

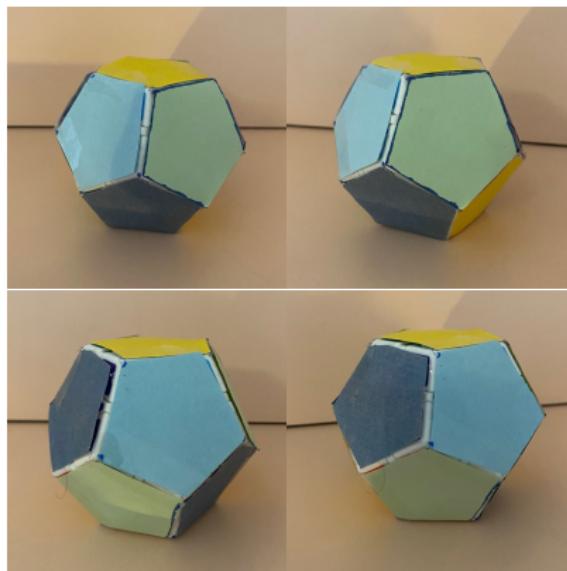
# Reconstruction d'objets convexes à partir de photographies

Présentation de **Lucie-Hélène Cuingnet**

Travail réalisé avec **Barnabé Baruchel**

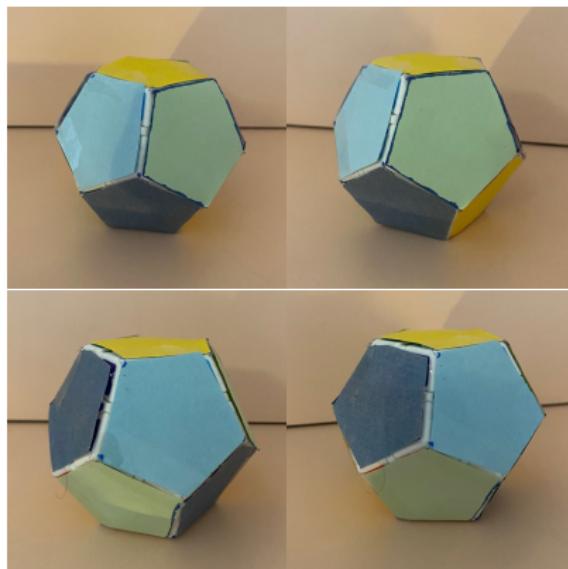
TIPE 2025

## Définition du problème

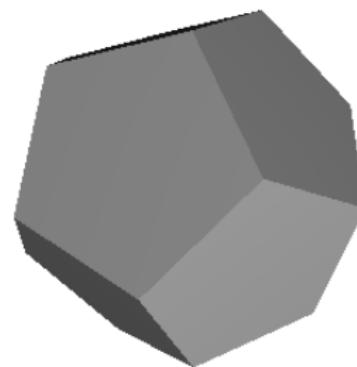


Données

## Définition du problème



Données



Objectif

# Plan

## 1. Selection

Points d'intérêts

Algorithme

Implémentation

## 2. Appariement

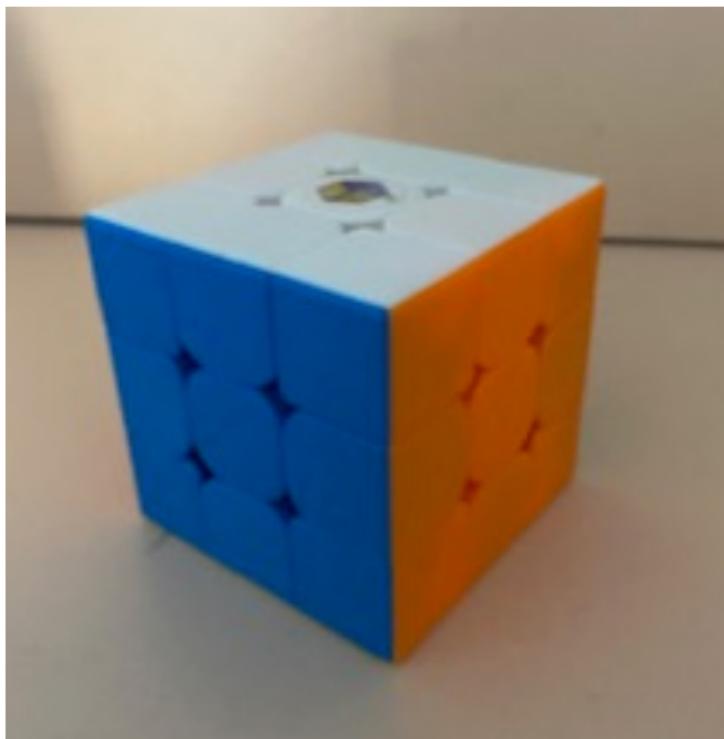
## 3. Calibration

## 4. Reconstruction des points

## 5. Enveloppe Convexe

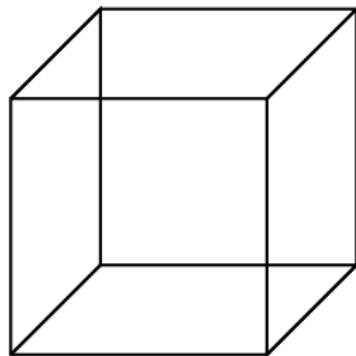
## 6. Analyse des résultats

## Points d'intérêts



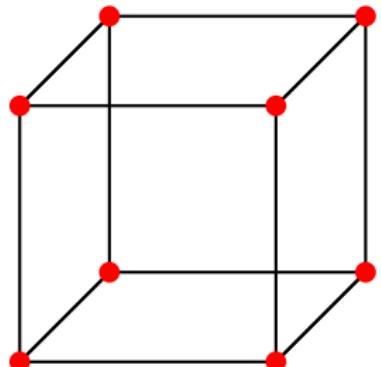
# Points d'intérêts

## Points d'intérêts sur un cube



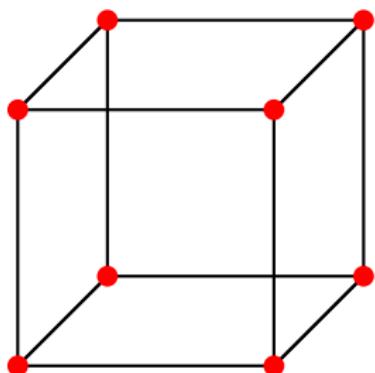
# Points d'intérêts

## Points d'intérêts sur un cube

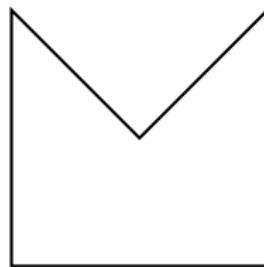


# Points d'intérêts

## Points d'intérêts sur un cube

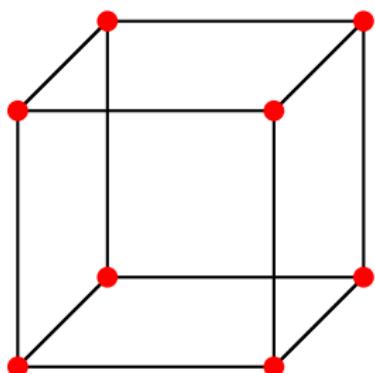


## Problème si non convexe

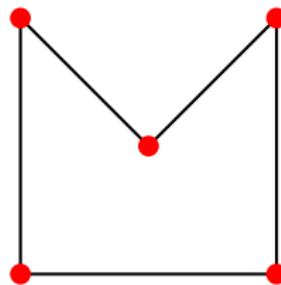


# Points d'intérêts

Points d'intérêts sur un cube

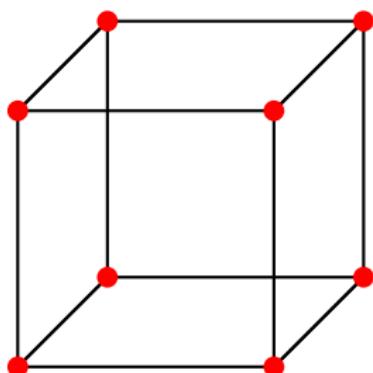


Problème si non convexe

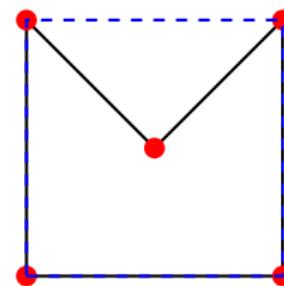


# Points d'intérêts

## Points d'intérêts sur un cube



## Problème si non convexe



Enveloppe convexe

Exemple

code

---

**Algorithme:** Moravec (minimum des variances)

---

**Input:** Image d'intensité image

---

Exemple

code

---

### **Algorithme:** Moravec (minimum des variances)

---

**Input:** Image d'intensité `image`

**Output:** Matrice des coins détecté

---

Exemple

code

---

**Algorithme:** Moravec (minimum des variances)

---

**Input:** Image d'intensité `image`

**Output:** Matrice des coins détecté

---

[Exemple](#)[code](#)

---

**Algorithme:** Moravec (minimum des variances)

---

**Input:** Image d'intensité `image`**Output:** Matrice des coins détecté**pour tout pixel  $(x, y)$  dans l'image faire**

---

[Exemple](#)[code](#)

---

**Algorithme:** Moravec (minimum des variances)

---

**Input:** Image d'intensité `image`**Output:** Matrice des coins détecté**pour tout** pixel  $(x, y)$  dans l'image **faire**└     `scores ← liste vide;`

[Exemple](#)[code](#)

---

## Algorithme: Moravec (minimum des variances)

---

**Input:** Image d'intensité  $image$

**Output:** Matrice des coins détecté

**pour tout** pixel  $(x, y)$  dans l'image **faire**

  └ scores  $\leftarrow$  liste vide;

**pour tout** direction  $(dx, dy)$  parmi : verticale, horizontale, diagonales

**faire**

---

**Exemple****code**

---

**Algorithme:** Moravec (minimum des variances)

---

**Input:** Image d'intensité  $image$ **Output:** Matrice des coins détecté**pour tout** pixel  $(x, y)$  dans l'image **faire**└ scores  $\leftarrow$  liste vide;**pour tout** direction  $(dx, dy)$  parmi : verticale, horizontale, diagonales**faire**└ Calculer la variance locale autour de  $(x, y)$  dans la direction  
 $(dx, dy)$ ;

└ Ajouter la variance à scores;

[Exemple](#)[code](#)

---

### Algorithme: Moravec (minimum des variances)

---

**Input:** Image d'intensité  $\text{image}$

**Output:** Matrice des coins détecté

**pour tout** pixel  $(x, y)$  dans l'image **faire**

scores  $\leftarrow$  liste vide;

**pour tout** direction  $(dx, dy)$  parmi : verticale, horizontale, diagonales  
**faire**

Calculer la variance locale autour de  $(x, y)$  dans la direction  
 $(dx, dy)$ ;

Ajouter la variance à scores;

$\text{score} \leftarrow \min(\text{scores});$

---

[Exemple](#)[code](#)

---

## Algorithme: Moravec (minimum des variances)

---

**Input:** Image d'intensité `image`

**Output:** Matrice des coins détecté

**pour tout** pixel  $(x, y)$  dans l'image **faire**

`scores`  $\leftarrow$  liste vide;

**pour tout** direction  $(dx, dy)$  parmi : verticale, horizontale, diagonales  
**faire**

Calculer la variance locale autour de  $(x, y)$  dans la direction  
 $(dx, dy)$ ;

Ajouter la variance à `scores`;

`score`  $\leftarrow \min(scores)$ ;

**if** `score` > SEUIL **then**

---

**Exemple****code**

---

## Algorithme: Moravec (minimum des variances)

---

**Input:** Image d'intensité `image`

**Output:** Matrice des coins détecté

**pour tout** pixel  $(x, y)$  dans l'image **faire**

  └ scores  $\leftarrow$  liste vide;

**pour tout** direction  $(dx, dy)$  parmi : verticale, horizontale, diagonales  
  **faire**

    └ Calculer la variance locale autour de  $(x, y)$  dans la direction  
       $(dx, dy)$ ;

    └ Ajouter la variance à `scores`;

  score  $\leftarrow \min(scores)$ ;

**if** score > SEUIL **then**

    └ Marquer  $(x, y)$  comme coin;

[Exemple](#)[code](#)

---

## Algorithme: Moravec (minimum des variances)

---

**Input:** Image d'intensité `image`

**Output:** Matrice des coins détecté

**pour tout** pixel  $(x, y)$  dans l'image **faire**

  └ `scores`  $\leftarrow$  liste vide;

**pour tout** direction  $(dx, dy)$  parmi : verticale, horizontale, diagonales  
**faire**

  └ Calculer la variance locale autour de  $(x, y)$  dans la direction  
     $(dx, dy)$ ;

  └ Ajouter la variance à `scores`;

`score`  $\leftarrow \min(scores)$ ;

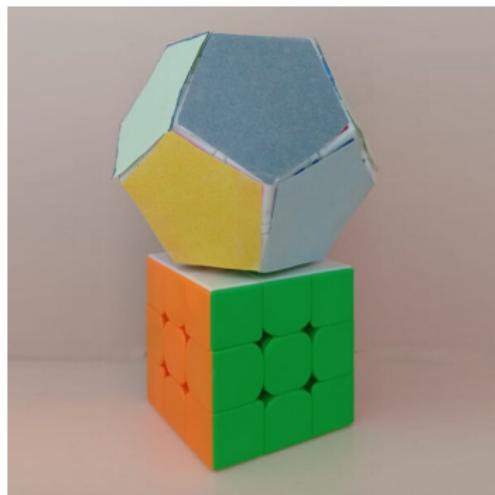
**if** `score`  $>$  SEUIL **then**

    └ Marquer  $(x, y)$  comme coin;

