

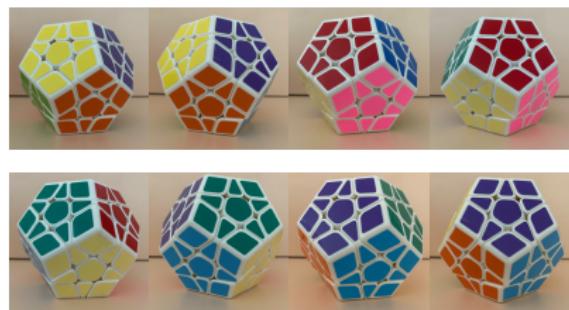
Reconstruction d'objets convexes à partir de photographies

Présentation de **Lucie-Hélène Cuingnet**

Travail réalisé avec **Barnabé Baruchel**

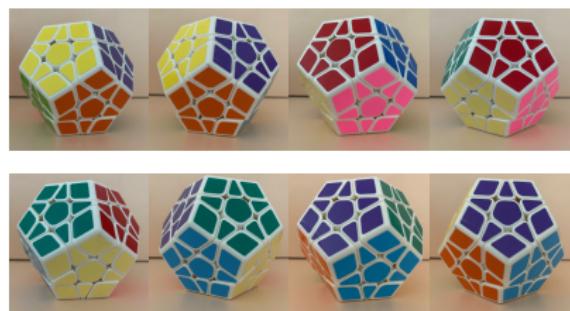
TIPE 2025

Définition du problème

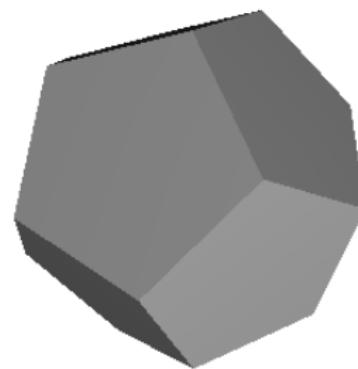


Données

Définition du problème



Données



Objectif

1. Selection

- Points d'intérêts
- Algorithme
- Implémentation

2. Appariement

3. Reconstruction

4. Reconstruction des points

5. Enveloppe Convexe

6. Considérations pratiques et résultats

Points d'intérêts



Contrainte de convexité

Exemple

Algorithme: Sélection de points d'intérêts

Input: Image d'intensité image

Exemple

Algorithme: Sélection de points d'intérêts

Input: Image d'intensité image

Output: Matrice des coins détecté

Exemple

Algorithme: Sélection de points d'intérêts

Input: Image d'intensité image

Output: Matrice des coins détecté

Exemple

Algorithme: Sélection de points d'intérêts

Input: Image d'intensité `image`

Output: Matrice des coins détecté

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

Exemple

Algorithme: Sélection de points d'intérêts

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

Exemple

Algorithme: Sélection de points d'intérêts

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

Algorithme: Sélection de points d'intérêts

Input: Image d'intensité image

Output: Matrice des coins détecté

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

 └ $\text{scores} \leftarrow$ 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

Algorithme: Sélection de points d'intérêts

Input: Image d'intensité `image`

Output: Matrice des coins détecté

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

 └ `scores` \leftarrow liste vide;

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

faire

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

 Ajouter la variance à `scores`;

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

Exemple**Algorithme:** Sélection de points d'intérêts**Input:** Image d'intensité `image`**Output:** Matrice des coins détecté**pour tout** pixel (x, y) dans l'image **faire** `scores` \leftarrow liste vide;**pour tout** direction (dx, dy) parmi : verticale, horizontale, diagonales
faire Calculer la variance locale autour de (x, y) dans la direction
 (dx, dy) ; Ajouter la variance à `scores`;`score` $\leftarrow \min(scores)$;**if** `score` $>$ SEUIL **then**

Exemple

Algorithme: Sélection de points d'intérêts

Input: Image d'intensité *image*

Output: Matrice des coins détecté

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

 └ scores \leftarrow liste vide;

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

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

 └ Ajouter la variance à *scores*;

 score $\leftarrow \min(scores)$;

if score > SEUIL **then**

 └ Marquer (x, y) comme coin;

Exemple

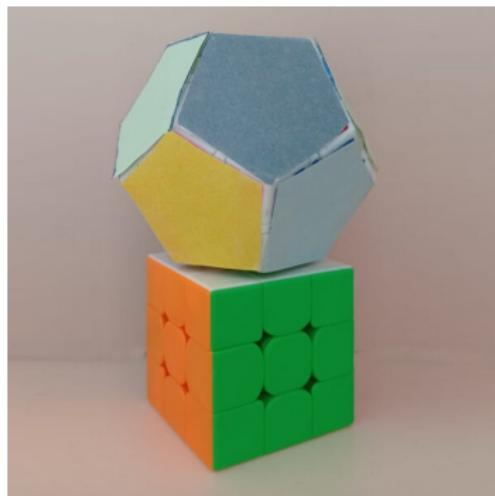
Algorithme: Sélection de points d'intérêts

