

# Reconstruction d'objets convexes à partir de photographies

Présentation de [Lucie-Hélène Cuingnet](#)

Travail réalisé avec [Barnabé Baruchel](#)

TIPE 2025

## Reconstruction 3D

2025-05-19

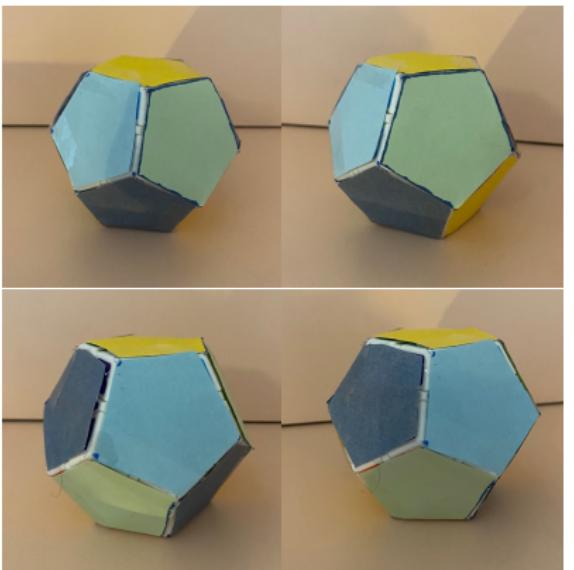
Pour ce TIPE je suis me suis donnée pour objectif de comprendre et surtout expérimenter comment on peut, à partir de simple photo, retrouver la géométrie complète d'un objet.

Reconstruction d'objets convexes à partir de photographies

Présentation de [Lucie-Hélène Cuingnet](#)  
Travail réalisé avec [Barnabé Baruchel](#)

TIPE 2025

## Définition du problème



**Données**

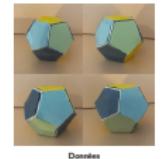
## Reconstruction 3D

2025-05-19

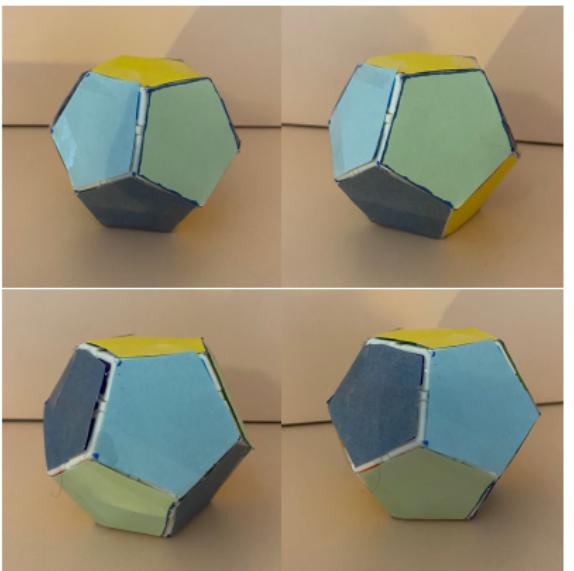
### └ Définition du problème

Ainsi, à partir d'un jeux de photo qui constituera mes données je souhaite reconstruire un fichier 3D de ce dernier prêt pour une impression 3D.

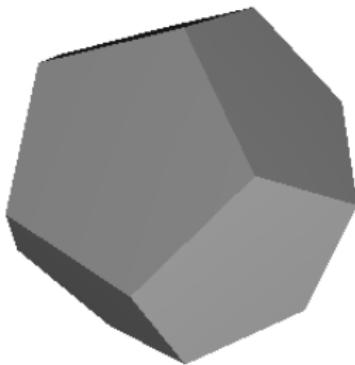
Définition du problème



## Définition du problème



Données



Objectif

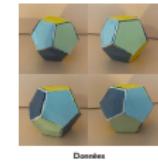
## Reconstruction 3D

2025-05-19

### └ Définition du problème

Ainsi, à partir d'un jeux de photo qui constituera mes données je souhaite reconstruire un fichier 3D de ce dernier prêt pour une impression 3D.

Définition du problème



Objectif

## 1. Selection

- Points d'intérêts
- Algorithme
- Implémentation

## 2. Appariement

## 3. Calibration

## 4. Reconstruction des points

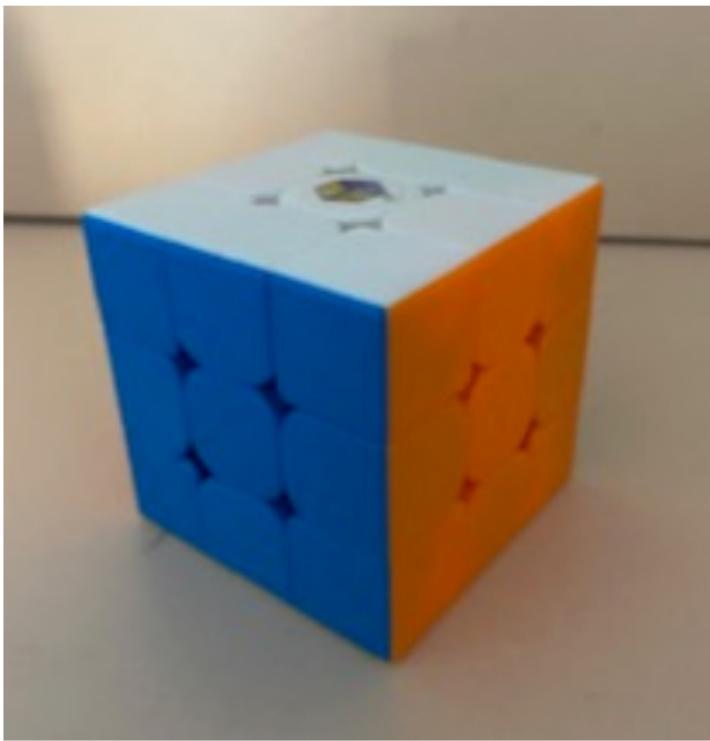
## Reconstruction 3D

### └ Selection

#### └ Plan[0.1cm]

2025-05-19

## Points d'intérêts



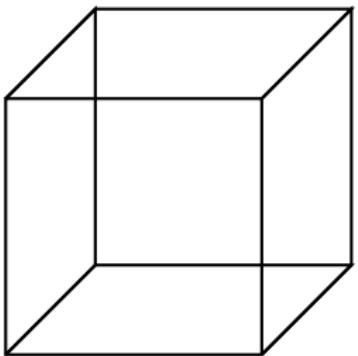
2025-05-19

- └ Selection
- └ Points d'intérêts
- └ Points d'intérêts



## Points d'intérêts

### Points d'intérêts sur un cube

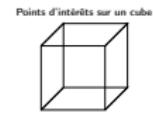


## Reconstruction 3D

- └ Selection
  - └ Points d'intérêts
    - └ Points d'intérêts

2025-05-19

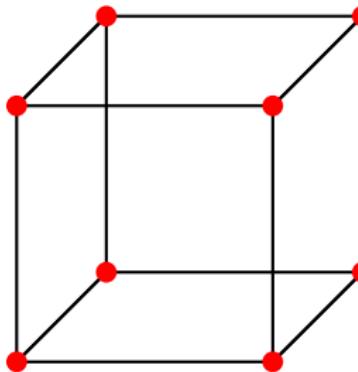
Points d'intérêts



Pour un objet convexe tel qu'un cube, il nous suffit de repérer ces sommets pour le reconstruire en 3D. Mais si l'objet est non convexe, comme dans le schéma de droite, on peut détecter des coins qui ne sont pas représentatifs de la structure globale. On voit ici que l'enveloppe convexe (en bleu) ne correspond pas exactement à la forme réelle. Ce genre de situation peut gêner certaines étapes de reconstruction.

# Points d'intérêts

## Points d'intérêts sur un cube

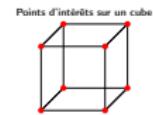


## Reconstruction 3D

2025-05-19

- └ Selection
  - └ Points d'intérêts
    - └ Points d'intérêts

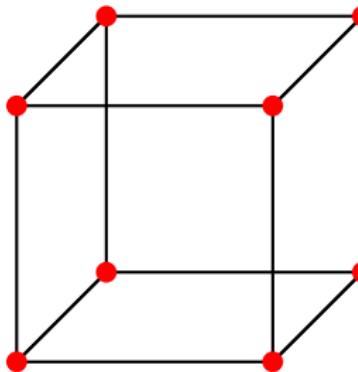
Points d'intérêts



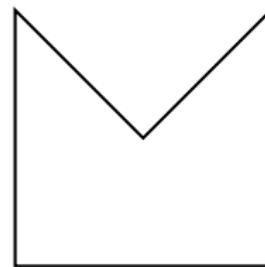
Pour un objet convexe tel qu'un cube, il nous suffit de réperer ces sommets pour le reconstruire en 3D. Mais si l'objet est non convexe, comme dans le schéma de droite, on peut détecter des coins qui ne sont pas représentatifs de la structure globale. On voit ici que l'enveloppe convexe (en bleu) ne correspond pas exactement à la forme réelle. Ce genre de situation peut gêner certaines étapes de reconstruction.

# Points d'intérêts

## Points d'intérêts sur un cube



## Problème si non convexe

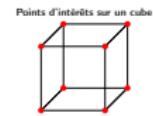


## Reconstruction 3D

2025-05-19

- └ Selection
  - └ Points d'intérêts
    - └ Points d'intérêts

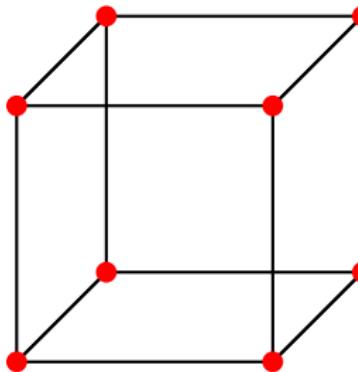
Points d'intérêts



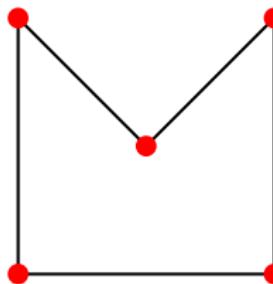
Pour un objet convexe tel qu'un cube, il nous suffit de réperer ces sommets pour le reconstruire en 3D. Mais si l'objet est non convexe, comme dans le schéma de droite, on peut détecter des coins qui ne sont pas représentatifs de la structure globale. On voit ici que l'enveloppe convexe (en bleu) ne correspond pas exactement à la forme réelle. Ce genre de situation peut gêner certaines étapes de reconstruction.

# Points d'intérêts

## Points d'intérêts sur un cube



## Problème si non convexe

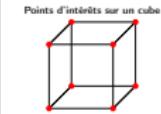


## Reconstruction 3D

2025-05-19

- └ Selection
- └ Points d'intérêts
- └ Points d'intérêts

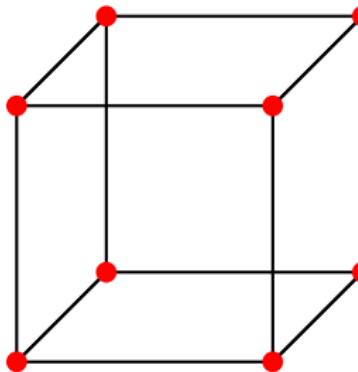
Points d'intérêts



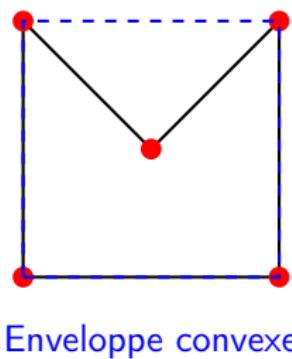
Pour un objet convexe tel qu'un cube, il nous suffit de réperer ces sommets pour le reconstruire en 3D. Mais si l'objet est non convexe, comme dans le schéma de droite, on peut détecter des coins qui ne sont pas représentatifs de la structure globale. On voit ici que l'enveloppe convexe (en bleu) ne correspond pas exactement à la forme réelle. Ce genre de situation peut gêner certaines étapes de reconstruction.

# Points d'intérêts

## Points d'intérêts sur un cube



## Problème si non convexe



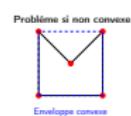
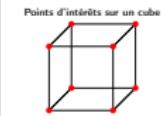
Enveloppe convexe

## Reconstruction 3D

2025-05-19

- Selection
- Points d'intérêts
- Points d'intérêts

Points d'intérêts



[Exemple](#)[code](#)**Algorithme:** Moravec (minimum des variances)**Input:** Image d'intensité image

2025-05-19

Algorithm: Moravec (minimum des variances)  
Input: Image d'intensité image

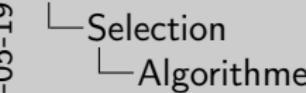
L'algorithme de Moravec est un détecteur de coins assez simple basé sur la variation locale de l'intensité. Pour chaque pixel, on mesure la variance de l'intensité dans plusieurs directions. Si la plus petite variance dépasse un certain seuil, on considère qu'on est probablement sur un coin — une zone où l'image change dans toutes les directions.