**return** Liste des points marqués

---

## Fichier de sortie PBM



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

# Influence des paramètres de Moravec

Paramètres utilisés :

- ▶ **THRESHOLD = 2000**
- ▶ **WINDOW = 4**



## Influence des paramètres de Moravec

Paramètres utilisés :

- ▶ **THRESHOLD** = 2000
- ▶ **WINDOW** = 6



## Influence des paramètres de Moravec

Paramètres utilisés :

- ▶ **THRESHOLD = 4000**
- ▶ **WINDOW = 10**



# Influence des paramètres de Moravec

Paramètres utilisés :

- ▶ **THRESHOLD = 500**
- ▶ **WINDOW = 2**



# Plan

1. Selection

2. Appariement

Pré-traitement

Filtrage epipolaire

Brief

Algorithme

Resultat

3. Calibration

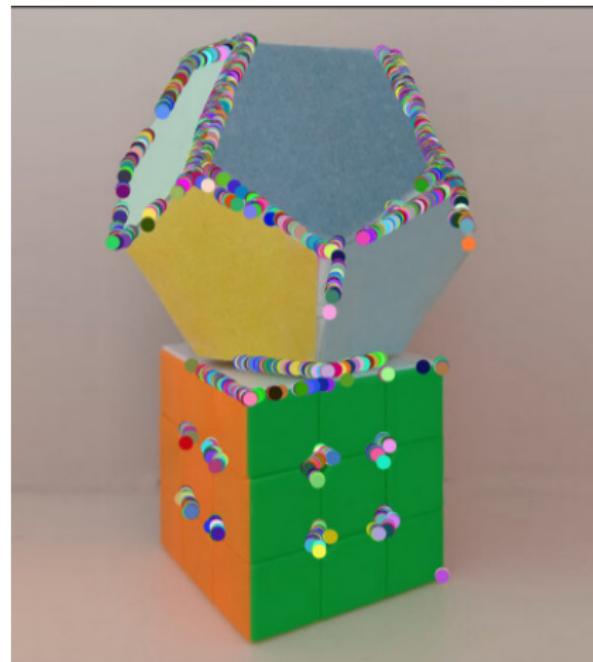
4. Reconstruction des points

5. Enveloppe Convexe

6. Analyse des résultats

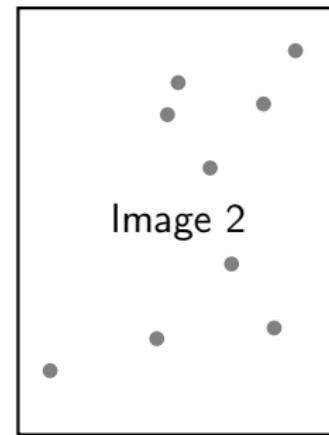
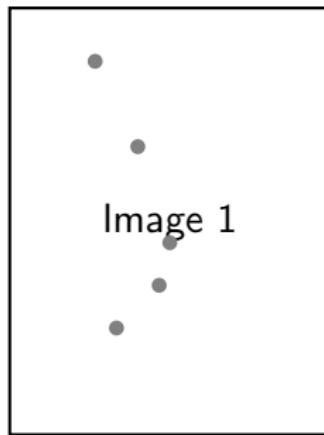
2- Appariement

## Le problème de l'appariement



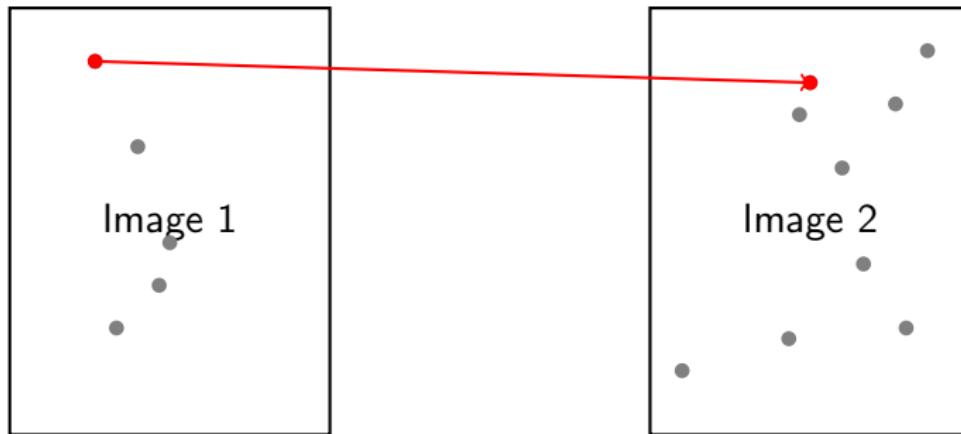
2- Appariement

# Le problème de l'appariement



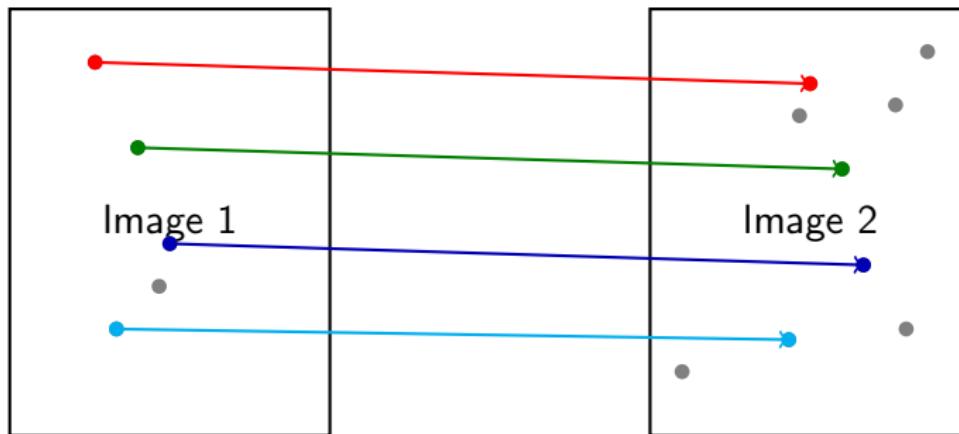
2- Appariement

## Le problème de l'appariement

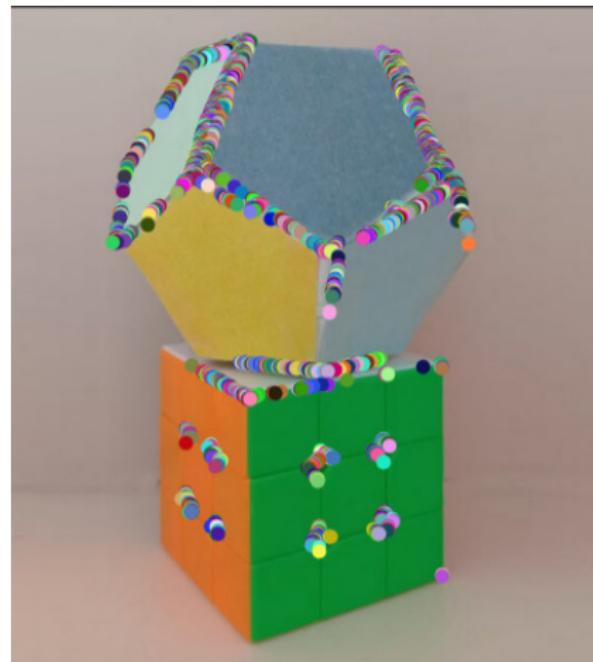


2- Appariement

## Le problème de l'appariement



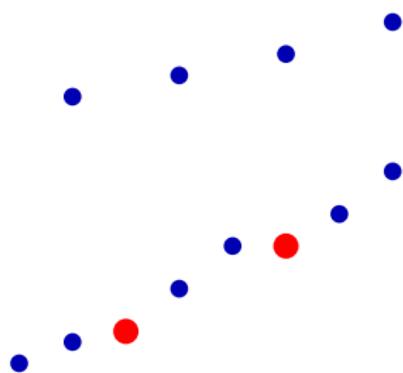
## Pré-traitement



## Ransac

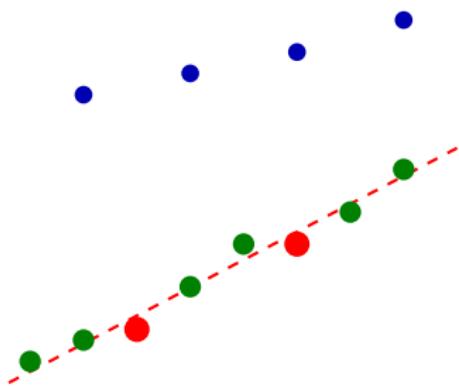


# Ransac



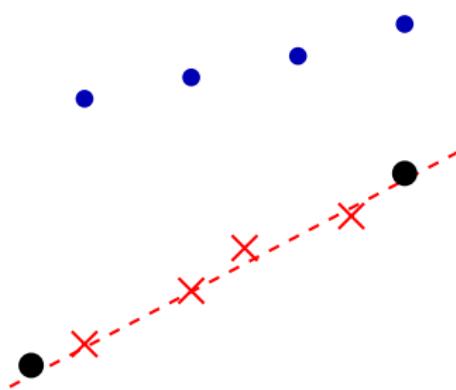
- ▶ Points d'entrée dans le nuage
- ▶ Sélection de deux points pour estimer une droite

# Ransac



- ▶ Points d'entrée dans le nuage
- ▶ Sélection de deux points pour estimer une droite
- ▶ Inliers détectés à proximité de la droite

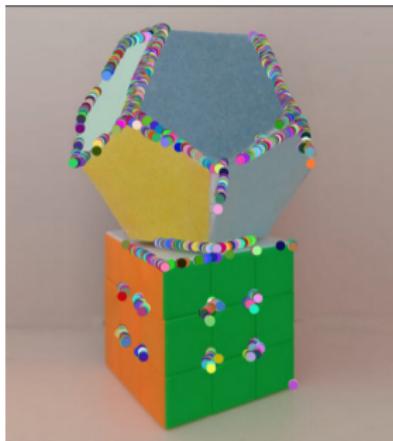
# Ransac



- ▶ Points d'entrée dans le nuage
- ▶ Sélection de deux points pour estimer une droite
- ▶ Inliers détectés à proximité de la droite
- ▶ Suppression des points internes, on garde les extrémités