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

Fichier de sortie PBM



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

Influence des paramètres

Paramètres utilisés :

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



Influence des paramètres

Paramètres utilisés :

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



Influence des paramètres

Paramètres utilisés :

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



Influence des paramètres

Paramètres utilisés :

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



1. Selection

2. Appariement

- Pré-traitement
- Filtrage epipolaire
- Descripteur
- Algorithme
- Resultat

3. Reconstruction

4. Reconstruction des points

5. Enveloppe Convexe

6. Considérations pratiques et résultats

2- Appariement

Le problème de l'appariement



2- Appariement

Le problème de l'appariement

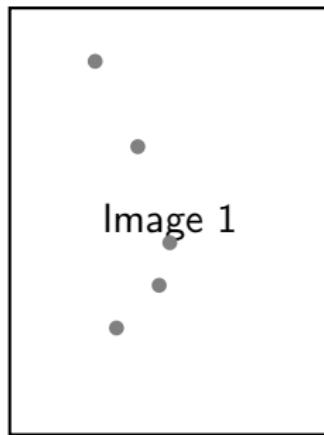


Image 1

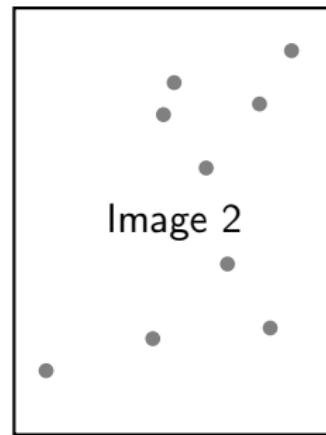


Image 2

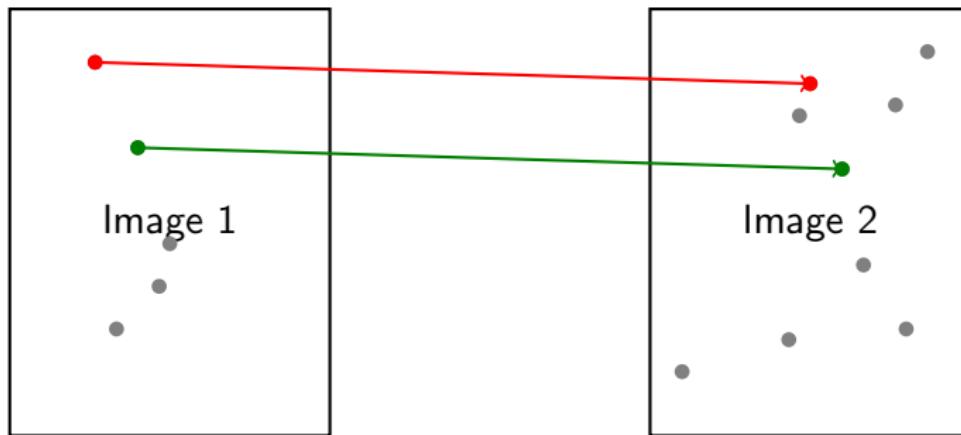
2- Appariement

Le problème de l'appariement



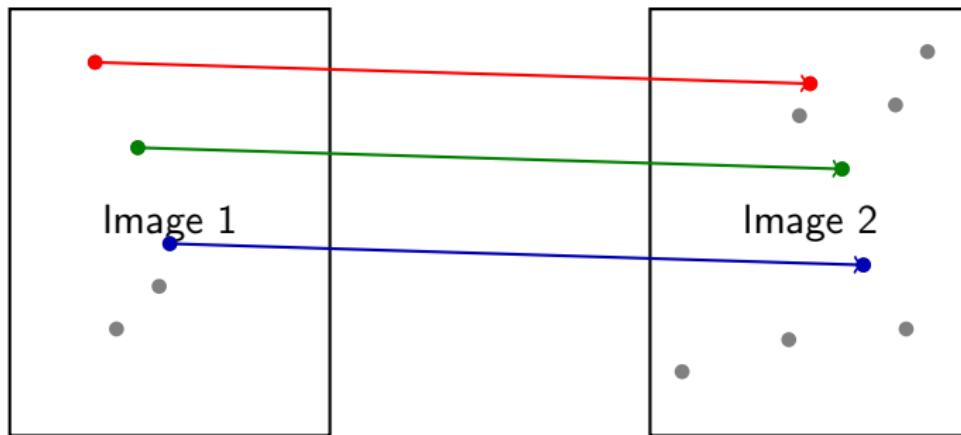
2- Appariement

Le problème de l'appariement



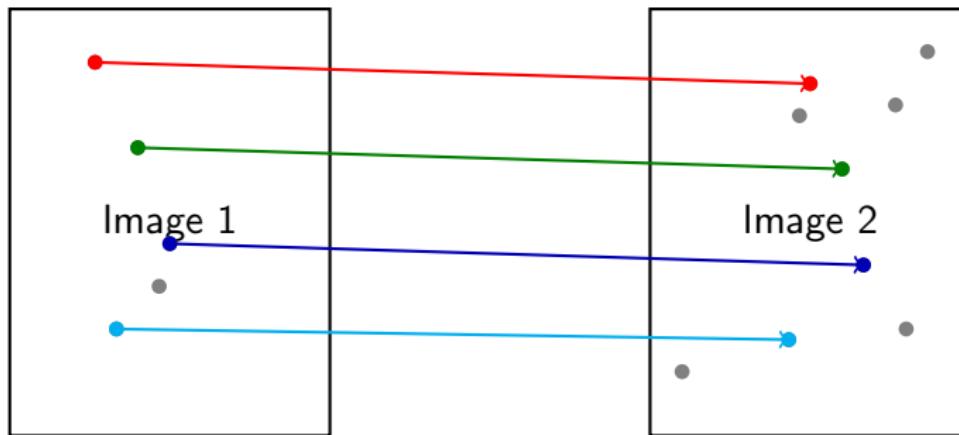
2- Appariement

Le problème de l'appariement



2- Appariement

Le problème de l'appariement