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

## Algorithme: Moravec (minimum des variances)

**Input:** Image d'intensité *image*

**Output:** Matrice des coins détecté



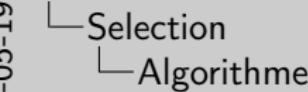
2025-05-19

Algorithmes: Moravec (minimum des variances)  
Input: Image d'intensité *image*  
Output: Matrice des coins détecté

L'algorithme de Moravec est un détecteur de coins assez simple basé sur la variation locale de l'intensité. Pour chaque pixel, on mesure la variance de l'intensité dans plusieurs directions. Si la plus petite variance dépasse un certain seuil, on considère qu'on est probablement sur un coin — une zone où l'image change dans toutes les directions.

[Exemple](#)[code](#)**Algorithme:** Moravec (minimum des variances)**Input:** Image d'intensité `image`**Output:** Matrice des coins détecté

## Reconstruction 3D



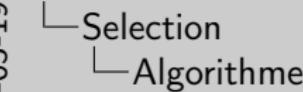
2025-05-19

Algorithme: Moravec (minimum des variances)  
Input: Image d'intensité `image`  
Output: Matrice des coins détecté

L'algorithme de Moravec est un détecteur de coins assez simple basé sur la variation locale de l'intensité. Pour chaque pixel, on mesure la variance de l'intensité dans plusieurs directions. Si la plus petite variance dépasse un certain seuil, on considère qu'on est probablement sur un coin — une zone où l'image change dans toutes les directions.

[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`

## Reconstruction 3D



2025-05-19

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

L'algorithme de Moravec est un détecteur de coins assez simple basé sur la variation locale de l'intensité. Pour chaque pixel, on mesure la variance de l'intensité dans plusieurs directions. Si la plus petite variance dépasse un certain seuil, on considère qu'on est probablement sur un coin — une zone où l'image change dans toutes les directions.

[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;

## Reconstruction 3D

  └ Selection  
    └ Algorithme

2025-05-19

Algorithmes: 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;

L'algorithme de Moravec est un détecteur de coins assez simple basé sur la variation locale de l'intensité. Pour chaque pixel, on mesure la variance de l'intensité dans plusieurs directions. Si la plus petite variance dépasse un certain seuil, on considère qu'on est probablement sur un coin — une zone où l'image change dans toutes les directions.

[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**

---

## Reconstruction 3D

└ Selection  
└ Algorithme

2025-05-19

Algorithm: 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*;

## Reconstruction 3D

- └ Selection
- └ Algorithme

2025-05-19

```

Algorithm: 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 *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  $\leftarrow \min(\text{scores})$ ;

## Reconstruction 3D

└ Selection  
  └ Algorithme

2025-05-19

```

Algorithmes: 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  $\leftarrow \min(\text{scores})$ .

```

[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*  $\leftarrow \min(\text{scores})$ ;

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

## Reconstruction 3D

- └ Selection
- └ Algorithme

2025-05-19

```

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

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

- Marquer  $(x, y)$  comme coin;

## Reconstruction 3D

- Selection
- Algorithme

2025-05-19

```

Algorithm: 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  $\leftarrow \min(\text{scores})$ ;
  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`  $\leftarrow \min(\text{scores})$ ;

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

    └ Marquer  $(x, y)$  comme coin;

**return** Liste des points marqués

## Reconstruction 3D

└ Selection  
  └ Algorithme

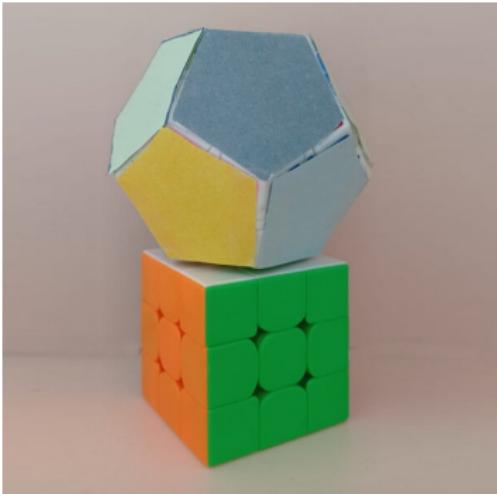
2025-05-19

```

Algorithmique: 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 toutes directions  $(dx, dy)$  parmi : verticale, horizontale, diagonales faire
    Calculer la variance locale autour de  $(x, y)$  dans la direction  $(dx, dy)$ ;
    Ajouter la variance à scores;
  score  $\leftarrow \min(\text{scores})$ ;
  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

2025-05-19

## Reconstruction 3D

- └ Selection
- └ Implémentation
- └ Fichier de sortie PBM

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

"Une fois les coins détectés, on peut produire un fichier PBM, ici représenté à droite sous forme de matrice binaire.

Un '1' correspond à un coin détecté. C'est ce fichier qui pourra être ensuite utilisé dans la suite du pipeline, notamment pour l'appariement."

## Influence des paramètres de Moravec

Paramètres utilisés :

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



## Reconstruction 3D

2025-05-19

- └ Selection
- └ Implémentation
- └ Influence des paramètres de Moravec

Influence des paramètres de Moravec

Paramètres utilisés :

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



"Voici quelques exemples de détection de coins selon différents paramètres. On voit que si on change la taille de la fenêtre ou le seuil, le résultat est très différent.

Avec une fenêtre plus grande, on est plus tolérant au bruit mais on perd en précision.

Et avec un seuil plus faible, on détecte beaucoup plus de coins, parfois à tort.

Il est donc crucial d'ajuster ces paramètres selon les images qu'on traite."

# Influence des paramètres de Moravec

Paramètres utilisés :

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



## Reconstruction 3D

Selection

Implémentation

Influence des paramètres de Moravec

2025-05-19

Influence des paramètres de Moravec

Paramètres utilisés :

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



"Voici quelques exemples de détection de coins selon différents paramètres. On voit que si on change la taille de la fenêtre ou le seuil, le résultat est très différent.

Avec une fenêtre plus grande, on est plus tolérant au bruit mais on perd en précision.

Et avec un seuil plus faible, on détecte beaucoup plus de coins, parfois à tort.

Il est donc crucial d'ajuster ces paramètres selon les images qu'on traite."

# Influence des paramètres de Moravec

Paramètres utilisés :

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



## Reconstruction 3D

2025-05-19

- └ Selection
- └ Implémentation
- └ Influence des paramètres de Moravec

Influence des paramètres de Moravec

Paramètres utilisés :

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



"Voici quelques exemples de détection de coins selon différents paramètres. On voit que si on change la taille de la fenêtre ou le seuil, le résultat est très différent.

Avec une fenêtre plus grande, on est plus tolérant au bruit mais on perd en précision.

Et avec un seuil plus faible, on détecte beaucoup plus de coins, parfois à tort.

Il est donc crucial d'ajuster ces paramètres selon les images qu'on traite."

# Influence des paramètres de Moravec

Paramètres utilisés :

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



## Reconstruction 3D

2025-05-19

└ Selection  
  └ Implémentation  
    └ Influence des paramètres de Moravec

Influence des paramètres de Moravec

Paramètres utilisés :

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



"Voici quelques exemples de détection de coins selon différents paramètres. On voit que si on change la taille de la fenêtre ou le seuil, le résultat est très différent.

Avec une fenêtre plus grande, on est plus tolérant au bruit mais on perd en précision.

Et avec un seuil plus faible, on détecte beaucoup plus de coins, parfois à tort.

Il est donc crucial d'ajuster ces paramètres selon les images qu'on traite."

# Plan

1. Selection

2. Appariement

Pré-traitement

Filtrage epipolaire

Brief

Algorithme

Resultat

3. Calibration

4. Reconstruction des points



## 2- Appariement

## Le problème de l'appariement

Reconstruction 3D  
└ Appariement

## └ Le problème de l'appariement

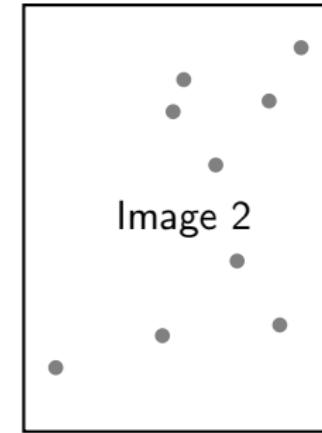
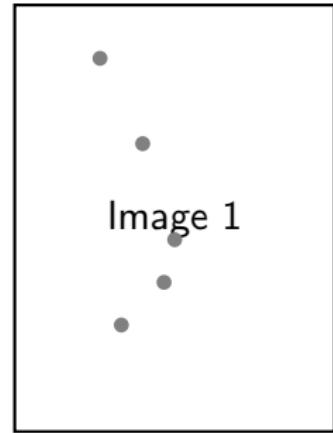
2025-05-19

Une fois les coins détectés dans deux images différentes, l'objectif est de retrouver les correspondances entre eux. C'est ce qu'on appelle l'appariement.

Le problème de l'appariement



## Le problème de l'appariement



2025-05-19

## Reconstruction 3D

- Appariement

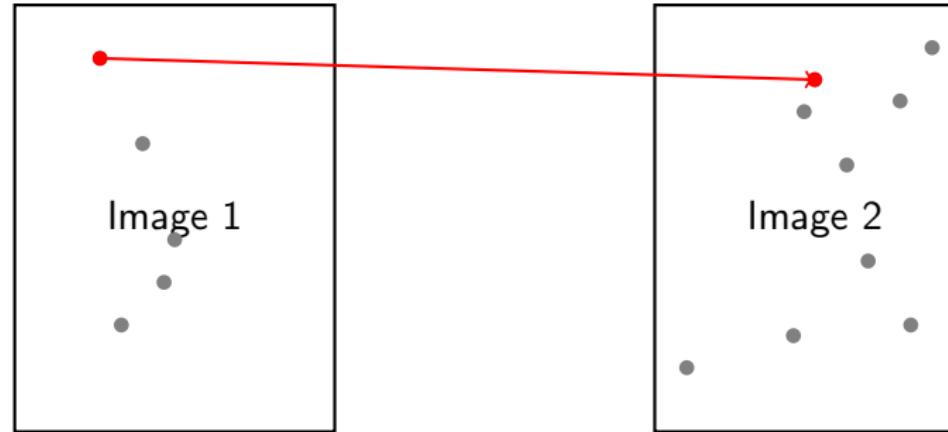
- Le problème de l'appariement

Le problème de l'appariement



Par exemple si l'on a deux images avec les points détectés comme ceci On remarque par ailleurs qu'il n'y a ni injectivité, ni surjectivité. Notre but est certe de garder un maximum de points mais le plus important est d'éviter tout erreur d'appariement qui peut complètement fausser la reconstruction dans le calcul de l'enveloppe convexe.

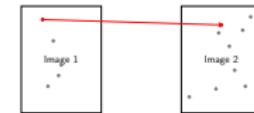
## Le problème de l'appariement



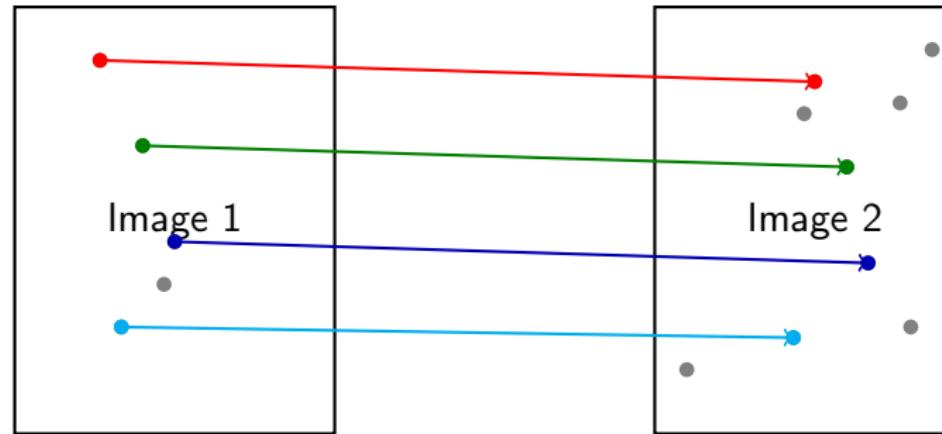
2025-05-19

### └ Le problème de l'appariement

Par exemple si l'on a deux images avec les points détectés comme ceci On remarque par ailleurs qu'il n'y a ni injectivité, ni surjectivité. Notre but est certe de garder un maximum de points mais le plus important est d'éviter tout erreur d'appariement qui peut complètement fausser la reconstruction dans le calcul de l'enveloppe convexe.