⇒ Nettoyage effectué : seuls les segments visibles sont conservés

## Résultats pré-traitement

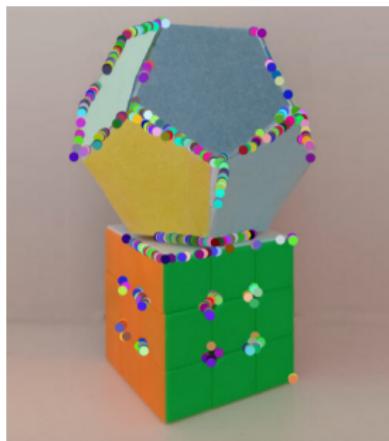


Moravec

## Résultats pré-traitement



Moravec

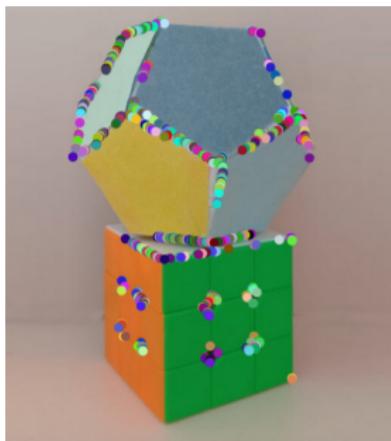


Trouve coin

## Résultats pré-traitement



Moravec

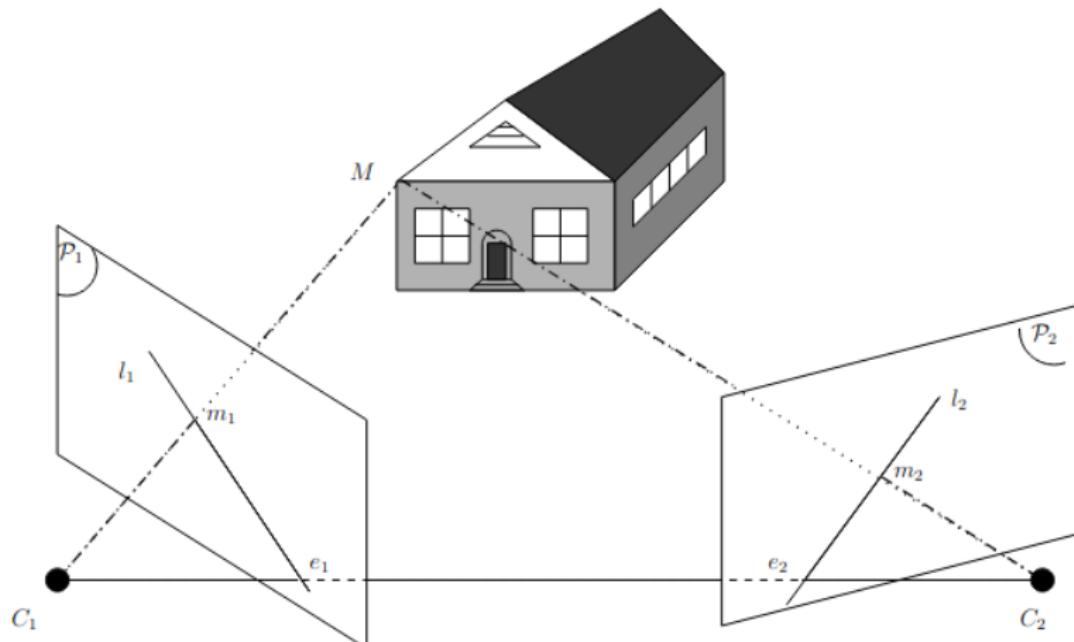


Trouve coin



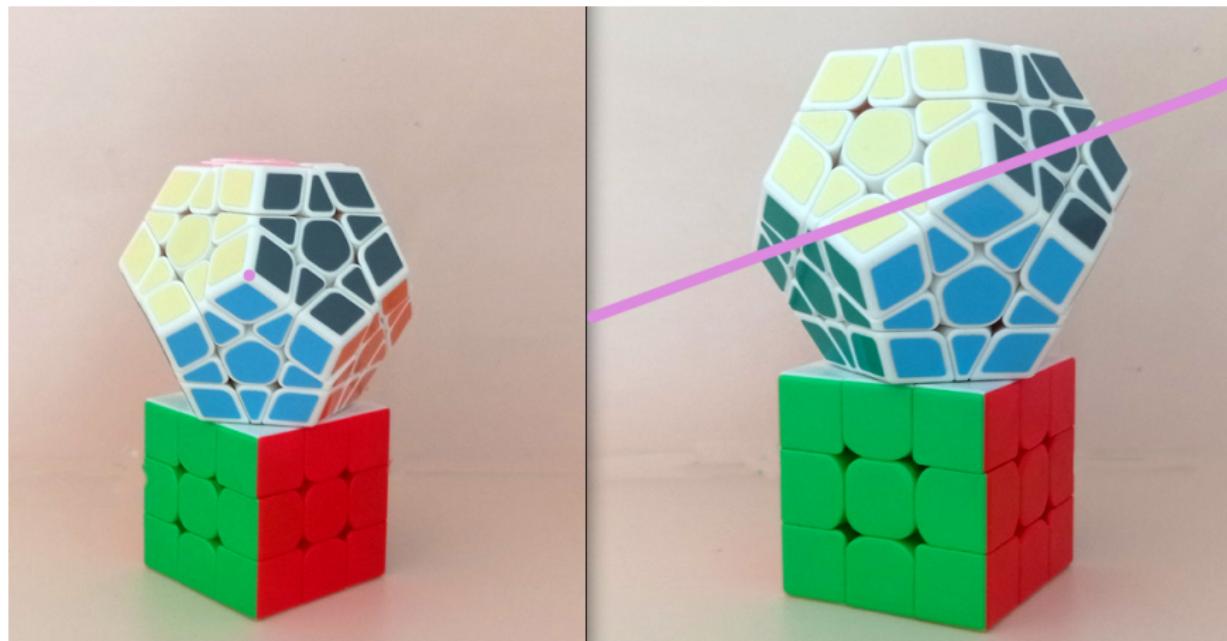
Ransac

# Geometrie epipolaire



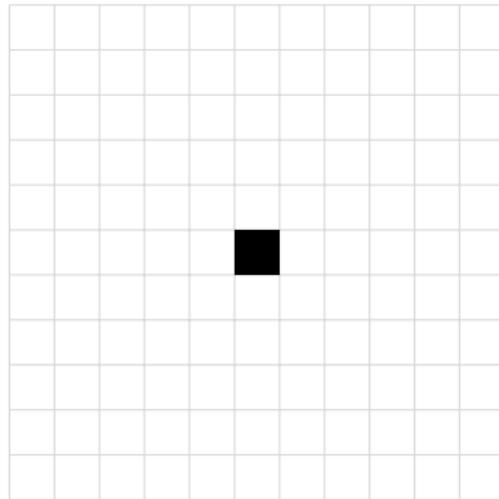
Source image: Quelques problèmes géométriques en vision par ordinateur - Frédéric SUR

## Filtrage epipolaire



Droite epipolaire

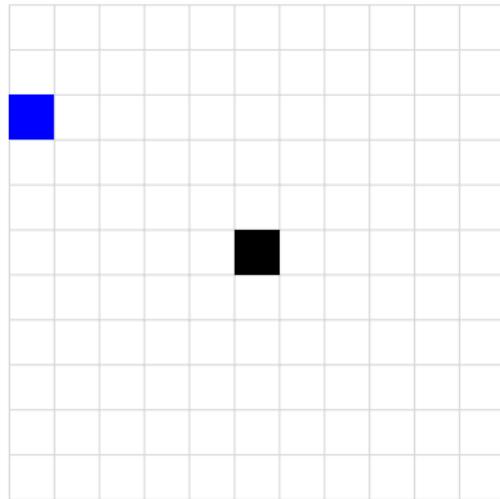
## Descripteur BRIEF : Principe



**Comparaisons binaires :**

- ▶ On considère une fenêtre autour d'un point clé.

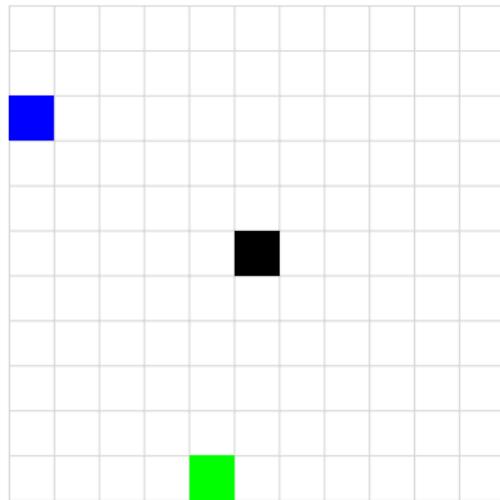
## Descripteur BRIEF : Principe



**Comparaisons binaires :**

- ▶ On considère une fenêtre autour d'un point clé.
- ▶ On choisit aléatoirement un pixel (ex: bleu).

## Descripteur BRIEF : Principe

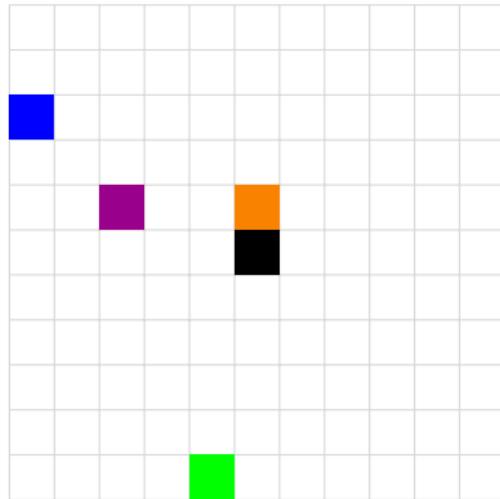


**Comparaisons binaires :**

Bleu < Vert  $\Rightarrow 1$

- ▶ On considère une fenêtre autour d'un point clé.
- ▶ On choisit aléatoirement un pixel (ex: bleu).
- ▶ On la compare à un autre (ex: vert) :  
 $BRIEF_1 = 1$  si intensité(bleu) < intensité(vert)

## Descripteur BRIEF : Principe



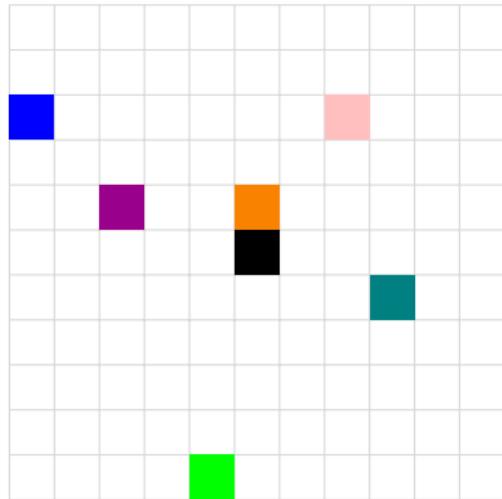
**Comparaisons binaires :**

Bleu < Vert  $\Rightarrow 1$

Orange > Violet  $\Rightarrow 0$

- ▶ On considère une fenêtre autour d'un point clé.
- ▶ On choisit aléatoirement un pixel (ex: bleu).
- ▶ On la compare à un autre (ex: vert) :  
 $BRIEF_1 = 1$  si intensité(bleu) < intensité(vert)
- ▶ On répète avec d'autres paires (ex: orange vs violet)...

## Descripteur BRIEF : Principe



### Comparaisons binaires :

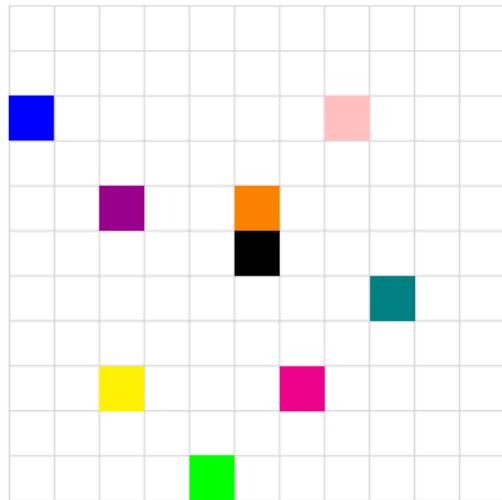
Bleu < Vert  $\Rightarrow 1$

Orange > Violet  $\Rightarrow 0$

Rose < Turquoise  $\Rightarrow 1$

- ▶ On considère une fenêtre autour d'un point clé.
- ▶ On choisit aléatoirement un pixel (ex: bleu).
- ▶ On la compare à un autre (ex: vert) :  
 $BRIEF_1 = 1$  si intensité(bleu) < intensité(vert)
- ▶ On répète avec d'autres paires (ex: orange vs violet)...

## Descripteur BRIEF : Principe



### Comparaisons binaires :

Bleu < Vert  $\Rightarrow 1$

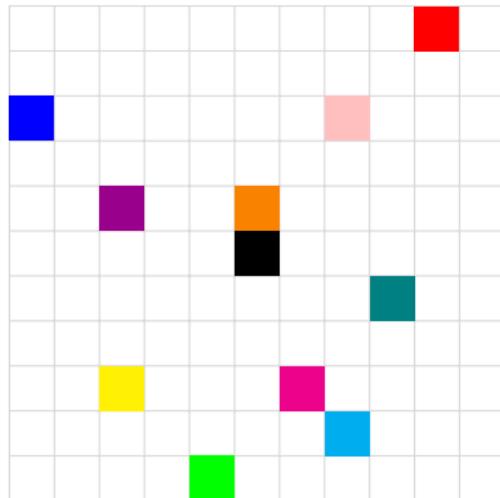
Orange > Violet  $\Rightarrow 0$

Rose < Turquoise  $\Rightarrow 1$

Jaune > Magenta  $\Rightarrow 0$

- ▶ On considère une fenêtre autour d'un point clé.
- ▶ On choisit aléatoirement un pixel (ex: bleu).
- ▶ On la compare à un autre (ex: vert) :  
 $BRIEF_1 = 1$  si intensité(bleu) < intensité(vert)
- ▶ On répète avec d'autres paires (ex: orange vs violet)...

## Descripteur BRIEF : Principe

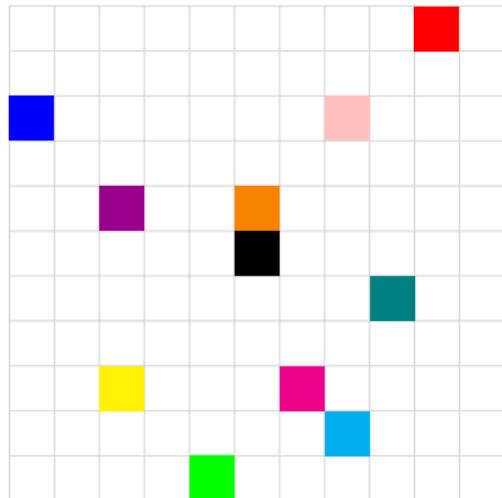


### Comparaisons binaires :

Bleu < Vert  $\Rightarrow 1$   
Orange > Violet  $\Rightarrow 0$   
Rose < Turquoise  $\Rightarrow 1$   
Jaune > Magenta  $\Rightarrow 0$   
Cyan < Rouge  $\Rightarrow 1$

- ▶ On considère une fenêtre autour d'un point clé.
- ▶ On choisit aléatoirement un pixel (ex: bleu).
- ▶ On la compare à un autre (ex: vert) :  
 $BRIEF_1 = 1$  si intensité(bleu) < intensité(vert)
- ▶ On répète avec d'autres paires (ex: orange vs violet)...

# Descripteur BRIEF : Principe



## Comparaisons binaires :

Bleu < Vert  $\Rightarrow 1$   
 Orange > Violet  $\Rightarrow 0$   
 Rose < Turquoise  $\Rightarrow 1$   
 Jaune > Magenta  $\Rightarrow 0$   
 Cyan < Rouge  $\Rightarrow 1$

## Descripteur final :

1 0 1 0 1

- ▶ On considère une fenêtre autour d'un point clé.
- ▶ On choisit aléatoirement un pixel (ex: bleu).
- ▶ On la compare à un autre (ex: vert) :  
 $BRIEF_1 = 1$  si intensité(bleu) < intensité(vert)
- ▶ On répète avec d'autres paires (ex: orange vs violet)...

