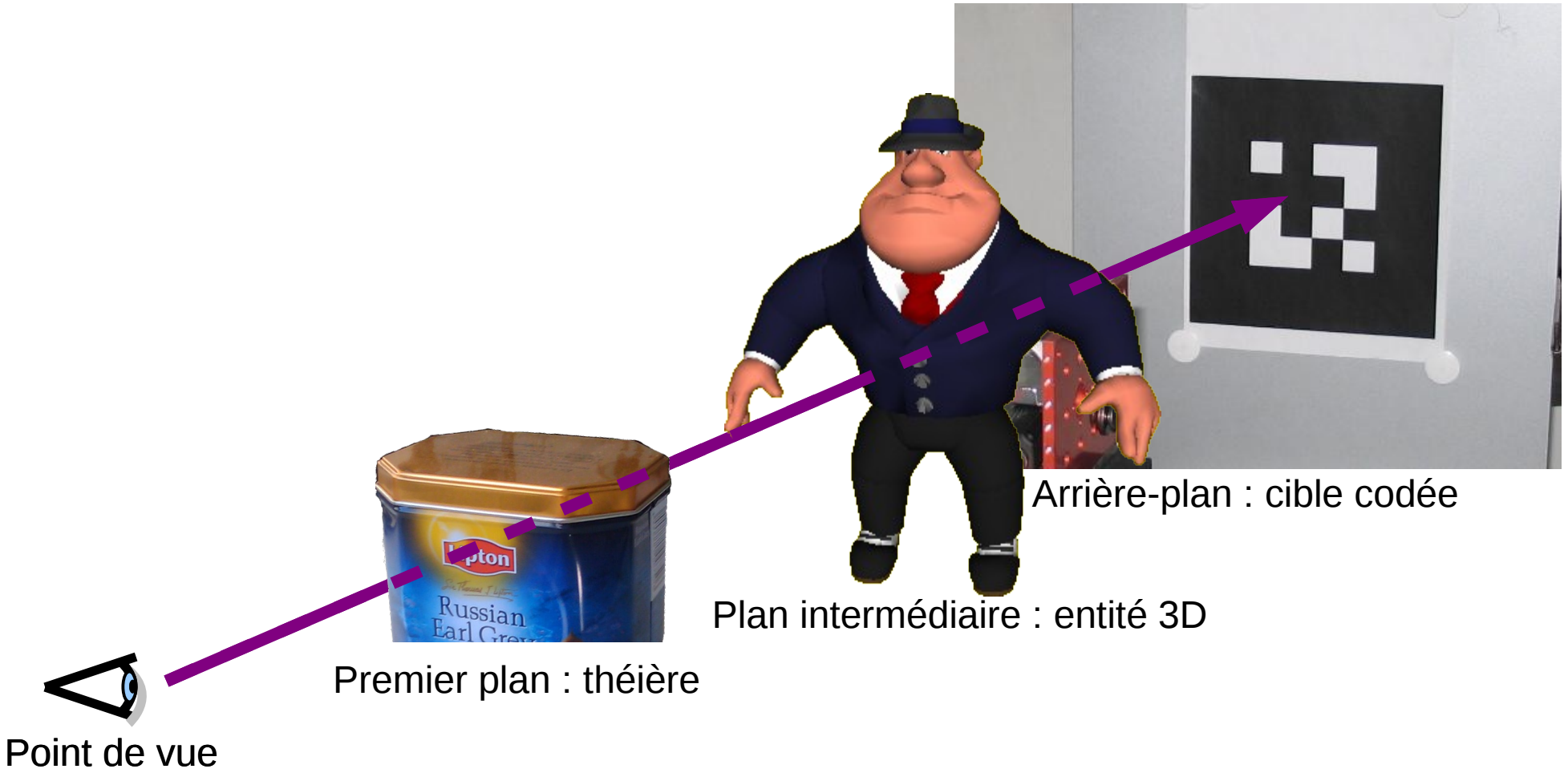
- Les plus connus (et les plus utiles) :
  - Le *framebuffer* :
    - contient l'image de la scène finale ;
  - Le *z-buffer* :
    - test de profondeur ;
  - Le *stencilbuffer* :
    - similaire au pochoir ;



Le rendu de l'image

- Deux techniques pour l'effectuer :
  - En transférant pixel à pixel dans le framebuffer :
    - Pas d'antialiasing ;
    - Pas disponible pour toutes les APIs ;
    - Reste très bas niveau.
  - En utilisant un plan texturé :
    - Parfois plus lent ;
    - Plus complexe à mettre en oeuvre car nécessite des calculs de positionnement du plan texturé à faire en fonction de la boîte englobante des objets 3D de la scène ;
    - Permet de corriger les distortions radiales et tangentielles.