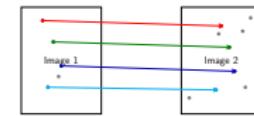
## Le problème de l'appariement



2025-05-19

### Le problème de l'appariement

Par exemple si l'on a deux images avec les points détectés comme ceci On remarque par ailleurs qu'il n'y a ni injectivité, ni surjectivité. Notre but est certe de garder un maximum de points mais le plus important est d'éviter tout erreur d'appariement qui peut complètement fausser la reconstruction dans le calcul de l'enveloppe convexe.



## Pré-traitement



2025-05-19

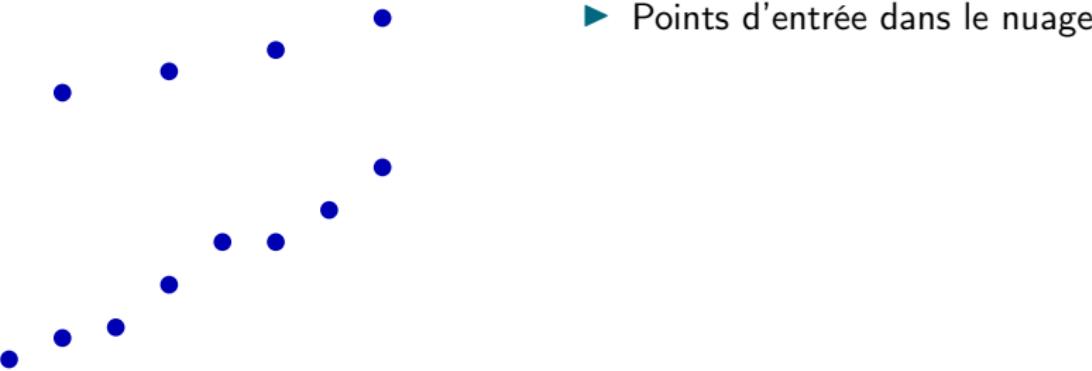
Reconstruction 3D  
└ Appariement  
  └ Pré-traitement  
    └ Pré-traitement

Pré-traitement



une première étape consiste à effectuer un traitement des points détecter par Moravec Ce traitement est réalisé à l'aide d'un premier algorithme "trouve coin" basé sur la rechercce d'extrema locaux implémenté par mon camarade Un second algorithme de type Ransac permet d'identifier les points alignés et d'éliminer ceux sur une même droite

## Ransac



► Points d'entrée dans le nuage

Reconstruction 3D  
└ Appariement  
  └ Pré-traitement  
    └ Ransac

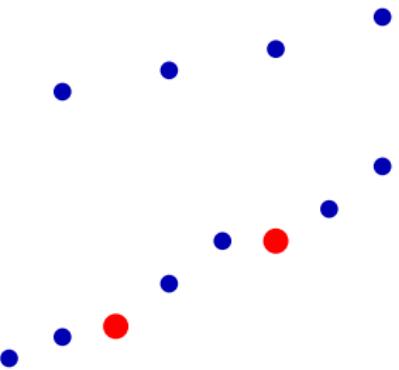
2025-05-19

Ransac



► Points d'entrée dans le nuage

## Ransac



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

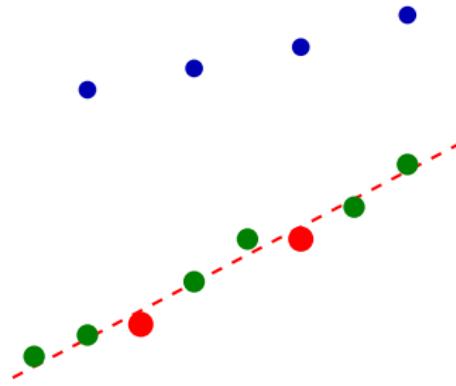
2025-05-19

Reconstruction 3D  
└ Appariement  
  └ Pré-traitement  
    └ Ransac

Ransac



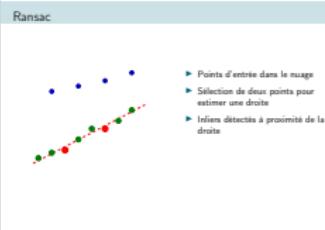
## Ransac



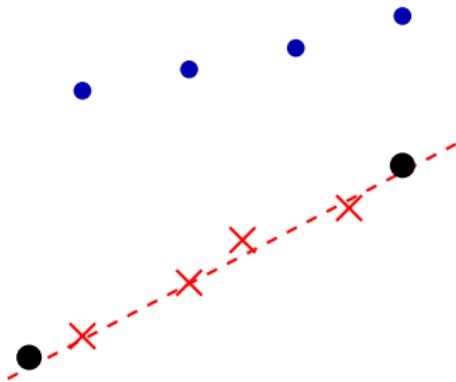
- ▶ Points d'entrée dans le nuage
- ▶ Sélection de deux points pour estimer une droite
- ▶ Inliers détectés à proximité de la droite

Reconstruction 3D  
└ Appariement  
  └ Pré-traitement  
    └ Ransac

2025-05-19



## Ransac

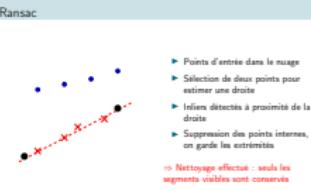


- ▶ Points d'entrée dans le nuage
  - ▶ Sélection de deux points pour estimer une droite
  - ▶ Inliers détectés à proximité de la droite
  - ▶ Suppression des points internes, on garde les extrémités
- ⇒ Nettoyage effectué : seuls les segments visibles sont conservés

2025-05-19

# Reconstruction 3D

- └ Appariement
- └ Pré-traitement
- └ Ransac



## Résultats pré-traitement

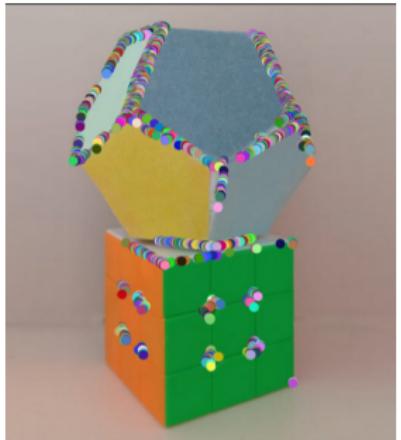


Moravec

2025-05-19



## Résultats pré-traitement

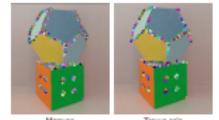


Moravec



Trouve coin

2025-05-19



## Résultats pré-traitement



Moravec



Trouve coin



Ransac

## Reconstruction 3D

Appariement

Pré-traitement

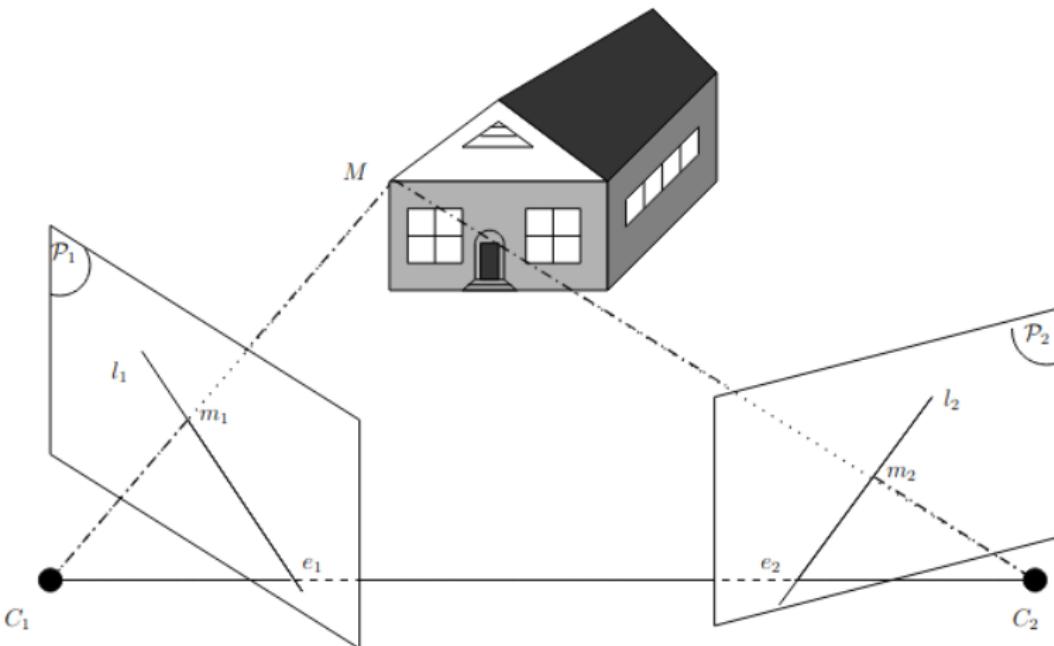
Résultats pré-traitement

2025-05-19

Résultats pré-traitement



# Geometrie epipolaire



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

## Reconstruction 3D

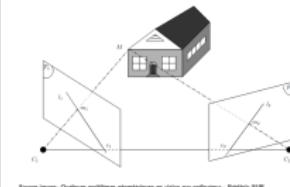
### Appariement

#### Filtrage epipolaire

##### Geometrie epipolaire

2025-05-19

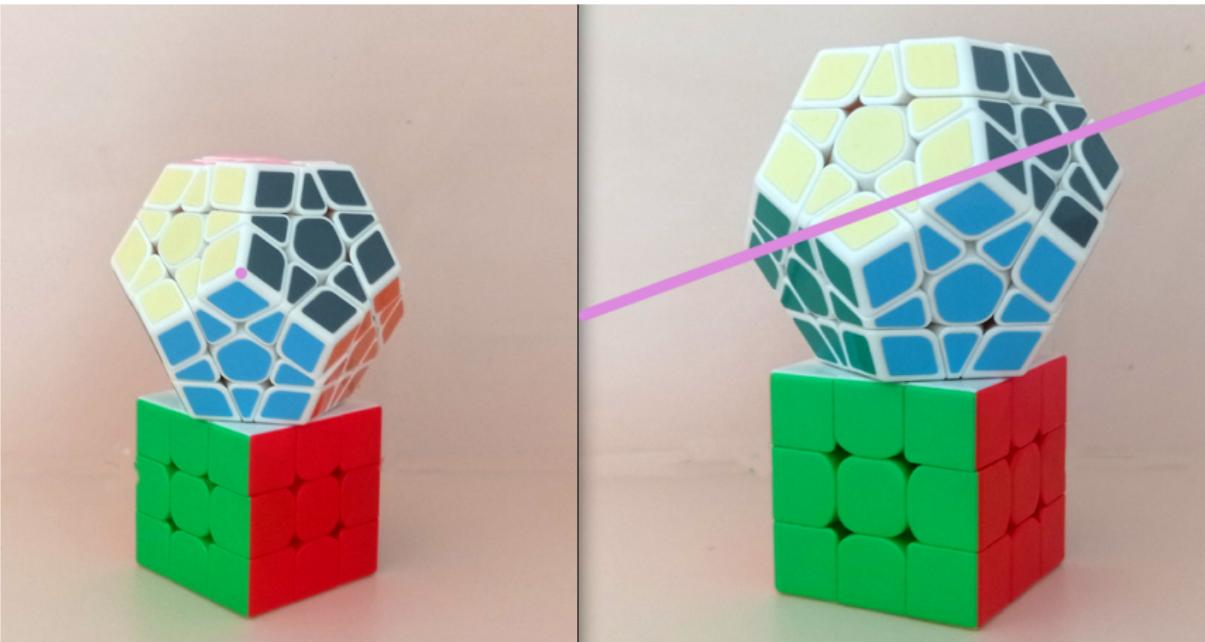
Geometrie epipolaire



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

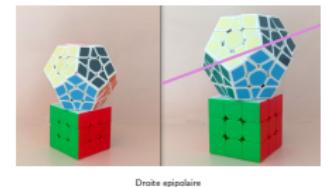
J'introduis ici la géométrie épipolaire : lorsque deux caméras observent une même scène, les points correspondants sont contraints de vérifier une relation géométrique. Les deux caméras ont pour centres optiques respectifs  $C_1$  et  $C_2$  et pour plans-images  $P_1$  et  $P_2$ . Pour des raisons de lisibilité, les plans-images sont placés dans ce schéma devant les centres optiques. Un point  $M$  de la scène se projette sur le plan-image de la caméra 1 (resp. 2) en  $m_1$  (resp.  $m_2$ ). Le point  $e_1$  (resp.  $e_2$ ) est l'image de  $C_2$  (resp.  $C_1$ ) sur  $P_1$  (resp.  $P_2$ ). Les points  $e_1$  et  $e_2$  sont appelés épipoles. La droite  $l_1$  (resp.  $l_2$ ) joignant les points  $e_1$  et  $m_1$  (resp.  $e_2$  et  $m_2$ ) est appelée droite épipolaire associée à  $m_2$  (resp.  $m_1$ ). C'est cette contrainte qui nous permettra de filtrer les correspondances incorrectes. Elle est encodée sous forme d'une matrice qu'on détermine à l'étape de calibration expliquée ultérieurement