# Comparaison de descripteurs BRIEF : distance de Hamming

**Descripteur 1 (image gauche)**

1 0 1 0 1

# Comparaison de descripteurs BRIEF : distance de Hamming

**Descripteur 1 (image gauche)**

1 0 1 0 1

**Descripteur 2 (image droite)**

1 1 0 0 1

# Comparaison de descripteurs BRIEF : distance de Hamming

**Descripteur 1 (image gauche)**

1 0 1 0 1

**Descripteur 2 (image droite)**

1 1 0 0 1

**Comparaison bit à bit :**

$$\begin{array}{ccccc} 1 & 0 & 1 & 0 & 1 \\ - & - & - & - & - \\ \hline 1 & 1 & 0 & 0 & 1 \\ \hline 0 & \color{red}{1} & \color{red}{1} & 0 & \color{green}{0} \end{array}$$

# Comparaison de descripteurs BRIEF : distance de Hamming

**Descripteur 1 (image gauche)**

1 0 1 0 1

**Descripteur 2 (image droite)**

1 1 0 0 1

**Comparaison bit à bit :**

$$\begin{array}{ccccc} 1 & 0 & 1 & 0 & 1 \\ - & - & - & - & - \\ \hline 1 & 1 & 0 & 0 & 1 \\ \hline 0 & \color{red}{1} & \color{red}{1} & \color{green}{0} & \color{green}{0} \end{array}$$

**Distance de Hamming = nombre de bits différents = 2**

# Comparaison de descripteurs BRIEF : distance de Hamming

**Descripteur 1 (image gauche)**

1 0 1 0 1

**Descripteur 2 (image droite)**

1 1 0 0 1

**Comparaison bit à bit :**

$$\begin{array}{ccccc} 1 & 0 & 1 & 0 & 1 \\ - & - & - & - & - \\ \hline 1 & 1 & 0 & 0 & 1 \\ \hline 0 & \color{red}{1} & \color{red}{1} & \color{green}{0} & \color{green}{0} \end{array}$$

**Distance de Hamming = nombre de bits différents = 2**

*Plus la distance est faible, plus les points sont similaires.*

# Amélioration progressive du descripteur BRIEF

## 1. BRIEF (intensité)

- Comparaison d'intensité de pixels en niveaux de gris
- *Simple et rapide, mais perte d'information sur la couleur*

# Amélioration progressive du descripteur BRIEF

## 1. BRIEF (intensité)

- Comparaison d'intensité de pixels en niveaux de gris
- *Simple et rapide, mais perte d'information sur la couleur*

## 2. BRIEF RGB

- Comparaison faite indépendamment sur les 3 canaux : R, G, B
- *Capture la couleur, mais très sensible aux variations de lumière*

# Amélioration progressive du descripteur BRIEF

## 1. BRIEF (intensité)

- Comparaison d'intensité de pixels en niveaux de gris
- *Simple et rapide, mais perte d'information sur la couleur*

## 2. BRIEF RGB

- Comparaison faite indépendamment sur les 3 canaux : R, G, B
- *Capture la couleur, mais très sensible aux variations de lumière*

## 3. BRIEF Lab

- Comparaison dans l'espace Lab :
  - $L$  : luminosité (luminance)
  - $a$ ,  $b$  : composantes de couleur perceptuelles
- *Plus robuste aux variations de luminosité et plus proche de la perception humaine*

# Amélioration progressive du descripteur BRIEF

## 1. BRIEF (intensité)

- Comparaison d'intensité de pixels en niveaux de gris
- *Simple et rapide, mais perte d'information sur la couleur*

## 2. BRIEF RGB

- Comparaison faite indépendamment sur les 3 canaux : R, G, B
- *Capture la couleur, mais très sensible aux variations de lumière*

## 3. BRIEF Lab

- Comparaison dans l'espace Lab :
  - $L$  : luminosité (luminance)
  - $a$ ,  $b$  : composantes de couleur perceptuelles
- *Plus robuste aux variations de luminosité et plus proche de la perception humaine*

## Amélioration globale :

- *Robustesse et précision accrues à chaque étape*
- *BRIEF Lab permet de meilleures correspondances entre images variées (éclairage, couleur)*

## Pseudo-code : Appariement de points

---

### Algorithme: Appariement

---

**Entrée:** Points  $P_1$  sur image 1, Points  $P_2$  sur image 2, Matrice fondamentale  $F$

**Sortie:** Liste de correspondances fiables

**Pré-tri des points sur image 1**

**pour tout**  $p \in P_1$  **faire**

**si**  $p$  n'est pas un coin **alors**

**retirer**  $p$

            // suppression non maximale locale

---

# Pseudo-code : Appariement de points

---

## Algorithme: Appariement

---

**Entrée:** Points  $P_1$  sur image 1, Points  $P_2$  sur image 2, Matrice fondamentale  $F$

**Sortie:** Liste de correspondances fiables

### Pré-tri des points sur image 1

**pour tout**  $p \in P_1$  **faire**

**si**  $p$  n'est pas un coin **alors**

        retirer  $p$

        // suppression non maximale locale

---

### Filtrage épipolaire

**pour tout**  $p_1 \in P_1$  **faire**

$I \leftarrow F \cdot p_1$

        // droite épipolaire dans image 2

$C(p_1) \leftarrow \{p_2 \in P_2 \mid \text{distance}(p_2, I) < \varepsilon\}$

## Pseudo-code : Appariement de points

## **Algorithme: Appariement**

**Entrée:** Points  $P_1$  sur image 1, Points  $P_2$  sur image 2, Matrice fondamentale  $F$

### **Sortie:** Liste de correspondances fiables

### **Pré-tri des points sur image 1**

pour tout  $p \in P_1$  faire

si  $p$  n'est pas un coin alors

```
// suppression non maximale locale
```

## Filtrage épipolaire

pour tout  $p_1 \in P_1$  faire

$$l \leftarrow F \cdot p_1$$

// droite épipolaire dans image 2

$$C(p_1) \leftarrow \{p_2 \in P_2 \mid \text{distance}(p_2, I) < \varepsilon\}$$

## Comparaison des descripteurs BRIEF

pour tout  $p_1 \in P_1$  faire

$$d_1 \leftarrow \text{BRIEF}(p_1)$$

**pour tout**  $p_2 \in C(p_1)$  **faire**

$$d_2 \leftarrow \text{BRIEF}(p_2)$$

$h \leftarrow \text{distance\_Hamming}(d_1, d_2)$

enregistrer  $(\bar{p_1}, p_2, h)$

## Pseudo-code : Appariement de points

## **Algorithme: Appariement**

**Entrée:** Points  $P_1$  sur image 1, Points  $P_2$  sur image 2, Matrice fondamentale  $F$

**Sortie:** Liste de correspondances fiables

### **Pré-tri des points sur image 1**

**pour tout**  $p \in P_1$  faire

si  $p$  n'est pas un coin alors

```
// suppression non maximale locale
```

## Filtrage épipolaire

**pour tout**  $p_1 \in P_1$  faire

$$l \leftarrow F \cdot p_1$$

// droite épipolaire dans image 2

$$C(p_1) \leftarrow \{p_2 \in P_2 \mid \text{distance}(p_2, I) < \varepsilon\}$$

## Comparaison des descripteurs BRIEF

pour tout  $p_1 \in P_1$  faire

$$d_1 \leftarrow \text{BRIEF}(p_1)$$

**pour tout**  $p_2 \in C(p_1)$  faire

$$d_2 \leftarrow \text{BRIEF}(p_2)$$

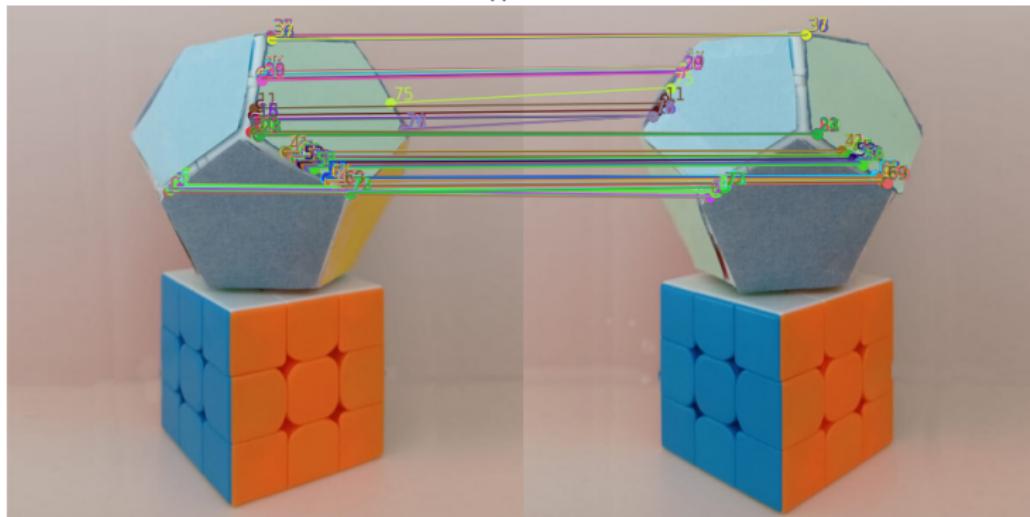
$h \leftarrow \text{distance\_Hamming}(d_1, d_2)$