Les occultations réel/virtuel



Les occultations réel/virtuel

Occultations non gérées :

La présence de l'augmentation perturbe la perception des profondeurs

Occultations gérées :

L'augmentation est intégrée à son environnement

Gestion des occultations

- Ici, les occultations du virtuel par le réel.
  - Notre composition simple ne le permet pas *a priori* !
  - Utiliser les buffers :
    - Stencilbuffer : effectuer une première passe dedans
      - Possible si la géométrie des occultations est connue.
    - Z-buffer : marquer directement les profondeurs
      - Nécessite de récupérer la profondeur des objets occultants.
    - Avantage :
      - Cablé dans toutes les cartes graphiques actuelles ;
    - Inconvénient :
      - Nécessite parfois un interfaçage bas niveau ou de court-circuiter certaines API 3D.

Les sources d'erreurs

- Les repères :
  - Repère image (projection) :
    - Point d'ancrage dans le plan image ;
    - Directions principales du repère ;
    - Paramètres positifs ou non (ex: Faugeras Toscani) ?
  - Repère monde (transformation spatiale) :
    - Placement du repère ;
    - Directions (main gauche ou main droite ?) ;
    - L'algorithme de pose fournit-il :
      - la pose de la caméra par rapport au repère monde ?
      - la pose de l'objet par rapport à la caméra ?
        - Inversion de la transformation donnée.

Les sources d'erreurs

- Les conventions :
  - Quaternions : w,x,y,z ou x,y,z,w ?
  - Notations anglo-saxonnes (matrice transposées) :
    - Utilisée par les API 3D modernes (OpenGL et Direct3D)

- Ca reste quand même noir !
  - ~~L'écran est il branché ?~~
  - Les plans de clipping sont ils corrects (near et far) ?

Eléments matriciels

- Inversion d'une matrice de transformation :

$$M = \begin{bmatrix} R_{33} & t_{31} \\ 0_{13} & 1 \end{bmatrix} \qquad M^{-1} = \begin{bmatrix} R_{33}^{t} & -R_{33}^{t} t_{31} \\ 0_{13} & 1 \end{bmatrix}$$

- Messieurs les anglais, tirez les premiers !
  - Matrice transposée = multiplication de vecteurs lignes !

$$\begin{bmatrix} X' \\ Y' \\ Z' \\ 1 \end{bmatrix} = \begin{bmatrix} R_{33} & t_{31} \\ 0_{13} & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \quad \leftrightarrow \quad \begin{bmatrix} X'Y'Z'1 \end{bmatrix} = \begin{bmatrix} XYZ1 \end{bmatrix} \begin{bmatrix} R_{33}^{t} & 0_{31} \\ t_{31}^{t} & 1 \end{bmatrix}$$

The French touch

The English one

Discussion sur la calibration

- Doit on calibrer précisément la caméra physique ?
  - Réponse courte : cela dépend !
  - Oui :
    - Si l'estimation des distances est souhaitée ;
    - Si les objets à afficher sont de grande taille par rapport à l'image de la scène finale ;
    - Si les informations de profondeur sont importantes.
  - Non :
    - Si aucun des points précédents n'est important ;
    - L'objet sera quand même affiché au bon endroit si l'algorithme de calcul de la pose utilise les mêmes paramètres de caméras.

Discussion sur le photo-réalisme

- Photo-réalisme :
  - Consiste à rendre les objets 3D incrustés dans la scène réaliste cad comme s'ils faisaient vraiment partie de la scène.
- Problèmes sous-jacent :
  - Réalisme au niveau des occultations ;
  - Réalisme du rendu des lumières et des réflexions de l'environnement sur l'objet.
- Est ce nécessaire ?
  - Tout dépend de l'application ;
  - Ex : une carte permet de mieux se répèrer qu'une photo satellite.

Questions

Ressources

- Thèses de l'équipe RATC :
  - http://evra.ibisc.univ-evry.fr/
- Page Gilles Simon :
  - http://webloria.loria.fr/~gsimon/ra/
- Etc ...