# Filtrage epipolaire



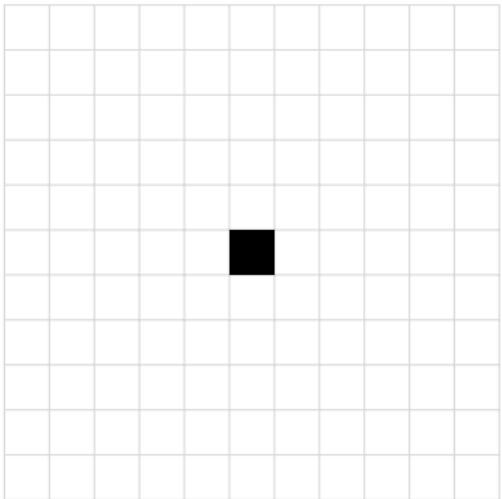
Droite epipolaire

2025-05-19



Droite epipolaire

## Descripteur BRIEF : Principe



**Comparaisons binaires :**

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

## Reconstruction 3D

Appariement

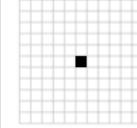
Brief

Descripteur BRIEF : Principe

2025-05-19

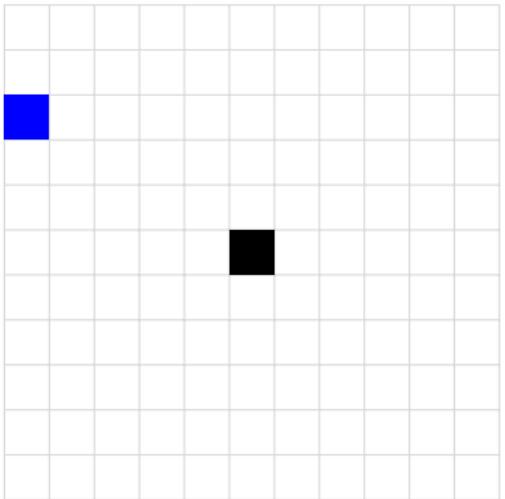
Descripteur BRIEF : Principe

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).

## Reconstruction 3D

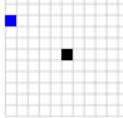
### Appariement Brief

#### Descripteur BRIEF : Principe

2025-05-19

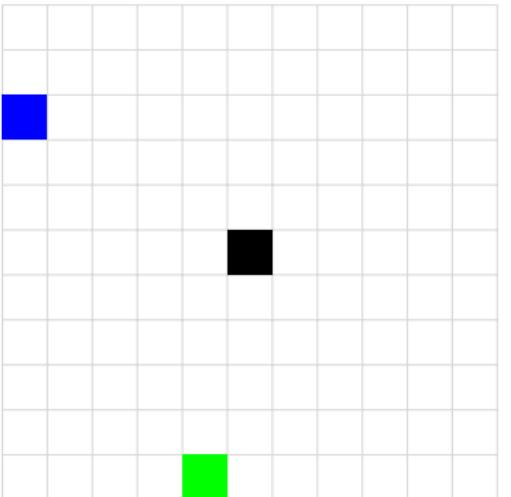
Descripteur BRIEF : Principe

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) :
- $\text{BRIEF}_1 = 1 \text{ si intensité(bleu)} < \text{intensité(vert)}$

## Reconstruction 3D

### Appariement

#### Brief

##### Descripteur BRIEF : Principe

2025-05-19

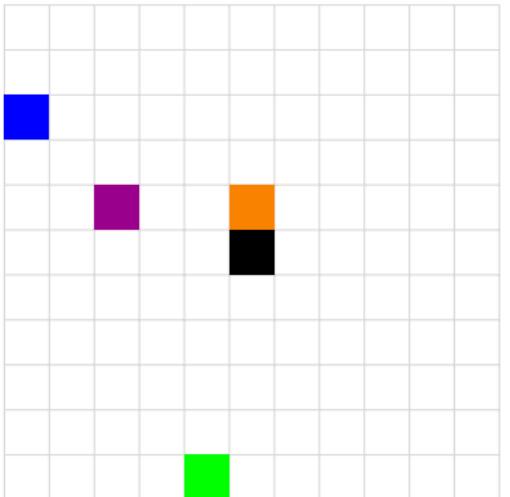
### Descripteur BRIEF : Principe

#### 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) :
- $\text{BRIEF}_1 = 1 \text{ si intensité(bleu)} < \text{intensité(vert)}$

## 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)...

## Reconstruction 3D

### Appariement

#### Brief

##### Descripteur BRIEF : Principe

2025-05-19

### Descripteur BRIEF : Principe

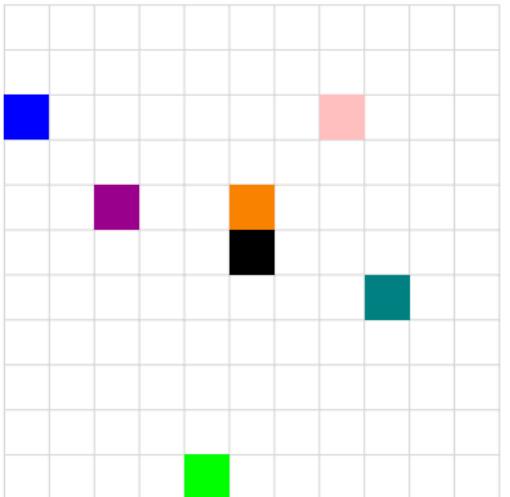
#### Comparaisons binaires :

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) :
 
$$\text{BRIEF}_1 = 1 \text{ si intensité(bleu)} < \text{intensité(vert)}$$
- ▶ On répète avec d'autres paires (ex: orange vs violet)...

## Reconstruction 3D



2025-05-19

### Descripteur BRIEF : Principe

#### Comparaisons binaires :

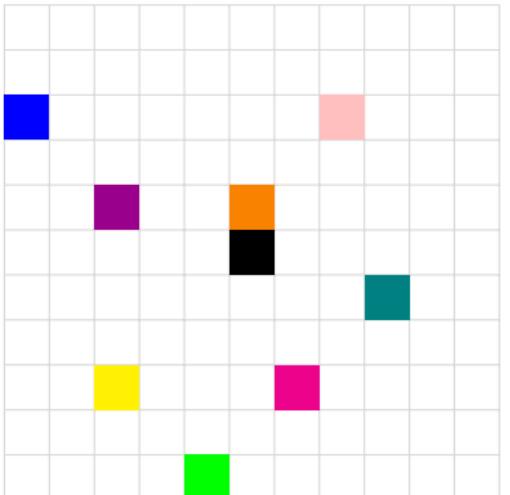
Bleu < Vert  $\Rightarrow 1$

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) :
 
$$\text{BRIEF}_1 = 1 \text{ si intensité(bleu)} < \text{intensité(vert)}$$
- ▶ On répète avec d'autres paires (ex: orange vs violet)...

## Descripteur BRIEF : Principe

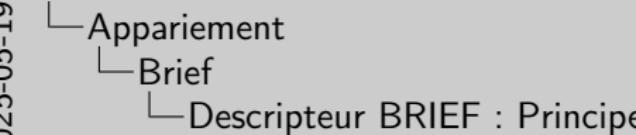


### Comparaisons binaires :

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

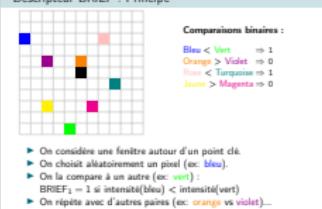
- ▶ On considère une fenêtre autour d'un point clé.
- ▶ On choisit aléatoirement un pixel (ex: bleu).
- ▶ On la compare à un autre (ex: vert) :  
 $BRIEF_1 = 1$  si intensité(bleu) < intensité(vert)
- ▶ On répète avec d'autres paires (ex: orange vs violet)...

## Reconstruction 3D

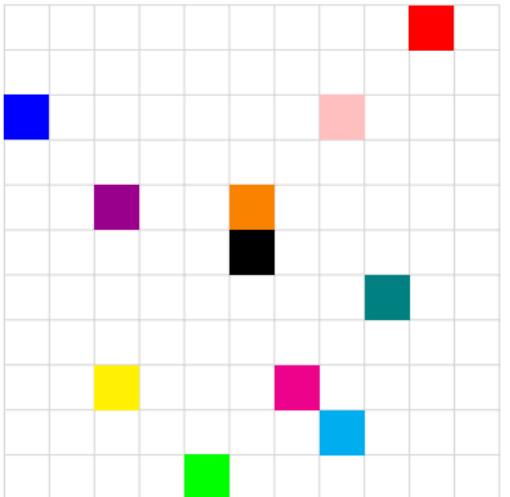


2025-05-19

### Descripteur BRIEF : Principe



## 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)...

## Reconstruction 3D

Appariement  
Brief

Descripteur BRIEF : Principe

2025-05-19

### Descripteur BRIEF : Principe

#### Comparaisons binaires :

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)...

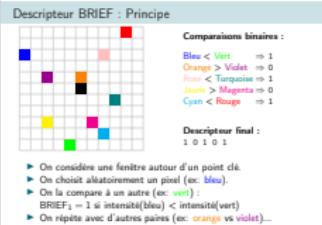
## Reconstruction 3D

### Appariement

#### Brief

##### Descripteur BRIEF : Principe

2025-05-19



## Comparaison de descripteurs BRIEF : distance de Hamming

**Descripteur 1 (image gauche)**

1 0 1 0 1

## Reconstruction 3D

## Appariement

## Brief

## Comparaison de descripteurs BRIEF : distance de Hamming

2025-05-19

## 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

## Reconstruction 3D

Appariement

Brief

Comparaison de descripteurs BRIEF : distance de Hamming

2025-05-19

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 :**

1	0	1	0	1
-	-	-	-	-
1	1	0	0	1
0	1	1	0	0

## Reconstruction 3D

Appariement

Brief

Comparaison de descripteurs BRIEF : distance de Hamming

2025-05-19

Descripteur 1 (image gauche)

1 0 1 0 1

Descripteur 2 (image droite)

1 1 0 0 1

Comparaison bit à bit :

1	0	1	0	1
-	-	-	-	-
1	1	0	0	1
0	1	1	0	0

## 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 :**

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

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

## Reconstruction 3D

Appariement

Brief

Comparaison de descripteurs BRIEF : distance de Hamming

2025-05-19

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 :

1 0 1 0 1

- - - - -

1 1 0 0 1

0 1 1 0 0

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 :**

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

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

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

Reconstruction 3D

Appariement

Brief

Comparaison de descripteurs BRIEF : distance de Hamming

2025-05-19

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 :	<table border="0"> <tr> <td>1 0 1 0 1</td> <td>- - - - -</td> </tr> <tr> <td>- - - - -</td> <td>1 1 0 0 1</td> </tr> <tr> <td>1 1 0 0 1</td> <td>0 1 1 0 0</td> </tr> </table>	1 0 1 0 1	- - - - -	- - - - -	1 1 0 0 1	1 1 0 0 1	0 1 1 0 0
1 0 1 0 1	- - - - -						
- - - - -	1 1 0 0 1						
1 1 0 0 1	0 1 1 0 0						
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*

## Reconstruction 3D

### Appariement

#### Brief

##### Amélioration progressive du descripteur BRIEF

## 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

2025-05-19

# 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*

2025-05-19

## 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. Amélioration progressive

# 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*

# Reconstruction 3D

## Appariement

### Brief

#### Amélioration progressive du descripteur BRIEF

2025-05-19

#### 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)*

## Reconstruction 3D

### Appariement

#### Brief

##### Amélioration progressive du descripteur BRIEF

2025-05-19

### 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

## Reconstruction 3D

### Appariement

#### Algorithme

##### Pseudo-code : Appariement de points

2025-05-19

```

Algorithme: Appariement
Entrée: Points P1 sur image 1, Points P2 sur image 2, Matrice fondamentale F
Sortie: Liste de correspondances fiables
Pré-tri des points sur image 1
pour tout p ∈ P1 faire
    si p n'est pas un coin alors
        retirer p // suppression non maximale locale
    
```