enregistrer  $(\bar{p_1}, p_2, h)$

**retourner** paires  $(p_1, p_2)$  avec plus petite distance de Hamming

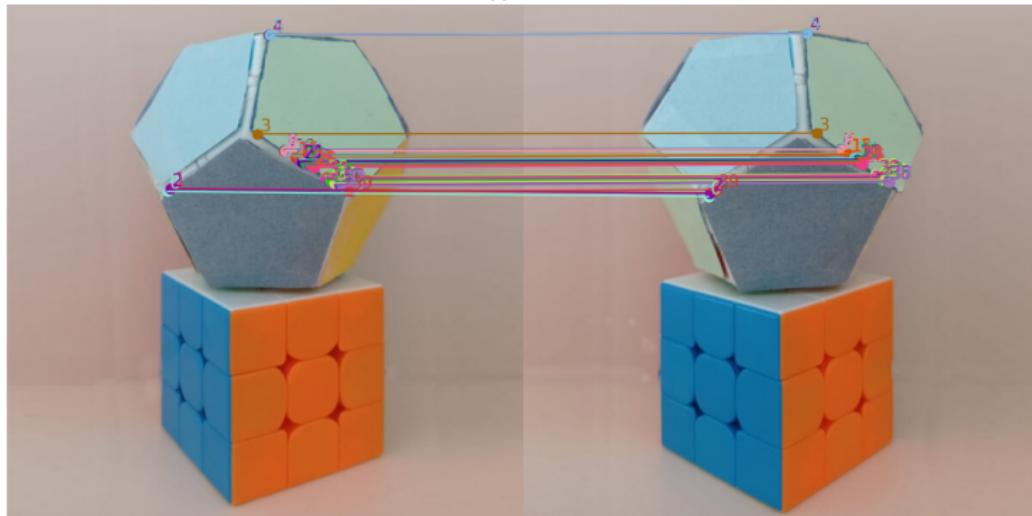
Resultat : Filtrage epipolaire

### Points appariés valides



## Resultat : droite epipolaire + BRIEF lab

Points appariés valides



## Résultats pré-traitement



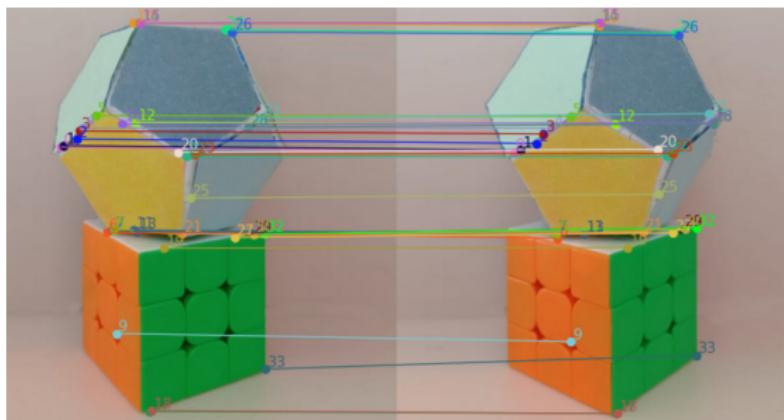
Points gardés

► constantes

# Résultats pré-traitement



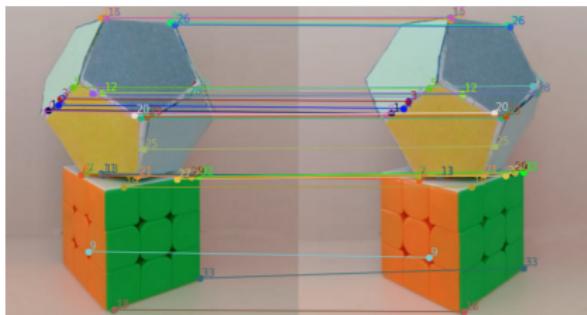
Points gardés



Appariement

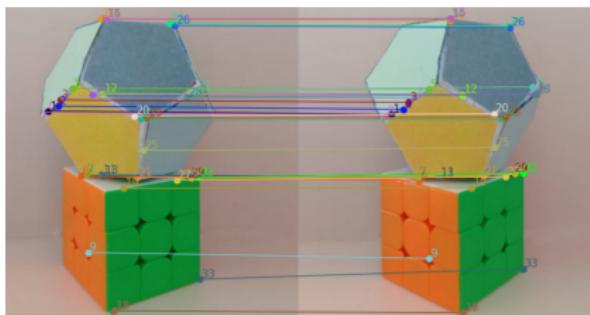
► constantes

## comparaison avec SIFT opencv

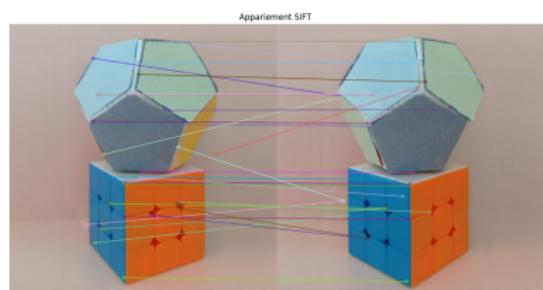


## Algorithme personnalisé

## comparaison avec SIFT opencv



## Algorithme personnalisé



SIFT

# Plan

1. Selection

2. Appariement

3. Calibration

Système de calibrage

En pratiques

Resolution système homogène

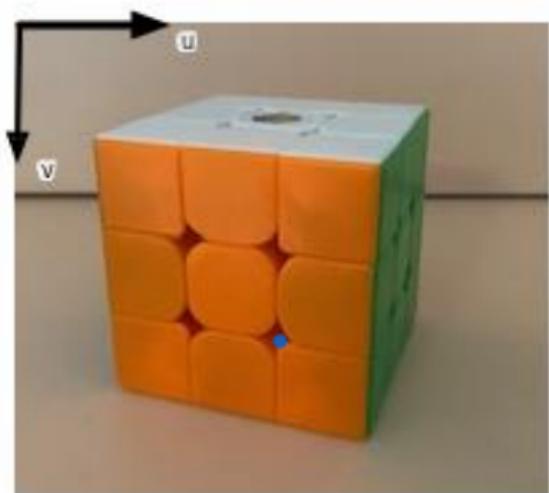
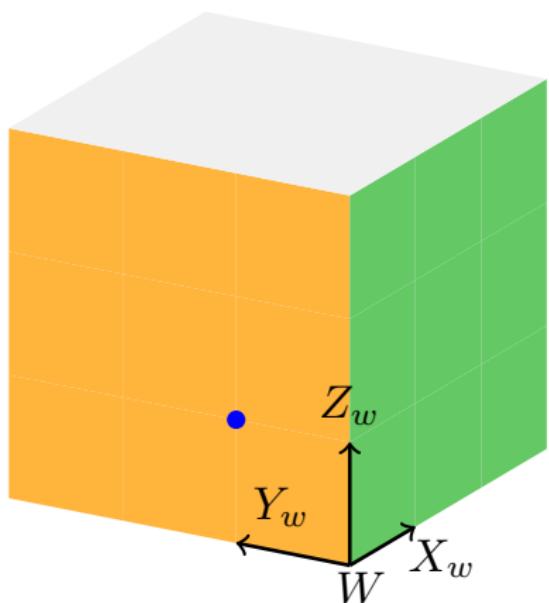
Implémentation

4. Reconstruction des points

5. Enveloppe Convexe

6. Analyse des résultats

# Les différents repères



Représentation du cube (vue 3D)

Cube sur une image

## Modèle de projection — Matrice $P$

- ▶ On considère un point 3D  $M = (X, Y, Z)$

# Modèle de projection — Matrice $P$

- ▶ On considère un point 3D  $M = (X, Y, Z)$
- ▶ Il se projette sur un point image  $m = (u, v)$

Compléments projections

## Modèle de projection — Matrice $P$

- ▶ On considère un point 3D  $M = (X, Y, Z)$
- ▶ Il se projette sur un point image  $m = (u, v)$  Compléments projections
- ▶ On cherche une relation linéaire homogène (pourquoi ?):

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} P = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

## Modèle de projection — Matrice $P$

- ▶ On considère un point 3D  $M = (X, Y, Z)$
- ▶ Il se projette sur un point image  $m = (u, v)$  Compléments projections
- ▶ On cherche une relation linéaire homogène (pourquoi ?):

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} P = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

- ▶  $P$  est une matrice  $3 \times 4$ , avec 12 inconnues

# Modèle de projection — Matrice $P$

- ▶ On considère un point 3D  $M = (X, Y, Z)$
- ▶ Il se projette sur un point image  $m = (u, v)$  Compléments projections
- ▶ On cherche une relation linéaire homogène (pourquoi ?):

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} P = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

- ▶  $P$  est une matrice  $3 \times 4$ , avec 12 inconnues
- ▶ En développant les lignes :

$$\lambda u = p_{11}X + p_{12}Y + p_{13}Z + p_{14}$$

$$\lambda v = p_{21}X + p_{22}Y + p_{23}Z + p_{24}$$

$$\lambda = p_{31}X + p_{32}Y + p_{33}Z + p_{34}$$

# Équations sans $\lambda$ et système matriciel

- ▶ Pour un point donné, on élimine  $\lambda$  :

$$u(p_{31}X + p_{32}Y + p_{33}Z + p_{34}) = p_{11}X + p_{12}Y + p_{13}Z + p_{14}$$

$$v(p_{31}X + p_{32}Y + p_{33}Z + p_{34}) = p_{21}X + p_{22}Y + p_{23}Z + p_{24}$$

# Équations sans $\lambda$ et système matriciel

- ▶ Pour un point donné, on élimine  $\lambda$  :

$$u(p_{31}X + p_{32}Y + p_{33}Z + p_{34}) = p_{11}X + p_{12}Y + p_{13}Z + p_{14}$$

$$v(p_{31}X + p_{32}Y + p_{33}Z + p_{34}) = p_{21}X + p_{22}Y + p_{23}Z + p_{24}$$

- ▶ Cela donne un système homogène ...

$$\begin{cases} 0 = p_{11}x_C + p_{12}y_C + p_{13}z_C + p_{14} - p_{31}ux_C - p_{32}uz_C - p_{33}uz_C - p_{34}u \\ 0 = p_{21}x_C + p_{22}y_C + p_{23}z_C + p_{24} - p_{31}vx_C - p_{32}vz_C - p_{33}vz_C - p_{34}v \end{cases}$$

# Équations sans $\lambda$ et système matriciel

- ▶ Pour un point donné, on élimine  $\lambda$  :

$$u(p_{31}X + p_{32}Y + p_{33}Z + p_{34}) = p_{11}X + p_{12}Y + p_{13}Z + p_{14}$$

$$v(p_{31}X + p_{32}Y + p_{33}Z + p_{34}) = p_{21}X + p_{22}Y + p_{23}Z + p_{24}$$

- ▶ Cela donne un système homogène ...

$$\begin{cases} 0 = p_{11}x_C + p_{12}y_C + p_{13}z_C + p_{14} - p_{31}ux_C - p_{32}uy_C - p_{33}uz_C - p_{34}u \\ 0 = p_{21}x_C + p_{22}y_C + p_{23}z_C + p_{24} - p_{31}vx_C - p_{32}vy_C - p_{33}vz_C - p_{34}v \end{cases}$$

- ▶ En faisant cela pour n points on obtient un système ...

# Équations sans $\lambda$ et système matriciel

$$\begin{pmatrix} x_C^{(1)} & y_C^{(1)} & z_C^{(1)} & 1 & 0 & 0 & 0 & 0 & -u^{(1)}x_C^{(1)} & -u^{(1)}y_C^{(1)} & -u^{(1)}z_C^{(1)} & -u^{(1)} \\ 0 & 0 & 0 & 0 & x_C^{(1)} & y_C^{(1)} & z_C^{(1)} & 1 & -v^{(1)}x_C^{(1)} & -v^{(1)}y_C^{(1)} & -v^{(1)}z_C^{(1)} & -v^{(1)} \\ \vdots & \vdots \\ x_C^{(i)} & y_C^{(i)} & z_C^{(i)} & 1 & 0 & 0 & 0 & 0 & -u^{(i)}x_C^{(i)} & -u^{(i)}y_C^{(i)} & -u^{(i)}z_C^{(i)} & -u^{(i)} \\ 0 & 0 & 0 & 0 & x_C^{(i)} & y_C^{(i)} & z_C^{(i)} & 1 & -v^{(i)}x_C^{(i)} & -v^{(i)}y_C^{(i)} & -v^{(i)}z_C^{(i)} & -v^{(i)} \\ \vdots & \vdots \\ x_C^{(6)} & y_C^{(6)} & z_C^{(6)} & 1 & 0 & 0 & 0 & 0 & -u^{(6)}x_C^{(6)} & -u^{(6)}y_C^{(6)} & -u^{(6)}z_C^{(6)} & -u^{(6)} \\ 0 & 0 & 0 & 0 & x_C^{(6)} & y_C^{(6)} & z_C^{(6)} & 1 & -v^{(6)}x_C^{(6)} & -v^{(6)}y_C^{(6)} & -v^{(6)}z_C^{(6)} & -v^{(6)} \end{pmatrix} = \begin{pmatrix} p_{11} \\ p_{12} \\ p_{13} \\ p_{14} \\ p_{21} \\ p_{22} \\ p_{23} \\ p_{24} \\ p_{31} \\ p_{32} \\ p_{33} \\ p_{34} \end{pmatrix}$$

# Calibration



Figure: Cube calibrage

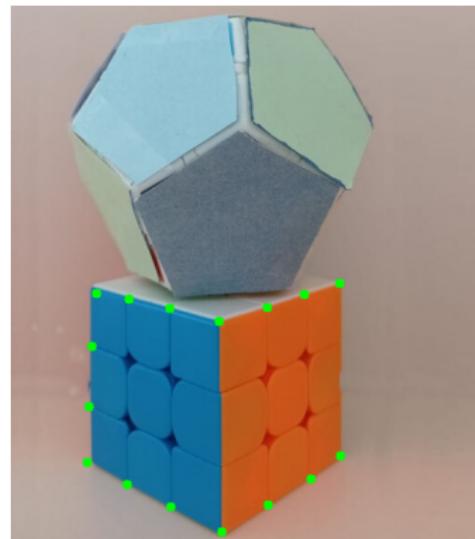
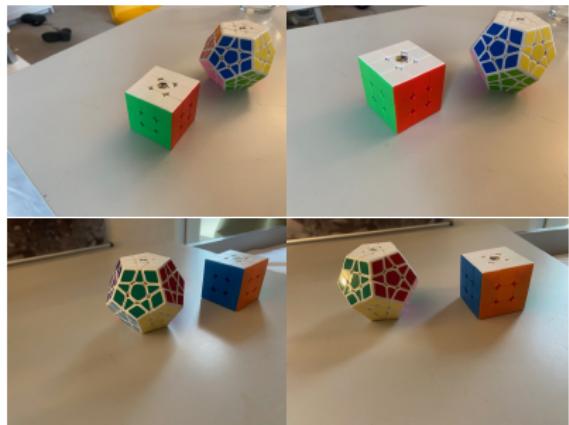
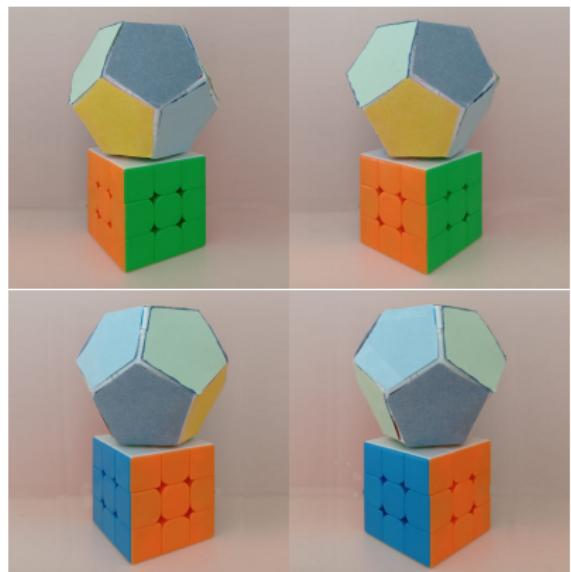


Figure: Selection des points

## Shooting photo : importance de la prise de vue



Vues initiales



Vues améliorées

# Équations sans $\lambda$ et système matriciel

$$\begin{pmatrix} x_C^{(1)} & y_C^{(1)} & z_C^{(1)} & 1 & 0 & 0 & 0 & 0 & -u^{(1)}x_C^{(1)} & -u^{(1)}y_C^{(1)} & -u^{(1)}z_C^{(1)} & -u^{(1)} \\ 0 & 0 & 0 & 0 & x_C^{(1)} & y_C^{(1)} & z_C^{(1)} & 1 & -v^{(1)}x_C^{(1)} & -v^{(1)}y_C^{(1)} & -v^{(1)}z_C^{(1)} & -v^{(1)} \\ \vdots & \vdots \\ x_C^{(i)} & y_C^{(i)} & z_C^{(i)} & 1 & 0 & 0 & 0 & 0 & -u^{(i)}x_C^{(i)} & -u^{(i)}y_C^{(i)} & -u^{(i)}z_C^{(i)} & -u^{(i)} \\ 0 & 0 & 0 & 0 & x_C^{(i)} & y_C^{(i)} & z_C^{(i)} & 1 & -v^{(i)}x_C^{(i)} & -v^{(i)}y_C^{(i)} & -v^{(i)}z_C^{(i)} & -v^{(i)} \\ \vdots & \vdots \\ x_C^{(6)} & y_C^{(6)} & z_C^{(6)} & 1 & 0 & 0 & 0 & 0 & -u^{(6)}x_C^{(6)} & -u^{(6)}y_C^{(6)} & -u^{(6)}z_C^{(6)} & -u^{(6)} \\ 0 & 0 & 0 & 0 & x_C^{(6)} & y_C^{(6)} & z_C^{(6)} & 1 & -v^{(6)}x_C^{(6)} & -v^{(6)}y_C^{(6)} & -v^{(6)}z_C^{(6)} & -v^{(6)} \end{pmatrix} = \begin{pmatrix} p_{11} \\ p_{12} \\ p_{13} \\ p_{14} \\ p_{21} \\ p_{22} \\ p_{23} \\ p_{24} \\ p_{31} \\ p_{32} \\ p_{33} \\ p_{34} \end{pmatrix}$$

## Problème d'optimisation sous contrainte

Solution triviale  $P = 0$ . On impose :  $\|p\|^2 = 1$

On reformule le problème comme une minimisation sous contrainte :

$$\min_{\|p\|=1} \|Ap\|^2 \Leftrightarrow \min_{\|p\|=1} p^T A^T A p$$

## Problème d'optimisation sous contrainte

Solution triviale  $P = 0$ . On impose :  $\|p\|^2 = 1$

On reformule le problème comme une minimisation sous contrainte :

$$\min_{\|p\|=1} \|Ap\|^2 \Leftrightarrow \min_{\|p\|=1} p^T A^T A p$$

**Propriété clé** : au minimum,  $p$  vérifie :

$$A^T A p = \lambda p$$

Compléments calcul

## Lien avec les valeurs propres

- ▶  $A^T A$  est symétrique : ses vecteurs propres forment une base.