// suppression non maximale locale

## Pseudo-code : Appariement de points

## Algorithmes : 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

$$l \leftarrow F \cdot p_1$$

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

6. *U. S. Fish Commission, Report for the Year 1877*, Part I, p. 10.

```

Algorithmic: Appairment
    Entrée: Points  $P_1$  sur image 1, Points  $P_2$  sur image 2, Matrice fondamentale  $F$ 
    Sortie: Liste des correspondances valides

    Pour tous les points sur image 1
        Pour tous les points sur image 2
            Pour tout  $p_2 \in P_2$ 
                Si  $p_1$  et  $p_2$  sont voisins
                    Calculer  $C(p_1, p_2)$  // distance entre  $p_1$  et  $p_2$ 
                    Si  $C(p_1, p_2) < \epsilon$  // suppression maximalement locale
                        Ajouter  $(p_1, p_2)$  à la liste des correspondances valides

```

# 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**

$I \leftarrow F \cdot p_1$

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

            // droite épipolaire dans image 2

### 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  $(p_1, p_2, h)$

# Reconstruction 3D

## Appariement

### Algorithme

#### Pseudo-code : Appariement de points

2025-05-19

#### Pseudo-code : Appariement de points

##### Algorithmic: 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  $p_1 \in P_1$  faire

**si**  $p_1$  est un coin alors

        retirer  $p_1$

            // suppression non maximale locale

Filtrage épipolaire

pour tout  $p_1 \in P_1$  faire

$I \leftarrow F \cdot p_1$

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

            // droite épipolaire dans image 2

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

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

$h = \text{BRIEF}(p_2)$

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

        enregistrer  $(p_1, p_2, h)$

# 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

        retirer  $p$

            // suppression non maximale locale

### Filtrage épipolaire

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

$I \leftarrow F \cdot p_1$

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

            // droite épipolaire dans image 2

### 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  $(p_1, p_2, h)$

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

# Reconstruction 3D

## Appariement

### Algorithme

#### Pseudo-code : Appariement de points

2025-05-19

## Pseudo-code : Appariement de points

### Algorithmique: 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  $p_1 \in P_1$  faire

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

    pour tout  $p_2 \in P_2$  faire

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

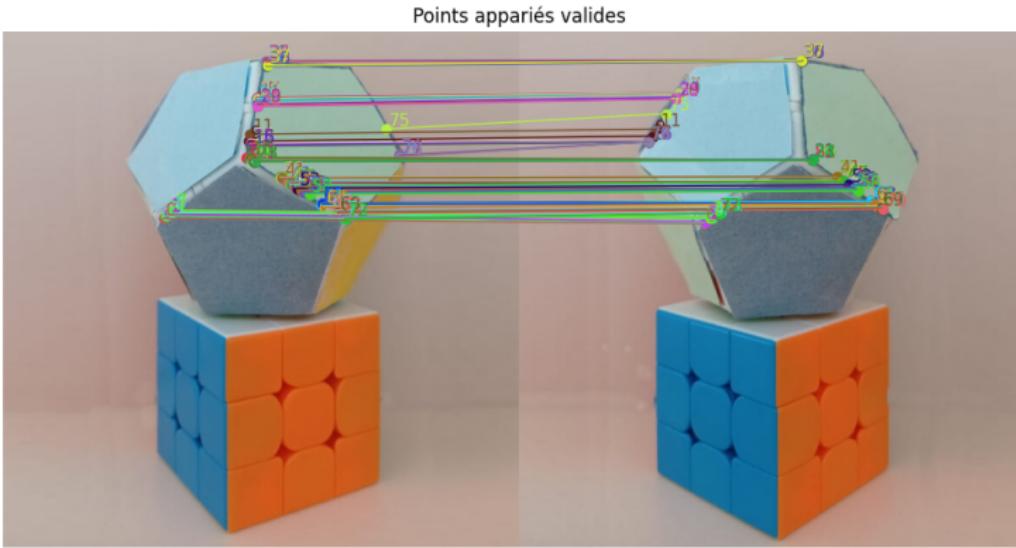
$h = \text{distance\_Hamming}(d_1, d_2)$

        enregistrer  $(p_1, p_2, h)$

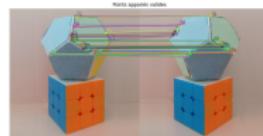
    retirer  $p$

        // suppression non maximale locale

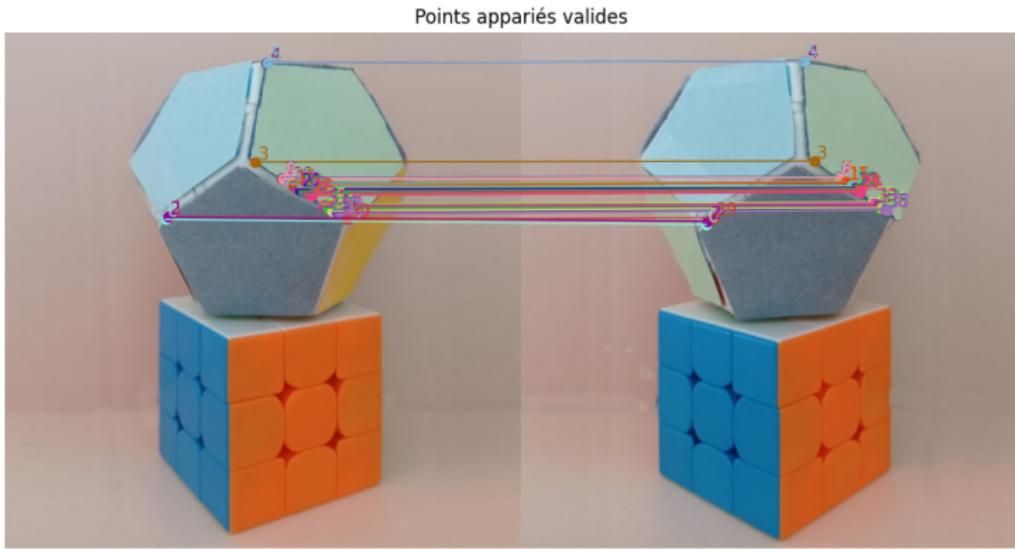
## Resultat : Filtrage epipolaire



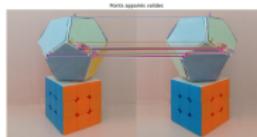
2025-05-19



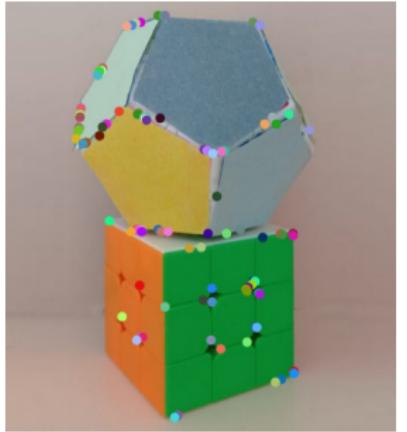
## Resultat : droite epipolaire + BRIEF lab



2025-05-19



## Résultats pré-traitement



Points gardés

▶ constantes

## Reconstruction 3D

Appariement

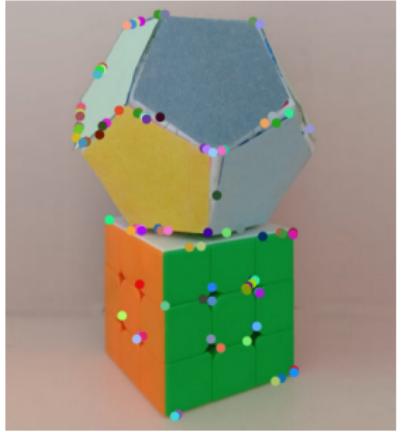
Resultat

Résultats pré-traitement

2025-05-19

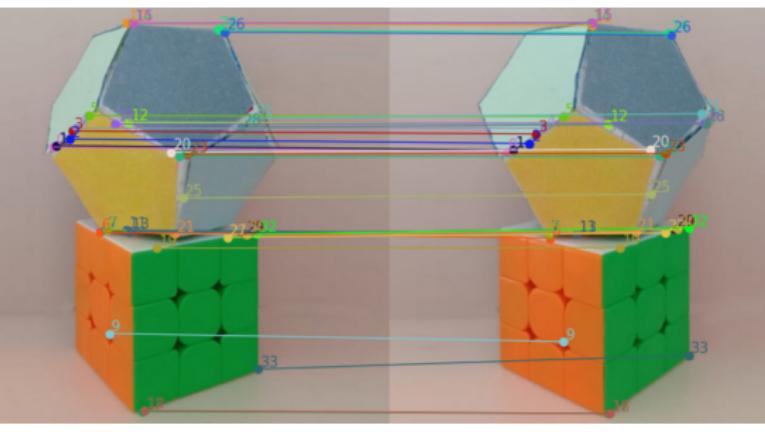


## Résultats pré-traitement

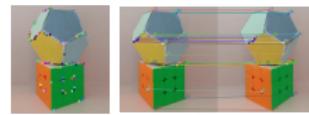


## Points gardés

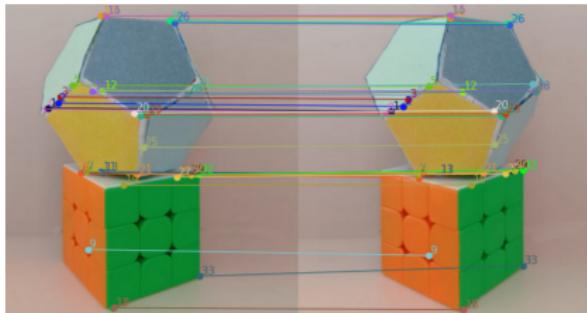
## ► constantes



## Appariement

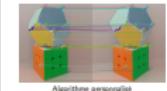


## comparaison avec SIFT opencv



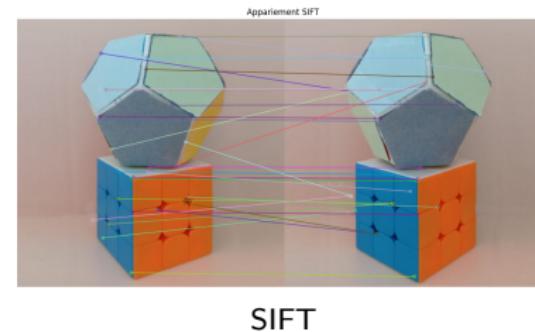
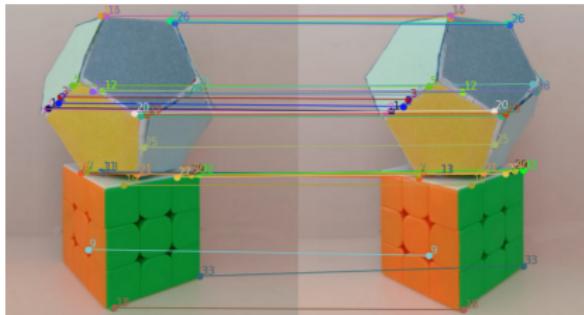
Algorithme personnalisé

2025-05-19

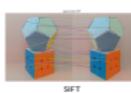
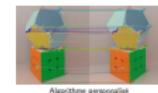


sift est vraiment nulle comparé à notre algo

## comparaison avec SIFT opencv



2025-05-19



sift est vraiment nulle comparé à notre algo

# Plan

1. Selection

2. Appariement

3. Calibration

Système de calibrage

En pratiques

Resolution système homogène

Implémentation

4. Reconstruction des points

Reconstruction 3D

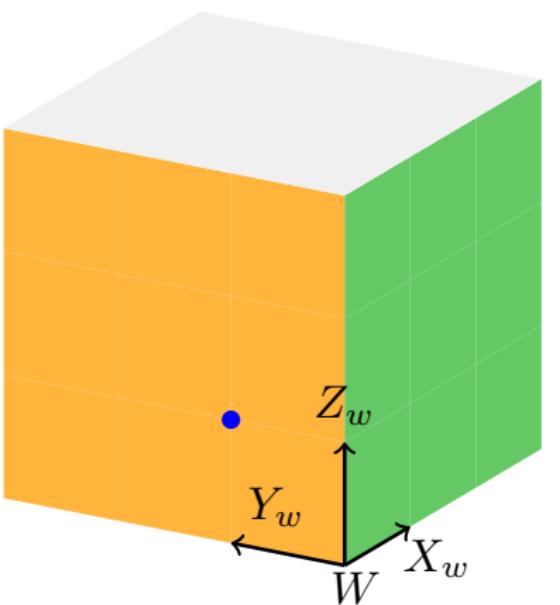
└ Calibration

└ Plan[0.1cm]

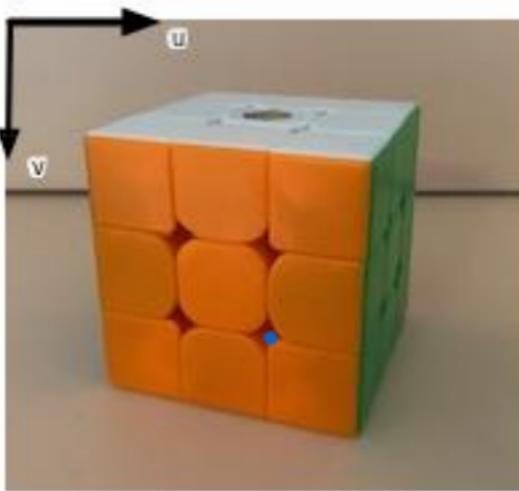
2025-05-19

Plan  
1. Selection  
2. Appariement  
3. Calibration  
Système de calibrage  
En pratiques  
Resolution système homogène  
Implémentation  
4. Reconstruction des points

## Les différents repères

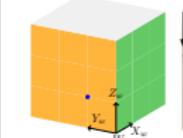


Représentation du cube (vue 3D)



Cube sur une image

2025-05-19



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

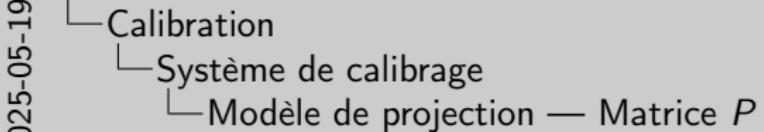
2025-05-19

## 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 ?):

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

## Reconstruction 3D



Modèle de projection — Matrice  $P$

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

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

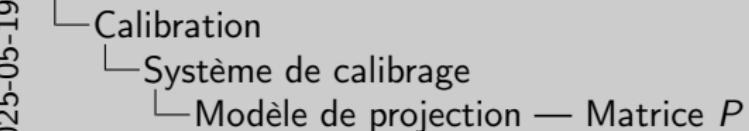
## Modèle de projection — Matrice $P$

- ▶ 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

## Reconstruction 3D



2025-05-19

Modèle de projection — Matrice  $P$

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

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

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

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

## Reconstruction 3D

## Calibration

## Système de calibrage

Modèle de projection — Matrice  $P$ 

2025-05-19

Modèle de projection — Matrice  $P$ 

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

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

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

2025-05-19

► Pour un point donné, on élimine  $\lambda$  :

$$u(p_{21}X + p_{22}Y + p_{23}Z + p_{24}) = p_{11}X + p_{12}Y + p_{13}Z + p_{14}$$

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

## Équations sans $\lambda$ et système matriciel

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

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

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

- ▶ Cela donne un système homogène ...

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

2025-05-19

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

$$\begin{aligned} u(p_{21}X + p_{22}Y + p_{23}Z + p_{24}) &= p_{11}X + p_{12}Y + p_{13}Z + p_{14} \\ v(p_{21}X + p_{22}Y + p_{23}Z + p_{24}) &= p_{31}X + p_{32}Y + p_{33}Z + p_{34} \end{aligned}$$

- ▶ 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 ...

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

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

2025-05-19

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

$$\begin{aligned} u(p_{21}X + p_{22}Y + p_{23}Z + p_{24}) &= p_{11}X + p_{12}Y + p_{13}Z + p_{14} \\ v(p_{21}X + p_{22}Y + p_{23}Z + p_{24}) &= p_{31}X + p_{32}Y + p_{33}Z + p_{34} \end{aligned}$$

- ▶ Cela donne un système homogène ...

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

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

# Équations sans $\lambda$ et système matriciel

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

2025-05-19

$$\left( \begin{array}{cccc|ccccc} a_1^{(1)} & a_2^{(1)} & a_3^{(1)} & 1 & 0 & 0 & 0 & -u^{(1)}a_1^{(1)} & -u^{(1)}a_2^{(1)} & -u^{(1)}a_3^{(1)} \\ a_1^{(2)} & a_2^{(2)} & a_3^{(2)} & 1 & 0 & 0 & 0 & -u^{(2)}a_1^{(2)} & -u^{(2)}a_2^{(2)} & -u^{(2)}a_3^{(2)} \\ a_1^{(3)} & a_2^{(3)} & a_3^{(3)} & 1 & 0 & 0 & 0 & -u^{(3)}a_1^{(3)} & -u^{(3)}a_2^{(3)} & -u^{(3)}a_3^{(3)} \\ a_1^{(4)} & a_2^{(4)} & a_3^{(4)} & 1 & 0 & 0 & 0 & -u^{(4)}a_1^{(4)} & -u^{(4)}a_2^{(4)} & -u^{(4)}a_3^{(4)} \\ a_1^{(5)} & a_2^{(5)} & a_3^{(5)} & 1 & 0 & 0 & 0 & -u^{(5)}a_1^{(5)} & -u^{(5)}a_2^{(5)} & -u^{(5)}a_3^{(5)} \\ a_1^{(6)} & a_2^{(6)} & a_3^{(6)} & 1 & 0 & 0 & 0 & -u^{(6)}a_1^{(6)} & -u^{(6)}a_2^{(6)} & -u^{(6)}a_3^{(6)} \\ \hline b_1 & b_2 & b_3 & 1 & 0 & 0 & 0 & b_1 & b_2 & b_3 \end{array} \right)$$

# Calibration



Figure: Cube calibrage



Figure: Selection des points

Reconstruction 3D  
└ Calibration  
  └ En pratiques  
    └ Calibration

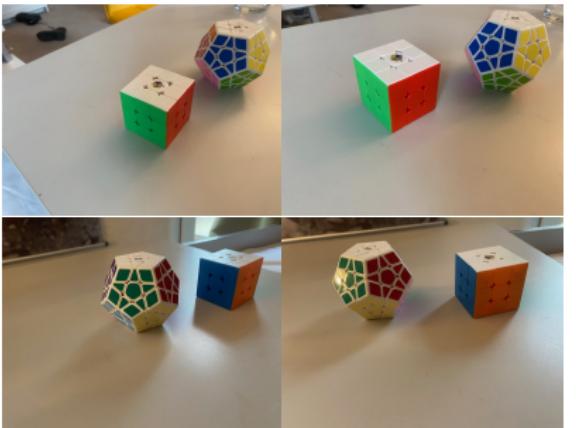
2025-05-19

Calibration

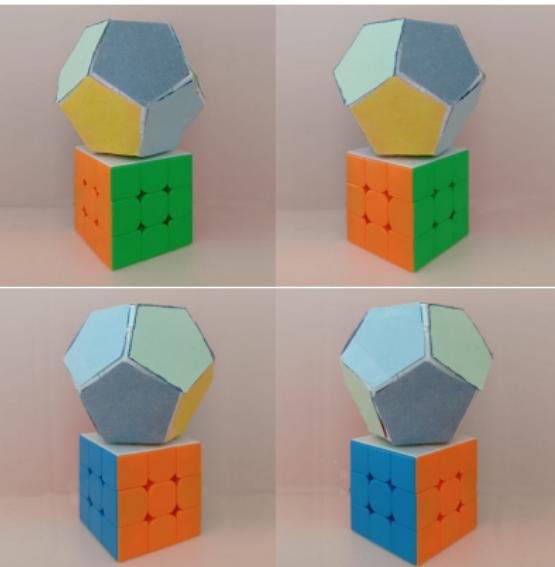


1. rubiks cube comme mire de calibrage, permet d'identifier facilement les faces et d'y apposer un repère On créer des fichiers types avec les coordonnées des points pour chaque paire de faces et ils nous suffit de sélectionner les points à la main pour calibrer

## Shooting photo : importance de la prise de vue



Vues initiales



Vues améliorées

## Reconstruction 3D

## Calibration

## Resolution système homogène

## Shooting photo : importance de la prise de vue

2025-05-19



Le système issu de la calibration est homogène, donc la matrice nulle est toujours solution, mais cette solution n'a aucun sens géométrique. Pour éviter cette solution triviale, on cherche une solution non nulle, à un facteur près, qu'on trouve en utilisant la SVD.

## Équations sans $\lambda$ et système matriciel

$$\left( \begin{array}{cccccc} v & x_1 & x_2 & \dots & x_n & x_{n+1} \\ w & y_1 & y_2 & \dots & y_n & y_{n+1} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ u & z_1 & z_2 & \dots & z_n & z_{n+1} \end{array} \right)$$

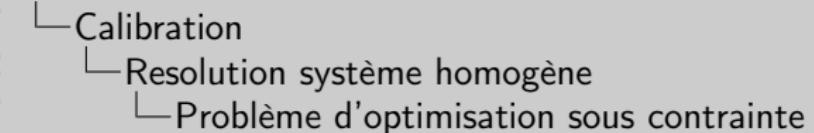
## Problème d'optimisation sous contrainte

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

On reformule le problème comme une minimisation sous contrainte :

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

## Reconstruction 3D



2025-05-19

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

On reformule le problème comme une minimisation sous contrainte :

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

Résolution classique -> solution triviale nulle Le système est homogène -> on peut fixé la norme du vecteur  $p$  à 1. On transforme le problème en une minimisation sous contrainte.

En appliquant les multiplicateurs de Lagrange, on obtient une condition d'optimalité : le gradient de la fonction est proportionnel au gradient de la contrainte, ce qui donne une équation aux valeurs propres.

# 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

2025-05-19

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 :

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

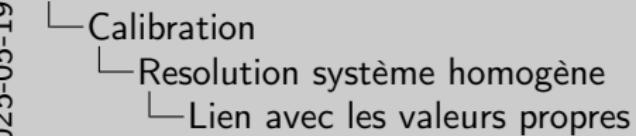
Résolution classique -> solution triviale nulle Le système est homogène -> on peut fixer la norme du vecteur  $p$  à 1. On transforme le problème en une minimisation sous contrainte.

En appliquant les multiplicateurs de Lagrange, on obtient une condition d'optimalité : le gradient de la fonction est proportionnel au gradient de la contrainte, ce qui donne une équation aux valeurs propres.

## Lien avec les valeurs propres

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

## Reconstruction 3D



Lien avec les valeurs propres

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

La matrice  $A^T A$  est symétrique, ce qui garantit qu'elle est diagonalisable et possède une base orthonormée de vecteurs propres.

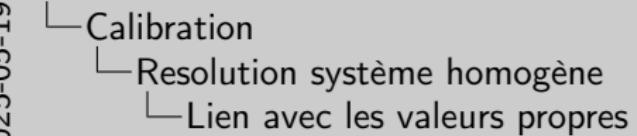
La solution optimale  $p$  est donc un vecteur propre. Pour minimiser  $\|Ap\|$ , il faut prendre celui associé à la plus petite valeur propre.

C'est exactement la direction dans laquelle l'application de  $A$  est la moins « amplifiée ».

## 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.

## Reconstruction 3D



2025-05-19

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.

La matrice  $A^T A$  est symétrique, ce qui garantit qu'elle est diagonalisable et possède une base orthonormée de vecteurs propres.

La solution optimale  $p$  est donc un vecteur propre. Pour minimiser  $\|Ap\|$ , il faut prendre celui associé à la plus petite valeur propre.

C'est exactement la direction dans laquelle l'application de  $A$  est la moins « amplifiée ».

## 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.

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

2025-05-19

- ▶  $A^T A$  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.

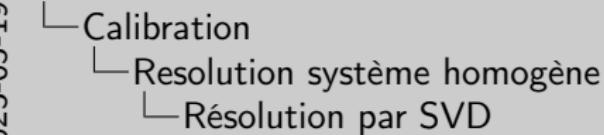
$$\min \|Ap\| \Rightarrow p = \text{vecteur propre de plus petite valeur propre de } A^T A$$

## Résolution par SVD

► On factorise  $A$  :

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

## Reconstruction 3D



Résolution par SVD

► On factorise  $A$  :  
$$A = U\Sigma V^T$$

La décomposition en valeurs singulières, ou SVD, permet de factoriser n'importe quelle matrice  $A$  en trois matrices : deux orthogonales  $U$  et  $V$ , et une diagonale  $\Sigma$  contenant les valeurs singulières.

Ces valeurs sont les racines carrées des valeurs propres de  $A^T A$ .

La dernière colonne de  $V$ , associée à la plus petite valeur singulière, donne donc directement le vecteur  $p$  recherché.

C'est une méthode très stable numériquement.

# Résolution par SVD

- On factorise  $A$  :

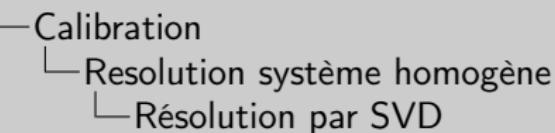
$$A = U\Sigma V^T$$

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

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

## Reconstruction 3D

2025-05-19



## 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  $\sigma_n$ .

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

La décomposition en valeurs singulières, ou SVD, permet de factoriser n'importe quelle matrice  $A$  en trois matrices : deux orthogonales  $U$  et  $V$ , et une diagonale  $\Sigma$  contenant les valeurs singulières.