## Lien avec les valeurs propres

- ▶  $A^T A$  est symétrique : ses vecteurs propres forment une base.
- ▶ La solution  $p$  est le vecteur propre associé à la plus petite valeur propre.

## Lien avec les valeurs propres

- ▶  $A^T A$  est symétrique : ses vecteurs propres forment une base.
- ▶ La solution  $p$  est le vecteur propre associé à la plus petite valeur propre.
- ▶ Cela minimise  $\|Ap\|$ , donc l'erreur de projection.

$$\min \|Ap\| \Rightarrow p = \text{vecteur propre de plus petite valeur propre de } A^T A$$

## Résolution par SVD

- On factorise  $A$  :

$$A = U\Sigma V^T$$

## Résolution par SVD

- ▶ On factorise  $A$  :

$$A = U\Sigma V^T$$

- ▶ La solution  $p$  est la dernière colonne de  $V$ , associée à la plus petite valeur singulière  $\sigma_n$ .

$$p = v_n \quad \text{tel que} \quad \sigma_n = \min \Sigma$$

# Décomposition SVD : idée générale

## 1. Produit symétrique :

$A^T A$  est symétrique, de taille  $n \times n$

Pseudo code

### Étapes principales :

- ▶ 1. Calculer  $A^T A$

On peut chercher ses valeurs propres  
via QR.

# Décomposition SVD : idée générale

## 2. Algorithme QR :

$$A_k = R_k Q_k \Rightarrow A_{k+1} = Q_k^T A_k Q_k$$

Pseudo code

### Étapes principales :

- ▶ 1. Calculer  $A^T A$
- ▶ 2. Appliquer QR à  $A^T A$

Converge vers matrice diagonale conservant  $\sigma_i^2$ .

# Décomposition SVD : idée générale

## 3. Valeurs singulières :

$$\sigma_i = \sqrt{\lambda_i}, \quad \text{avec } \lambda_i \text{ valeur propre}$$

Pseudo code

### Étapes principales :

- ▶ 1. Calculer  $A^T A$
- ▶ 2. Appliquer QR à  $A^T A$
- ▶ 3. Extraire valeurs propres  
 $\sigma_i^2$

On range les  $\sigma_i$  du plus grand au plus petit.

# Décomposition SVD : idée générale

## 4. Vecteurs singuliers :

$$v_i \text{ propre de } A^T A \Rightarrow u_i = \frac{1}{\sigma_i} A v_i$$

Pseudo code

### Étapes principales :

- ▶ 1. Calculer  $A^T A$
- ▶ 2. Appliquer QR à  $A^T A$
- ▶ 3. Extraire valeurs propres  
 $\sigma_i^2$
- ▶ 4. Déduire  $V$  puis  
 $U = \frac{1}{\sigma} A v$

On normalise chaque  $u_i$  pour obtenir  $U$ .

# Plan

1. Selection
2. Appariement
3. Calibration
4. Reconstruction des points
5. Enveloppe Convexe
6. Analyse des résultats

4- Reconstruction des points

## Reconstruction : retrouver les coordonnées 3D

- ▶ On connaît les matrices de projection  $P_1$  et  $P_2$

## 4- Reconstruction des points

## Reconstruction : retrouver les coordonnées 3D

- ▶ On connaît les matrices de projection  $P_1$  et  $P_2$
- ▶ Deux points image correspondants :

$$x_1 = (u_1, v_1), \quad x_2 = (u_2, v_2)$$

## 4- Reconstruction des points

## Reconstruction : retrouver les coordonnées 3D

- ▶ On connaît les matrices de projection  $P_1$  et  $P_2$

- ▶ Deux points image correspondants :

$$x_1 = (u_1, v_1), \quad x_2 = (u_2, v_2)$$

- ▶ Inconnue : coordonnées 3D  $X, Y, Z$

## 4- Reconstruction des points

## Reconstruction : retrouver les coordonnées 3D

- ▶ On connaît les matrices de projection  $P_1$  et  $P_2$

- ▶ Deux points image correspondants :

$$x_1 = (u_1, v_1), \quad x_2 = (u_2, v_2)$$

- ▶ Inconnue : coordonnées 3D  $X, Y, Z$

- ▶ On a :

$$\lambda_1 x_1 = P_1 X, \quad \lambda_2 x_2 = P_2 X$$

$$\left\{ \begin{array}{l} \lambda_1 u_1 = p_{11}^1 x_C + \cdots + p_{14}^1 \\ \lambda_1 v_1 = p_{21}^1 x_C + \cdots + p_{24}^1 \\ \lambda_1 = p_{31}^1 x_C + \cdots + p_{34}^1 \\ \lambda_2 u_2 = p_{11}^2 x_C + \cdots + p_{14}^2 \\ \lambda_2 v_2 = p_{21}^2 x_C + \cdots + p_{24}^2 \\ \lambda_2 = p_{31}^2 x_C + \cdots + p_{34}^2 \end{array} \right.$$

## 4- Reconstruction des points

## Reconstruction : retrouver les coordonnées 3D

- ▶ On connaît les matrices de projection  $P_1$  et  $P_2$

- ▶ Deux points image correspondants :

$$x_1 = (u_1, v_1), \quad x_2 = (u_2, v_2)$$

- ▶ Inconnue : coordonnées 3D  $X, Y, Z$

- ▶ On a :

$$\lambda_1 x_1 = P_1 X, \quad \lambda_2 x_2 = P_2 X$$

- ▶ Système de 6 équations linéaires homogènes

$$\left\{ \begin{array}{l} \lambda_1 u_1 = p_{11}^1 x_C + \cdots + p_{14}^1 \\ \lambda_1 v_1 = p_{21}^1 x_C + \cdots + p_{24}^1 \\ \lambda_1 = p_{31}^1 x_C + \cdots + p_{34}^1 \\ \lambda_2 u_2 = p_{11}^2 x_C + \cdots + p_{14}^2 \\ \lambda_2 v_2 = p_{21}^2 x_C + \cdots + p_{24}^2 \\ \lambda_2 = p_{31}^2 x_C + \cdots + p_{34}^2 \end{array} \right.$$

## 4- Reconstruction des points

## Reconstruction : retrouver les coordonnées 3D

- ▶ On connaît les matrices de projection  $P_1$  et  $P_2$

- ▶ Deux points image correspondants :

$$x_1 = (u_1, v_1), \quad x_2 = (u_2, v_2)$$

- ▶ Inconnue : coordonnées 3D  $X, Y, Z$

- ▶ On a :

$$\lambda_1 x_1 = P_1 X, \quad \lambda_2 x_2 = P_2 X$$

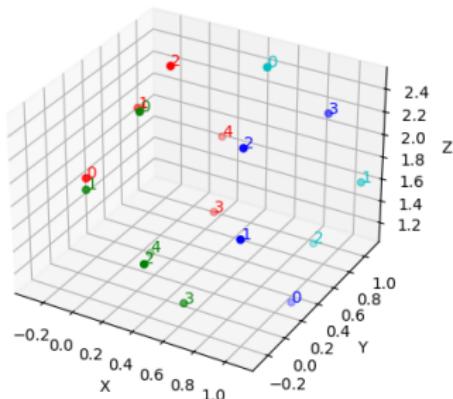
- ▶ Système de 6 équations linéaires homogènes
- ▶ Résolution par SVD

$$\left\{ \begin{array}{l} \lambda_1 u_1 = p_{11}^1 x_C + \cdots + p_{14}^1 \\ \lambda_1 v_1 = p_{21}^1 x_C + \cdots + p_{24}^1 \\ \lambda_1 = p_{31}^1 x_C + \cdots + p_{34}^1 \\ \lambda_2 u_2 = p_{11}^2 x_C + \cdots + p_{14}^2 \\ \lambda_2 v_2 = p_{21}^2 x_C + \cdots + p_{24}^2 \\ \lambda_2 = p_{31}^2 x_C + \cdots + p_{34}^2 \end{array} \right.$$

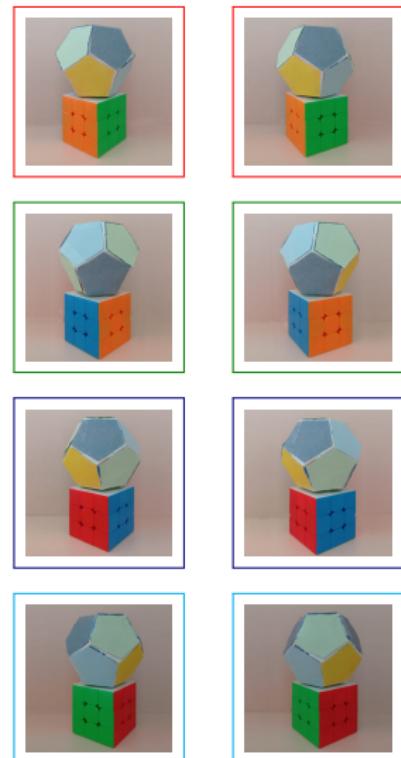
4- Reconstruction des points

# Reconstruction 3D multi-vues

Points 3D



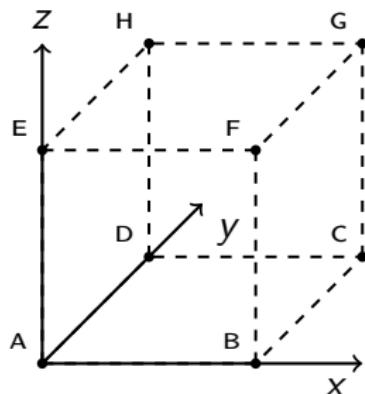
Nuage de points



# Plan

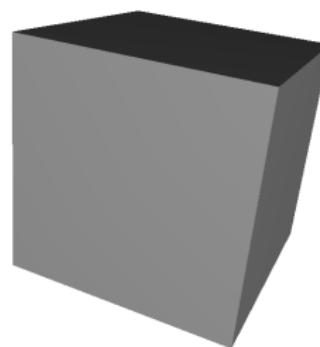
1. Selection
2. Appariement
3. Calibration
4. Reconstruction des points
5. Enveloppe Convexe  
Définition du problème
6. Analyse des résultats

# Définition du problème



# Le format STL

```
solid cube
facet normal 0 0 1 // Face supérieure
outer loop
    vertex 0 0 1
    vertex 1 0 1
    vertex 0 1 1
endloop
endfacet
facet normal 0 0 1
outer loop
    vertex 1 0 1
    vertex 1 1 1
    vertex 0 1 1
endloop
endfacet
// Autres faces...
endsolid cube
```

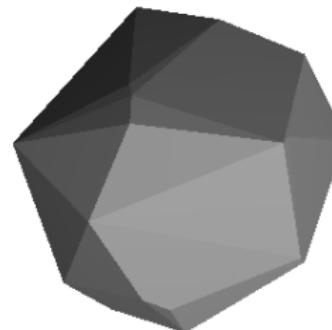
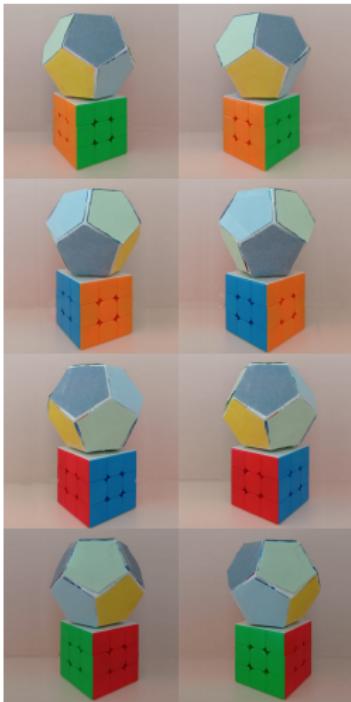


fichier STL visualisé avec [viewstl.com](http://viewstl.com)

# Plan

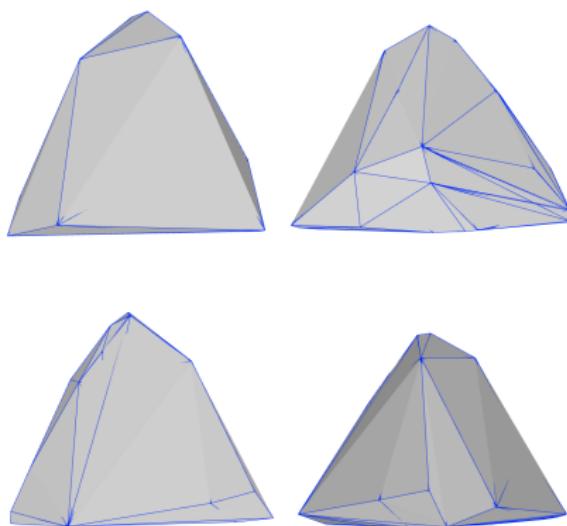
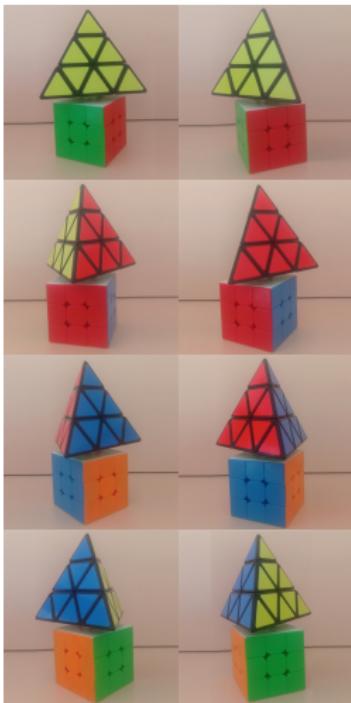
1. Selection
2. Appariement
3. Calibration
4. Reconstruction des points
5. Enveloppe Convexe
6. Analyse des résultats
  - Quelques exemples
  - Des pistes d'amélioration
  - Limite de la méthode

## Le dodécahèdre



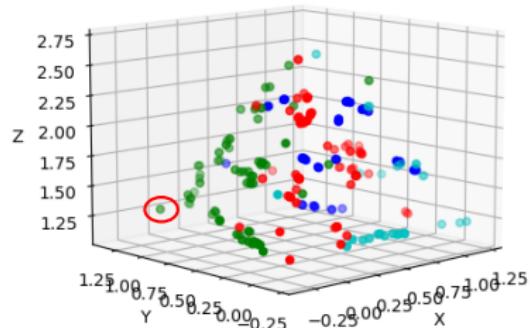
*visualisation sur viewstl.com*

# La pyramide

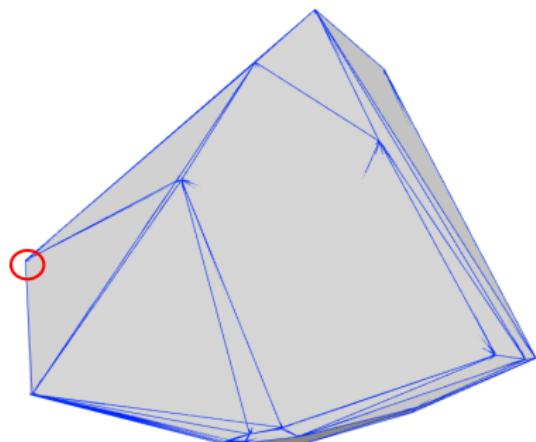


*visualisation sur 3dviewer.net*

# Une importante sensibilité aux erreurs



*points localisés par le programme*



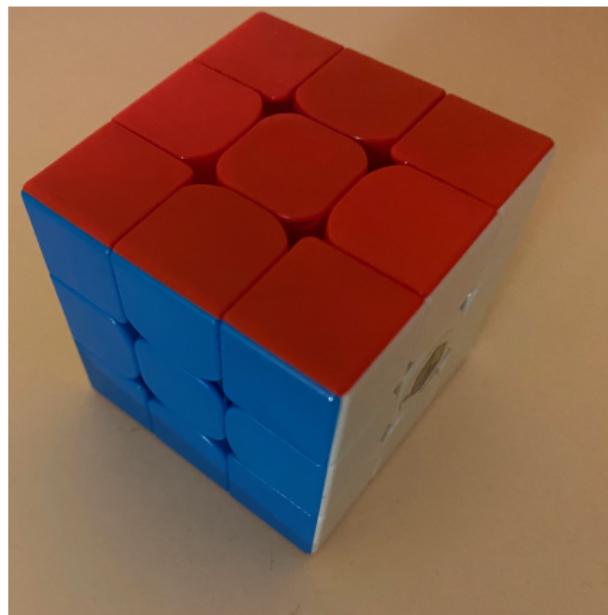
*enveloppe convexe visualisé avec  
3dview.net*

7- [

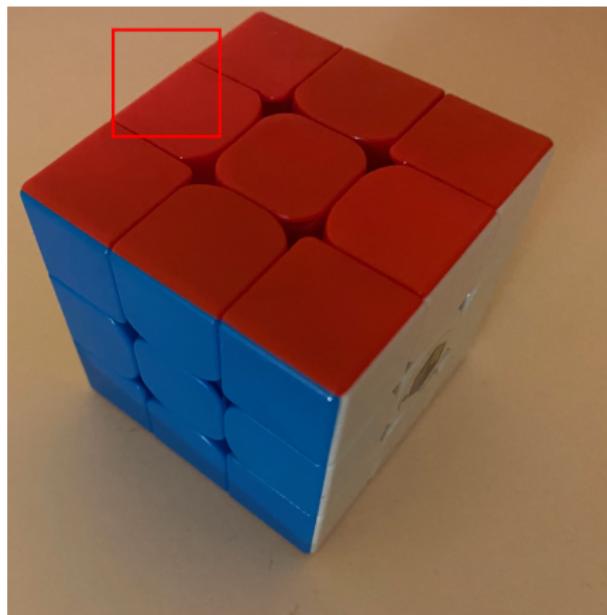
## Appendix]Appendix

7- [

## Exemple Moravec



## Exemple Moravec



7- [

23	26	29	28	70
25	21	22	32	63
28	24	23	65	65
63	66	64	64	68
73	74	77	69	69

$$w = 2$$

7- [

23	26	29	28	70
25	21	22	32	63
28	24	23	65	65
63	66	64	64	68
73	74	77	69	69

$$w = 2$$

direction horizontal

$$dx = 1, \ dy = 0$$

pixel considéré

pixel comparé

$$i = -2$$

$$S = 28 \quad S^2 = 784$$

7- [

23	26	29	28	70
25	21	22	32	63
28	24	23	65	65
63	66	64	64	68
73	74	77	69	69

$$w = 2$$

direction horizontal

$$dx = 1, dy = 0$$

pixel considéré

pixel comparé

$$i = -1$$

$$S = 52 \quad S^2 = 1248$$

7- [

23	26	29	28	70
25	21	22	32	63
28	24	23	65	65
63	66	64	64	68
73	74	77	69	69

$$w = 2$$

direction horizontal

$$dx = 1, \ dy = 0$$

pixel considéré

pixel comparé

$$i = 2$$

$$S = 205 \quad S^2 = 10339$$

$$Var(1, 0) = 386.8$$

7- [

23	26	29	28	70
25	21	22	32	63
28	24	23	65	65
63	66	64	64	68
73	74	77	69	69

$$w = 2$$

$$T = 300$$

$$\text{Var}(1,0)=386.8$$

$$\text{Var}(0,1)=526.8$$

$$\text{Var}(1,1)=439.7$$

$$\text{Var}(1,-1)=471.2$$

$$S = 386.8$$

$S > T \implies$  point d'intérêt

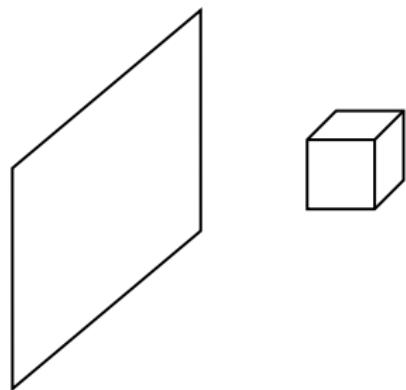
7- [

Voir le schéma détaillé

7- [

7- [

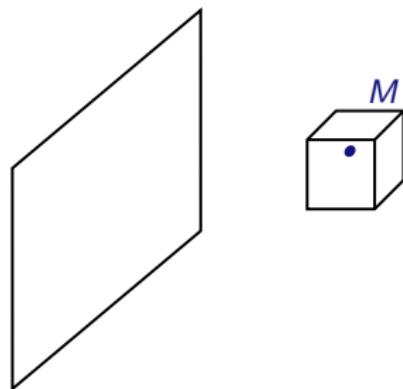
## Les différents repères



7- [

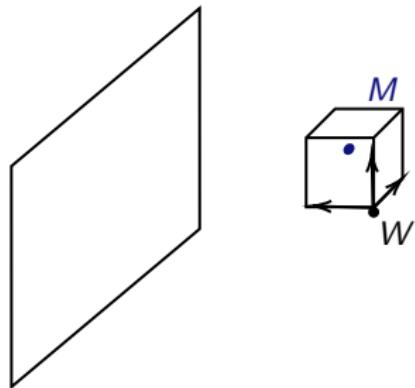
## Les différents repères

►  $M$  : point réel



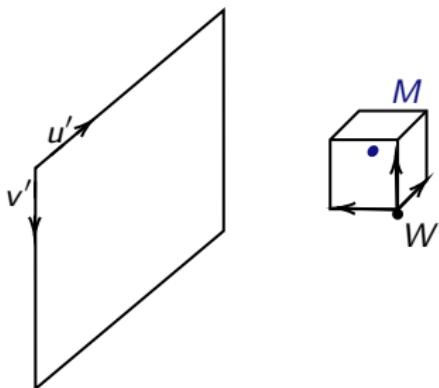
7- [

## Les différents repères



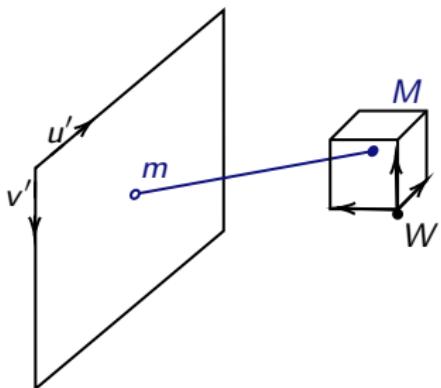
- ▶  $M$  : point réel
- ▶  $W$  : origine du repère du monde

## Les différents repères



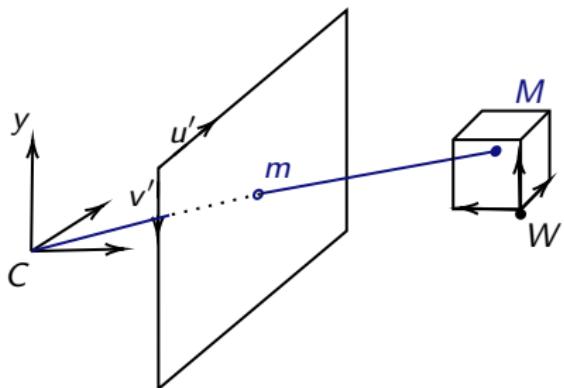
- ▶  $M$  : point réel
- ▶  $W$  : origine du repère du monde
- ▶  $(u', v')$  : coordonnées dans le plan image en pixels

## Les différents repères



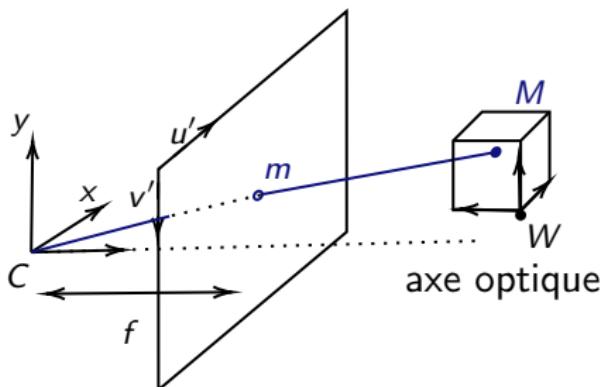
- ▶  $M$  : point réel
- ▶  $W$  : origine du repère du monde
- ▶  $(u', v')$  : coordonnées dans le plan image en pixels
- ▶  $m$  : projection de  $M$  dans le plan image

## Les différents repères



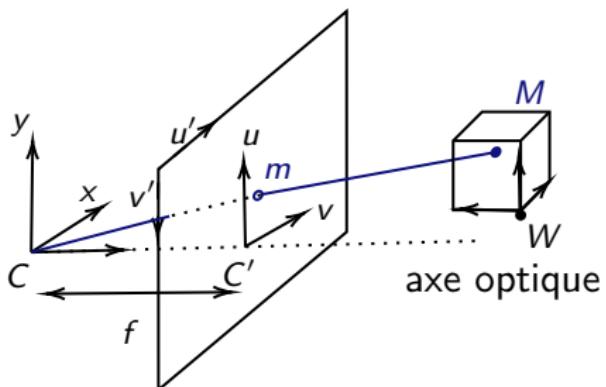
- ▶  $M$  : point réel
- ▶  $W$  : origine du repère du monde
- ▶  $(u', v')$  : coordonnées dans le plan image en pixels
- ▶  $m$  : projection de  $M$  dans le plan image
- ▶  $C$  : origine du repère de la caméra

## Les différents repères



- ▶  $M$  : point réel
- ▶  $W$  : origine du repère du monde
- ▶  $(u', v')$  : coordonnées dans le plan image en pixels
- ▶  $m$  : projection de  $M$  dans le plan image
- ▶  $C$  : origine du repère de la caméra

## Les différents repères



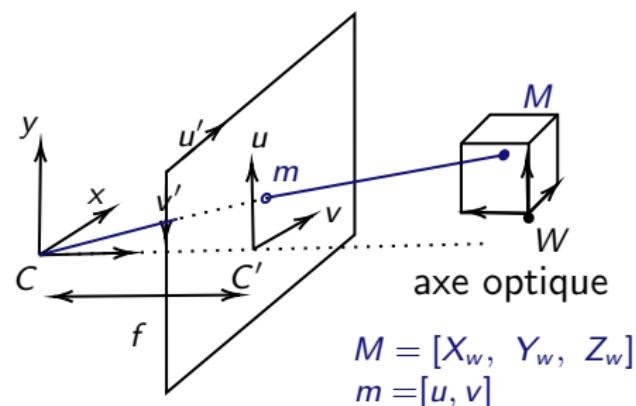
- ▶  $M$  : point réel
- ▶  $W$  : origine du repère du monde
- ▶  $(u', v')$  : coordonnées dans le plan image en pixels
- ▶  $m$  : projection de  $M$  dans le plan image
- ▶  $C$  : origine du repère de la caméra
- ▶  $C'$  : origine du repère de l'image par projection de  $C$