Ces valeurs sont les racines carrées des valeurs propres de  $A^T A$ .

La dernière colonne de  $V$ , associée à la plus petite valeur singulière, donne donc directement le vecteur  $p$  recherché.

C'est une méthode très stable numériquement.

## Décomposition SVD : idée générale

Pseudo code

## Étapes principales :

- 1. Calculer  $A^T A$

## 1. Produit symétrique :

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

On peut chercher ses valeurs propres

via QR.

## Reconstruction 3D

## └ Calibration

## └ Implémentation

## └ Décomposition SVD : idée générale

2025-05-19

## 1. Produit symétrique :

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

## Etapes principales :

- 1. Calculer  $A^T A$   
via QR.

On peut chercher ses valeurs propres

## Décomposition SVD : idée générale

Pseudo code

## Étapes principales :

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

## 2. Algorithme QR :

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

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

## Reconstruction 3D

## Calibration

## Implémentation

## Décomposition SVD : idée générale

2025-05-19

## Décomposition SVD : idée générale

Pseudo code

## Étapes principales :

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

## 3. Valeurs singulières :

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

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

## Reconstruction 3D

## Calibration

## Implémentation

## Décomposition SVD : idée générale

2025-05-19

## 3. Valeurs singulières :

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

## Etapes 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

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  
 $U = \frac{1}{\sigma} Av$

## 4. Vecteurs singuliers :

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

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

## Reconstruction 3D

## Calibration

## Implémentation

## Décomposition SVD : idée générale

2025-05-19

## 4. Vecteurs singuliers :

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

## Etapes principales :

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

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

# Plan

1. Selection
2. Appariement
3. Calibration
4. Reconstruction des points

Reconstruction 3D  
└ Reconstruction des points  
    └ Plan[0.1cm]

2025-05-19

Plan  
1. Selection  
2. Appariement  
3. Calibration  
4. Reconstruction des points

## 4- Reconstruction des points

## Reconstruction : retrouver les coordonnées 3D

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

## Reconstruction 3D

## └ Reconstruction des points

## └ Reconstruction : retrouver les coordonnées 3D

Reconstruction : retrouver les coordonnées 3D

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

2025-05-19

## 4- Reconstruction des points

## Reconstruction : retrouver les coordonnées 3D

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

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

## Reconstruction 3D

## └ Reconstruction des points

## └ Reconstruction : retrouver les coordonnées 3D

2025-05-19

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  $X, Y, Z$

## Reconstruction 3D

## └ Reconstruction des points

## └ Reconstruction : retrouver les coordonnées 3D

2025-05-19

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 :

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

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

## Reconstruction 3D

## └ Reconstruction des points

## └ Reconstruction : retrouver les coordonnées 3D

2025-05-19

- 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 :

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

## 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

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

## Reconstruction 3D

## └ Reconstruction des points

## └ Reconstruction : retrouver les coordonnées 3D

2025-05-19

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

## 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

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

## Reconstruction 3D

## └ Reconstruction des points

## └ Reconstruction : retrouver les coordonnées 3D

2025-05-19

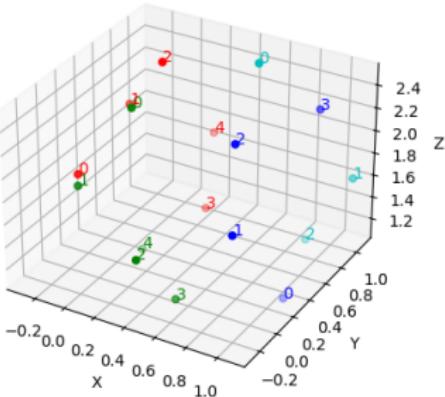
- On connaît les matrices de projection  $P_1$  et  $P_2$
- Deux points image correspondants :
- $x_1 = (u_1, v_1), \quad x_2 = (u_2, v_2)$
- Inconnue : coordonnées 3D  $X, Y, Z$
- On a :
- $\lambda_1 x_1 = P_1 X, \quad \lambda_2 x_2 = P_2 X$
- Système de 6 équations linéaires homogènes
- Résolution par SVD

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

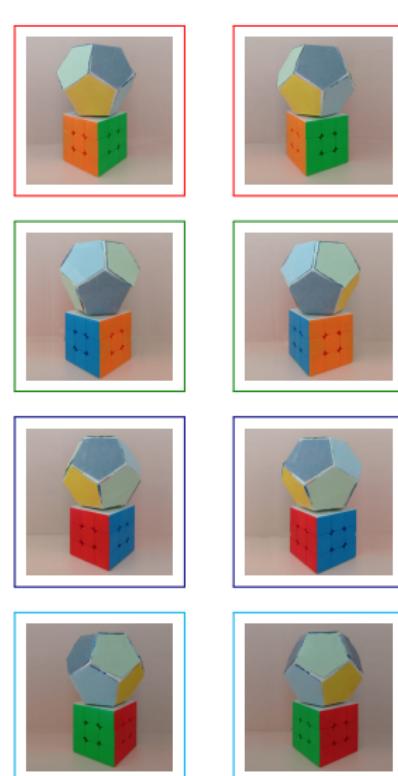
## 4- Reconstruction des points

## Reconstruction 3D multi-vues

Points 3D



Nuage de points

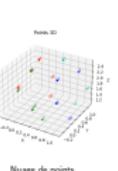


## Reconstruction 3D

## └ Reconstruction des points

## └ Reconstruction 3D multi-vues

2025-05-19



5- [

Appendix]Appendix

## Reconstruction 3D

└ Reconstruction des points

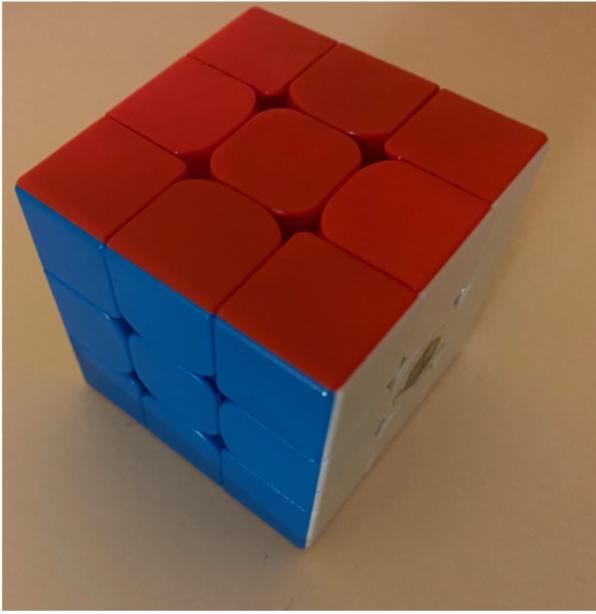
└ Reconstruction 3D multi-vues

2025-05-19



5- [

## Exemple Moravec



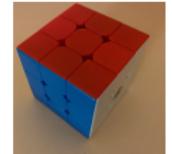
Reconstruction 3D

[

Exemple Moravec

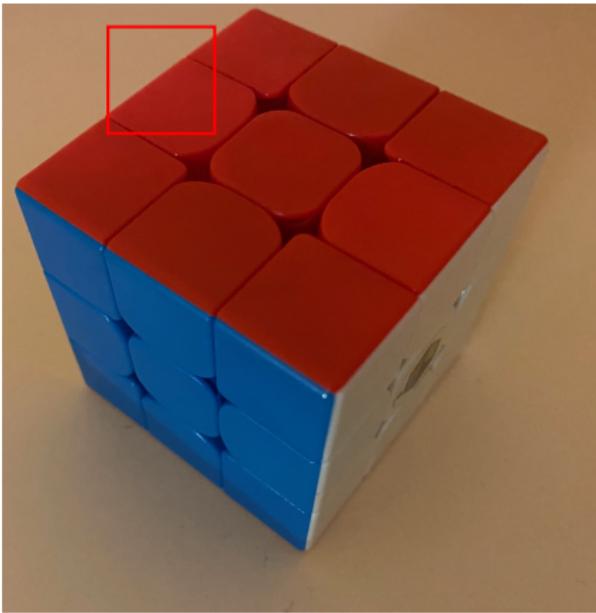
2025-05-19

Exemple Moravec



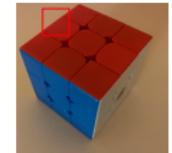
5- [

## Exemple Moravec



2025-05-19

└ Exemple Moravec



5- [

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

 $w = 2$ 

└ [

2025-05-19

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

 $w = 2$ 

Voici une animation de l'évolution de la détection avec l'algorithme de Moravec.

On visualise étape par étape comment les coins apparaissent au fur et à mesure.

Ce genre de visualisation aide à comprendre la sensibilité de l'algorithme selon l'image et les paramètres.

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

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

## Reconstruction 3D

[

2025-05-19

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

Voici une animation de l'évolution de la détection avec l'algorithme de Moravec.

On visualise étape par étape comment les coins apparaissent au fur et à mesure.

Ce genre de visualisation aide à comprendre la sensibilité de l'algorithme selon l'image et les paramètres.

5- [

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

 $i = -1$  $S = 52 \quad S^2 = 1248$ 

## Reconstruction 3D



2025-05-19

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  
 $i = -1$   
 $S = 52 \quad S^2 = 1248$

Voici une animation de l'évolution de la détection avec l'algorithme de Moravec.

On visualise étape par étape comment les coins apparaissent au fur et à mesure.

Ce genre de visualisation aide à comprendre la sensibilité de l'algorithme selon l'image et les paramètres.

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

 $i = 2$  $S = 205 \quad S^2 = 10339$  $Var(1, 0) = 386.8$ 

## Reconstruction 3D

[

2025-05-19

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  
 $dx \geq 1, dy = 0$   
 pixel considéré  
 pixel comparé  
 $i = 2$   
 $S = 205 \quad S^2 = 10339$   
 $Var(1, 0) = 386.8$

Voici une animation de l'évolution de la détection avec l'algorithme de Moravec.

On visualise étape par étape comment les coins apparaissent au fur et à mesure.

Ce genre de visualisation aide à comprendre la sensibilité de l'algorithme selon l'image et les paramètres.

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

 $w = 2$  $T = 300$  $\text{Var}(1,0)=386.8$  $\text{Var}(0,1)=526.8$  $\text{Var}(1,1)=439.7$  $\text{Var}(1,-1)=471.2$  $S = 386.8$  $S > T \implies \text{point d'intérêt}$ 

## Reconstruction 3D



2025-05-19

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  
Var(1,0):=386.8  
Var(0,1):=526.8  
Var(1,1):=439.7  
Var(1,-1):=471.2  
S = 386.8  
 $S > T \implies \text{point d'intérêt}$

Voici une animation de l'évolution de la détection avec l'algorithme de Moravec.

On visualise étape par étape comment les coins apparaissent au fur et à mesure.

Ce genre de visualisation aide à comprendre la sensibilité de l'algorithme selon l'image et les paramètres.

5- [

Voir le schéma détaillé



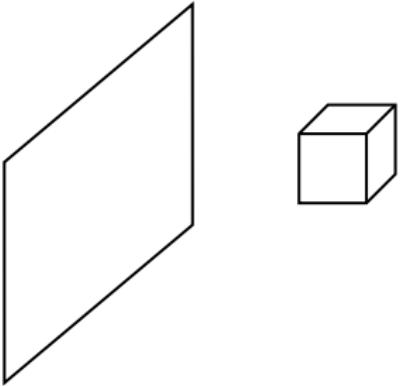
2025-05-19

.../...



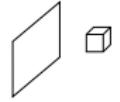
2025-05-19

## Les différents repères



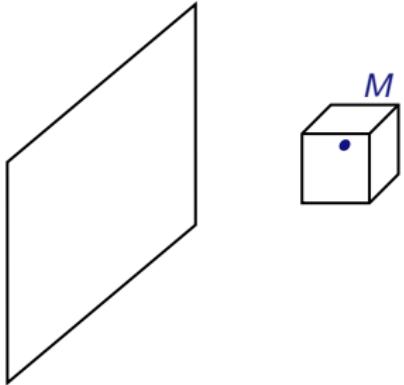
2025-05-19

└ Les différents repères



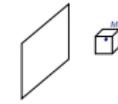
## Les différents repères

►  $M$  : point réel

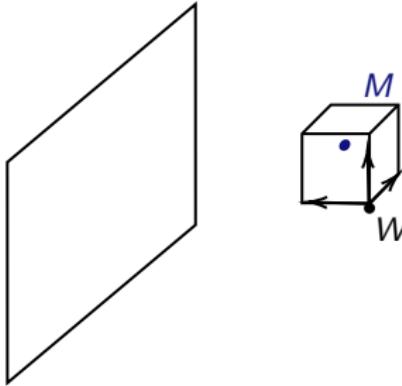


2025-05-19

## └ Les différents repères



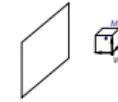
## Les différents repères



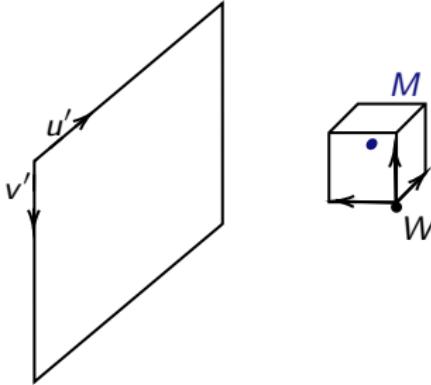
- ▶  $M$  : point réel
- ▶  $W$  : origine du repère du monde

## Les différents repères

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



## Les différents repères



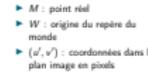
- ▶  $M$  : point réel
- ▶  $W$  : origine du repère du monde
- ▶  $(u', v')$  : coordonnées dans le plan image en pixels

2025-05-19

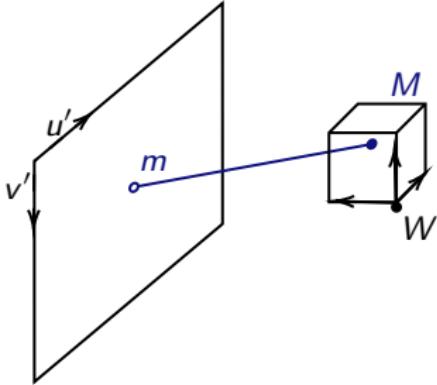
## Reconstruction 3D

### └ Les différents repères

Les différents repères



## Les différents repères



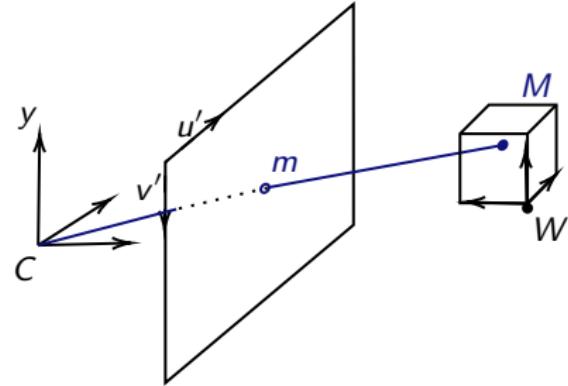
- ▶  $M$  : point réel
- ▶  $W$  : origine du repère du monde
- ▶  $(u', v')$  : coordonnées dans le plan image en pixels
- ▶  $m$  : projection de  $M$  dans le plan image

## 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

## Les différents repères

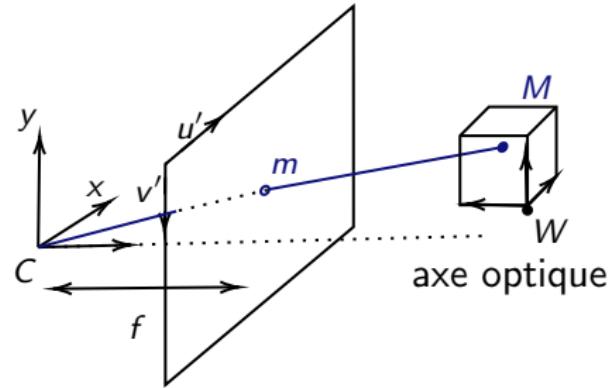


- ▶  $M$  : point réel
- ▶  $W$  : origine du repère du monde
- ▶  $(u', v')$  : coordonnées dans le plan image en pixels
- ▶  $m$  : projection de  $M$  dans le plan image
- ▶  $C$  : origine du repère de la caméra

## 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

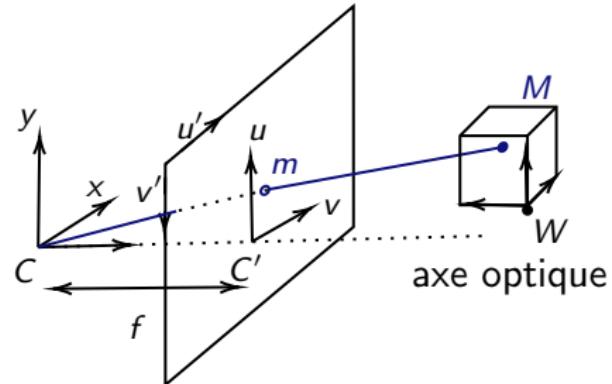


- ▶  $M$  : point réel
- ▶  $W$  : origine du repère du monde
- ▶  $(u', v')$  : coordonnées dans le plan image en pixels
- ▶  $m$  : projection de  $M$  dans le plan image
- ▶  $C$  : origine du repère de la caméra

## Les différents repères

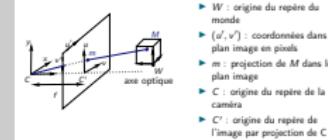
- ▶  $M$  : point réel
- ▶  $W$  : origine du repère du monde
- ▶  $(u', v')$  : coordonnées dans le plan image en pixels
- ▶  $m$  : projection de  $M$  dans le plan image
- ▶  $C$  : origine du repère de la caméra

## 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  $C$

## Les différents repères



5- [

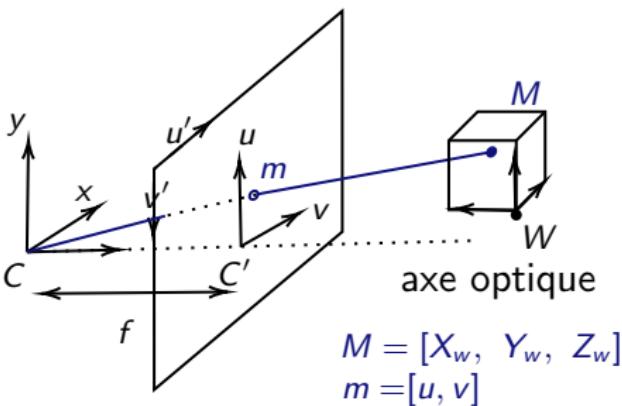
## Projection d'un point 3D sur le plan image

Par le théorème de Thalès  
(projection perspective)

$$u = fx_c$$

$$v = fy_c$$

$$w = z_c$$



2025-05-19

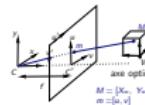
## Projection d'un point 3D sur le plan image

Par le théorème de Thalès  
(projection perspective)

$$u = \frac{fx_c}{w}$$

$$v = \frac{fy_c}{w}$$

$$w = z_c$$



5- [

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

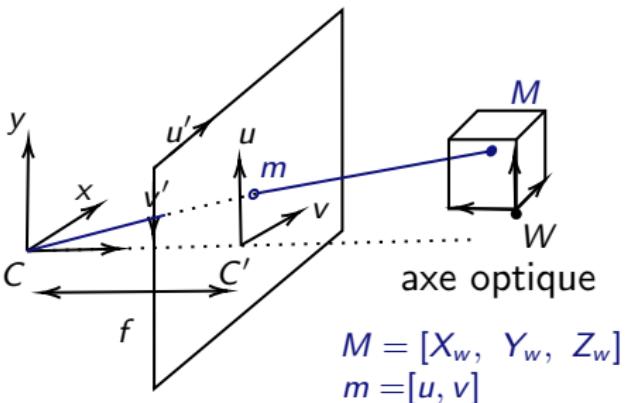
Par le théorème de Thalès  
(projection perspective)

$$u = fx_c$$

$$v = fy_c$$

$$w = z_c$$

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



2025-05-19

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

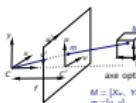
Par le théorème de Thalès  
(projection perspective)

$$u = fx_c$$

$$v = fy_c$$

$$w = z_c$$

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

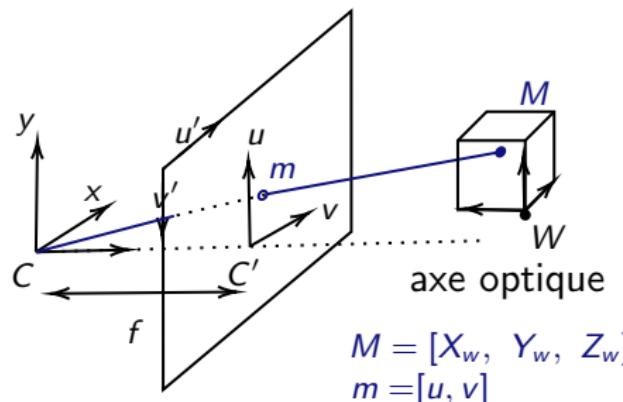


## Projection d'un point 3D sur le plan image

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.

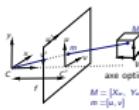


## Projection d'un point 3D sur le plan image

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.



5- [

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

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

2025-05-19

Projection d'un point 3D sur le plan image

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

5- [

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

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

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

2025-05-19

## 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

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

## └ Les différents repères

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

## Système d'optimisation à contrainte unitaire

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



└ Système d'optimisation à contrainte unitaire

2025-05-19

## 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

2025-05-19

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 :

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



2025-05-19

## 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 :

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

On introduit les fonctions :

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



2025-05-19

## 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 :

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

On introduit les fonctions :

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

## 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 :

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

On introduit les fonctions :

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

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

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



2025-05-19

## 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 :

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

On introduit les fonctions :

$$\triangleright f(p) = p^T A^T Ap$$

$$\triangleright g(p) = p^T p - 1$$

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

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

## Lien avec les valeurs propres

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

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

## Reconstruction 3D



└ Lien avec les valeurs propres

2025-05-19

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

## 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 :

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

## Reconstruction 3D



└ Lien avec les valeurs propres

2025-05-19

### 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 :

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

5- [

## 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 :

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

## Reconstruction 3D

[

### Lien avec les valeurs propres

2025-05-19

#### 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



2025-05-19

## Lien avec les valeurs propres

### 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

## Triangulation : formulation du système

- $P_1$  et  $P_2$  déterminées



Triangulation : formulation du système

2025-05-19

## Triangulation : formulation du système

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

## Reconstruction 3D

2025-05-19

### └ 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

## Reconstruction 3D

2025-05-19

### └ 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

## Reconstruction 3D

2025-05-19

### └ Triangulation : formulation du système

#### 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  $AX = 0$

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

[

## Triangulation : formulation du système

2025-05-19

- ▶  $P_1$  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  $AX = 0$

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

## 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**

$v_j \leftarrow A_{:,j}$

**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**  $Q, R$

## Reconstruction 3D

2025-05-19

```

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
   $v_j \leftarrow A_{:,j}$ 
  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  $Q, R$ 

```

5- [

## 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  $1[i, i] > \varepsilon$  alors
         $\Sigma^2[i, i] \leftarrow V[i, i]$ 
    sinon
         $\Sigma^2[i, i] \leftarrow 0$ 
retourner  $\Sigma^2, Q_{\text{acc}}$ 

```

## Reconstruction 3D

2025-05-19



**Algorithmique: algorithme QR**

---

**Entrée:**  $B \in \mathbb{R}^{n \times n}$  symétrique  
**Sortie:**  $\Sigma^2, V$  tels que  $B = V\Sigma^2V^T$

---

```

Qacc ← In
δ ← 1, kmax ← 1000, k ← 0
tant que δ > 10-9 et k < kmax faire
    | Q, R ← décomposition QR de B
    | Bnouveau ← R · Q
    | Qacc ← Qacc · Q
    | δ ← ∑i |diag(Bnouveau)i - diag(B)i|
    | A ← Bnouveau
    | k ← k + 1
    |
    pour i = 1 à n faire
        si 1[i, i] > ε alors
            Σ2[i, i] ← V[i, i]
        sinon
            Σ2[i, i] ← 0
    retourner Σ2, Qacc

```

---

5- [

## 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\*)

$v_i \leftarrow i^e$  colonne de  $V$

$u_i \leftarrow A \cdot v_i$

$u_i \leftarrow u_i / \sigma$

    normaliser  $u_i$

    insérer  $u_i$  comme  $i^e$  colonne de  $U$

## Reconstruction 3D

2025-05-19

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

```

Entrée:  $A \in \mathbb{R}^{m \times n}$  tels que  $A \approx U\Sigma V^T$ 
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  $i = 1$  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*)
     $v_i \leftarrow i^e$  colonne de  $V$ 
     $u_i \leftarrow A \cdot v_i$ 
     $u_i \leftarrow u_i / \sigma$ 
    normaliser  $u_i$ 
    insérer  $u_i$  comme  $i^e$  colonne de  $U$ 

```