7- [

# Projection d'un point 3D sur le plan image

Par le théorème de Thalès  
 (projection perspective)

$$\begin{aligned} u &= fx_c \\ v &= fy_c \\ w &= z_c \end{aligned}$$



$$\begin{aligned} M &= [X_w, Y_w, Z_w] \\ m &= [u, v] \end{aligned}$$

7- [

# Projection d'un point 3D sur le plan image

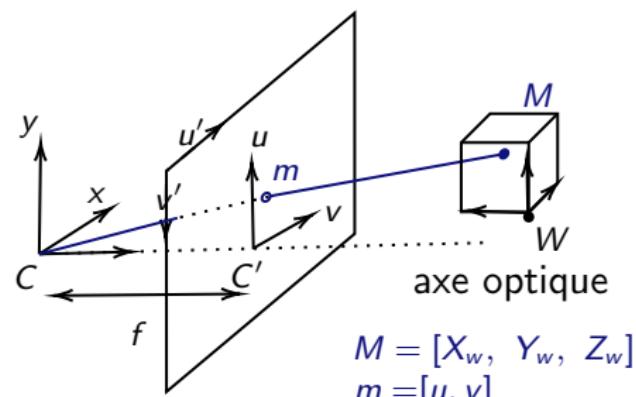
Par le théorème de Thalès  
 (projection perspective)

$$u = fx_c$$

$$v = fy_c$$

$$w = z_c$$

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix}$$



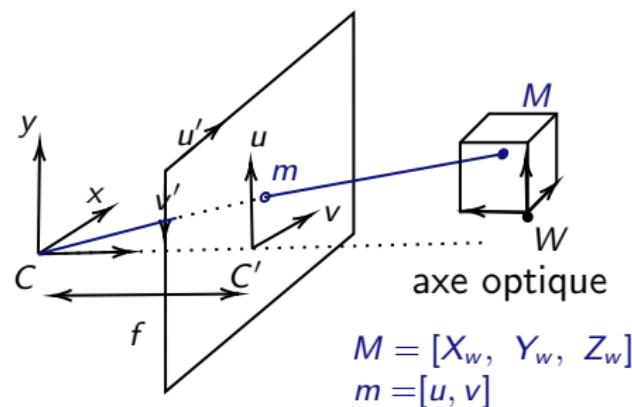
7- [

## Projection d'un point 3D sur le plan image

Le changement de repère s'écrit avec une transformation homogène :

$$\begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

Où  $R \in \mathbb{R}^{3 \times 3}$  est une rotation,  $T \in \mathbb{R}^3$  une translation.



$$M = [X_w, Y_w, Z_w]$$

$$m = [u, v]$$

7- [

## Projection d'un point 3D sur le plan image

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \underbrace{\begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\text{chaîne de projection}} \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

7- [

## Projection d'un point 3D sur le plan image

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \underbrace{\begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\text{chaîne de projection}} \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = P \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \quad \text{avec } P \in \mathcal{M}_{3 \times 4}(\mathbb{R})$$

## Les différents repères

$$\lambda_i \begin{pmatrix} u^{(i)} \\ v^{(i)} \\ 1 \end{pmatrix} = \begin{pmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{pmatrix} \begin{pmatrix} x_C^{(i)} \\ y_C^{(i)} \\ z_C^{(i)} \\ 1 \end{pmatrix}$$

7- [

## Système d'optimisation à contrainte unitaire

On souhaite résoudre le système en évitant la solution triviale  $P = 0$ .

## Système d'optimisation à contrainte unitaire

On souhaite résoudre le système en évitant la solution triviale  $P = 0$ . Sachant que la matrice  $P$  ne peut être déterminée qu'à un facteur près, on peut imposer :

$$\|P\|^2 = 1$$

et reformuler le système comme un problème d'optimisation :

## Système d'optimisation à contrainte unitaire

On souhaite résoudre le système en évitant la solution triviale  $P = 0$ . Sachant que la matrice  $P$  ne peut être déterminée qu'à un facteur près, on peut imposer :

$$\|P\|^2 = 1$$

et reformuler le système comme un problème d'optimisation :

$$\min_{\|p\|^2=1} \|Ap\|^2 = \min_{\|p\|^2=1} p^T A^T A p$$

## Système d'optimisation à contrainte unitaire

On souhaite résoudre le système en évitant la solution triviale  $P = 0$ . Sachant que la matrice  $P$  ne peut être déterminée qu'à un facteur près, on peut imposer :

$$\|P\|^2 = 1$$

et reformuler le système comme un problème d'optimisation :

$$\min_{\|p\|^2=1} \|Ap\|^2 = \min_{\|p\|^2=1} p^T A^T A p$$

On introduit les fonctions :

- ▶  $f(p) = p^T A^T A p$
- ▶  $g(p) = p^T p - 1$

## Système d'optimisation à contrainte unitaire

On souhaite résoudre le système en évitant la solution triviale  $P = 0$ . Sachant que la matrice  $P$  ne peut être déterminée qu'à un facteur près, on peut imposer :

$$\|P\|^2 = 1$$

et reformuler le système comme un problème d'optimisation :

$$\min_{\|p\|^2=1} \|Ap\|^2 = \min_{\|p\|^2=1} p^T A^T A p$$

On introduit les fonctions :

- $f(p) = p^T A^T A p$
- $g(p) = p^T p - 1$

D'après le théorème d'optimisation sous contrainte (Lagrange), au point optimal  $P^*$ , il existe  $\lambda \in \mathbb{R}$  tel que :

$$\nabla f(P^*) = \lambda \nabla g(P^*)$$

7- [

## Lien avec les valeurs propres

Posons  $M = A^T A$ . Alors :

$$f(p) = \sum_{i=1}^n \sum_{j=1}^n p_i M_{ij} p_j$$

7- [

## Lien avec les valeurs propres

Posons  $M = A^T A$ . Alors :

$$f(p) = \sum_{i=1}^n \sum_{j=1}^n p_i M_{ij} p_j$$

Comme  $M$  est symétrique :

$$\frac{\partial f}{\partial p} = 2Mp \quad \text{et} \quad \frac{\partial g}{\partial p} = 2p$$

7- [

## Lien avec les valeurs propres

Posons  $M = A^T A$ . Alors :

$$f(p) = \sum_{i=1}^n \sum_{j=1}^n p_i M_{ij} p_j$$

Comme  $M$  est symétrique :

$$\frac{\partial f}{\partial p} = 2Mp \quad \text{et} \quad \frac{\partial g}{\partial p} = 2p$$

On a donc :

$$\frac{\partial f}{\partial p} = \lambda \frac{\partial g}{\partial p} \quad \Rightarrow \quad \boxed{A^T A p = \lambda p}$$

## Lien avec les valeurs propres

Posons  $M = A^T A$ . Alors :

$$f(p) = \sum_{i=1}^n \sum_{j=1}^n p_i M_{ij} p_j$$

Comme  $M$  est symétrique :

$$\frac{\partial f}{\partial p} = 2Mp \quad \text{et} \quad \frac{\partial g}{\partial p} = 2p$$

On a donc :

$$\frac{\partial f}{\partial p} = \lambda \frac{\partial g}{\partial p} \quad \Rightarrow \quad A^T A p = \lambda p$$

C'est une équation aux valeurs propres :

- ▶  $p$  est un vecteur propre de  $A^T A$
- ▶  $\lambda$  est la valeur propre associée

7- [

## Triangulation : formulation du système

- ▶  $P_1$  et  $P_2$  déterminées

7- [

## Triangulation : formulation du système

- ▶  $P_1$  et  $P_2$  déterminées
- ▶ On cherche les coordonnées  $X = (x_C, y_C, z_C, 1)^T$

## Triangulation : formulation du système

- ▶  $P_1$  et  $P_2$  déterminées
- ▶ On cherche les coordonnées  $X = (x_C, y_C, z_C, 1)^T$
- ▶ Pour chaque paire  $(x_1, x_2)$  de projections

## Triangulation : formulation du système

- ▶  $P_1$  et  $P_2$  déterminées
- ▶ On cherche les coordonnées  $X = (x_C, y_C, z_C, 1)^T$
- ▶ Pour chaque paire  $(x_1, x_2)$  de projections
- ▶ On élimine  $\lambda_1, \lambda_2$  et on écrit un système homogène

## Triangulation : formulation du système

- ▶  $P_1$  et  $P_2$  déterminées
- ▶ On cherche les coordonnées  $X = (x_C, y_C, z_C, 1)^T$
- ▶ Pour chaque paire  $(x_1, x_2)$  de projections
- ▶ On élimine  $\lambda_1, \lambda_2$  et on écrit un système homogène
- ▶ Système sous la forme  $AX = 0$

$$A = \begin{pmatrix} p_{31}^1 u_1 - p_{11}^1 & p_{32}^1 u_1 - p_{12}^1 & p_{33}^1 u_1 - p_{13}^1 & p_{34}^1 u_1 - p_{14}^1 \\ p_{31}^1 v_1 - p_{21}^1 & p_{32}^1 v_1 - p_{22}^1 & p_{33}^1 v_1 - p_{23}^1 & p_{34}^1 v_1 - p_{24}^1 \\ p_{31}^2 u_2 - p_{11}^2 & p_{32}^2 u_2 - p_{12}^2 & p_{33}^2 u_2 - p_{13}^2 & p_{34}^2 u_2 - p_{14}^2 \\ p_{31}^2 v_2 - p_{21}^2 & p_{32}^2 v_2 - p_{22}^2 & p_{33}^2 v_2 - p_{23}^2 & p_{34}^2 v_2 - p_{24}^2 \end{pmatrix}$$

---

## Algorithme: Décomposition QR via Gram-Schmidt

---

**Entrée:**  $A \in \mathbb{R}^{m \times n}$

**Sortie:**  $Q \in \mathbb{R}^{m \times n}$ ,  $R \in \mathbb{R}^{n \times n}$  tels que  $A = QR$

**pour**  $j \leftarrow 1$  **to**  $n$  **faire**

$v_j \leftarrow A_{:,j}$

(\*Copie de la  $j^{\text{ème}}$  colonne de  $A$ \*)

**pour**  $i \leftarrow 1$  **to**  $j - 1$  **faire**

$R_{i,j} \leftarrow \langle Q_{:,i}, A_{:,j} \rangle$

$v_j \leftarrow v_j - R_{i,j} Q_{:,i}$

$R_{j,j} \leftarrow \|v_j\|$

**si**  $R_{j,j} > \epsilon$  **alors**

$Q_{:,j} \leftarrow \frac{v_j}{R_{j,j}}$

**sinon**

$Q_{:,j} \leftarrow 0$

**retourner**  $Q, R$

---

---

## Algorithme: algorithme QR

---

**Entrée:**  $B \in \mathbb{R}^{n \times n}$  symétrique

**Sortie:**  $\Sigma^2$ ,  $V$  tels que  $B = V\Sigma^2V^T$

$Q_{\text{acc}} \leftarrow I_n$

(\*Accumule les produits de  $Q$ \*)

$\delta \leftarrow 1$ ,  $k_{\max} \leftarrow 1000$ ,  $k \leftarrow 0$

**tant que**  $\delta > 10^{-9}$  et  $k < k_{\max}$  **faire**

$Q, R \leftarrow$  décomposition QR de  $B$

$B_{\text{nouveau}} \leftarrow R \cdot Q$

$Q_{\text{acc}} \leftarrow Q_{\text{acc}} \cdot Q$

$\delta \leftarrow \sum_i |\text{diag}(B_{\text{nouveau}})_i - \text{diag}(B)_i|$

$A \leftarrow B_{\text{nouveau}}$

$k \leftarrow k + 1$

**pour**  $i = 1$  à  $n$  **faire**

**si**  $|B[i, i]| > \varepsilon$  **alors**

$\Sigma^2[i, i] \leftarrow V[i, i]$

**sinon**

$\Sigma^2[i, i] \leftarrow 0$

**retourner**  $\Sigma^2, Q_{\text{acc}}$

---

---

## Algorithme: SVD via algorithme QR sur $A^T A$

---

**Entrée:**  $A \in \mathbb{R}^{m \times n}$

**Sortie:**  $U, \Sigma, V$  tels que  $A \approx U\Sigma V^T$

$A^T \leftarrow$  transposée de  $A$

$A^T A \leftarrow A^T \cdot A$  (\*Symétrique et définie positive\*)

algorithme\_QR( $A^T A, \Sigma^2, V$ ) (\* $\Sigma^2$  diagonale,  $V$  orthogonale\*)

**pour**  $i \leftarrow 1$  **to**  $n$  **faire**

$\sigma^2 \leftarrow \Sigma^2[i, i]$

**si**  $\sigma^2 < 10^{-12}$  **alors**

        continuer (\*Ignorer valeur singulière nulle\*)

$\sigma \leftarrow \sqrt{\sigma^2}$

$\Sigma[i, i] \leftarrow \sigma$  (\*Met à jour la vraie valeur singulière\*)

$v_i \leftarrow i^{\text{e}} \text{ colonne de } V$

$u_i \leftarrow A \cdot v_i$  (\* $u_i$  non normalisé\*)

$u_i \leftarrow u_i / \sigma$

    normaliser  $u_i$

    insérer  $u_i$  comme  $i^{\text{e}}$  colonne de  $U$