Pré-traitement



Sélection

Pré-traitement

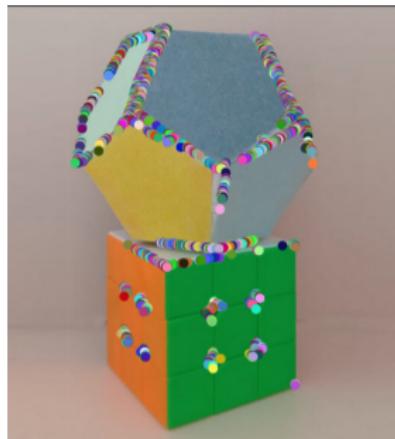


Sélection



Trouve coin

Pré-traitement



Sélection



Trouve coin



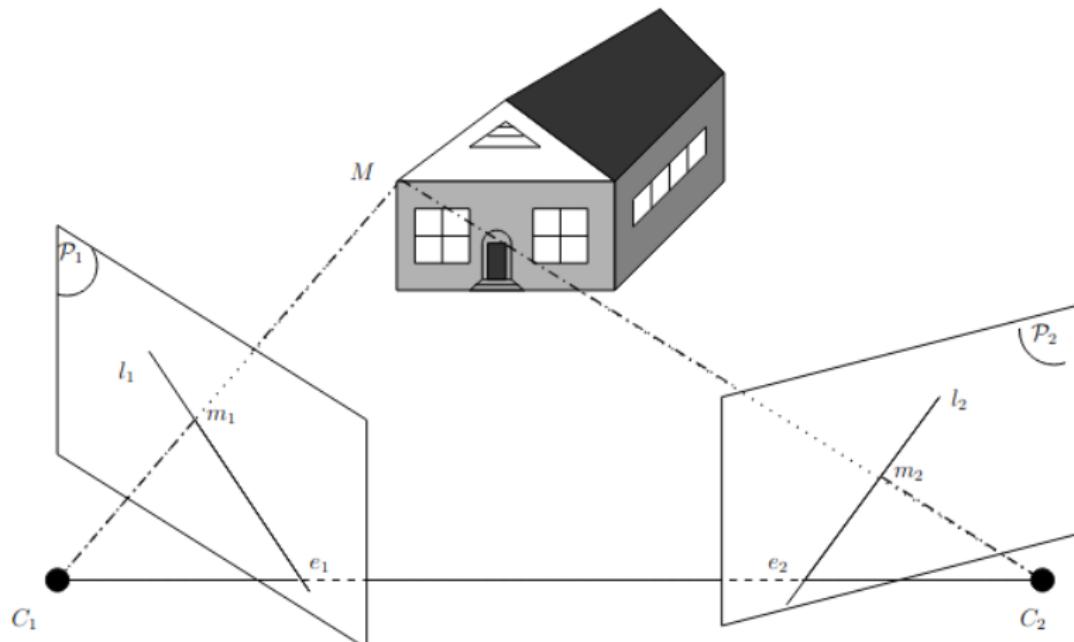
Ransac

Explication Ransac

Appariement

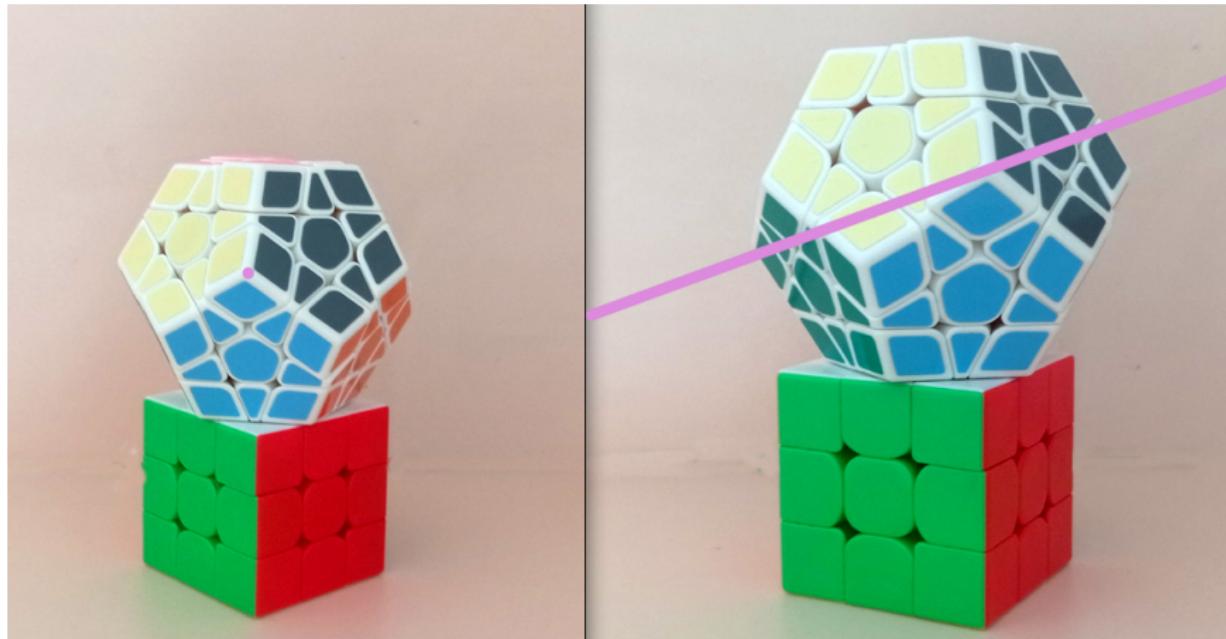


Geometrie epipolaire



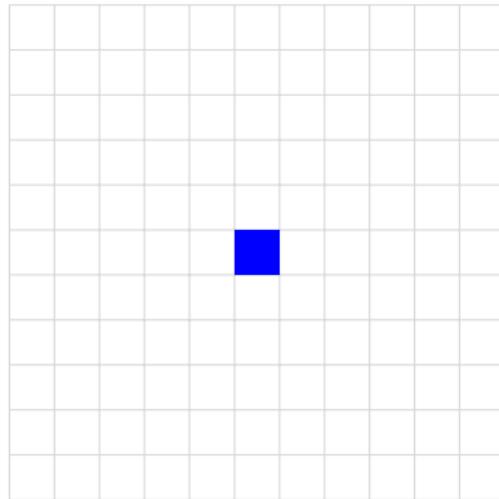
Source image : Quelques problèmes géométriques en vision par ordinateur - Frédéric SUR

Filtrage epipolaire



Droite epipolaire

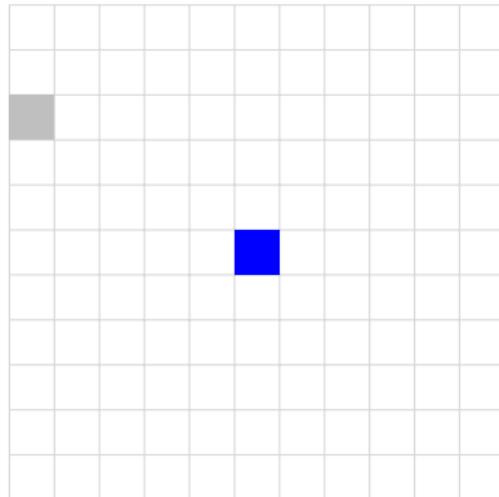
Descripteur BRIEF : Principe



Comparaisons binaires :

- On considère une fenêtre autour d'un point clé, ici le centre $(0, 0)$.

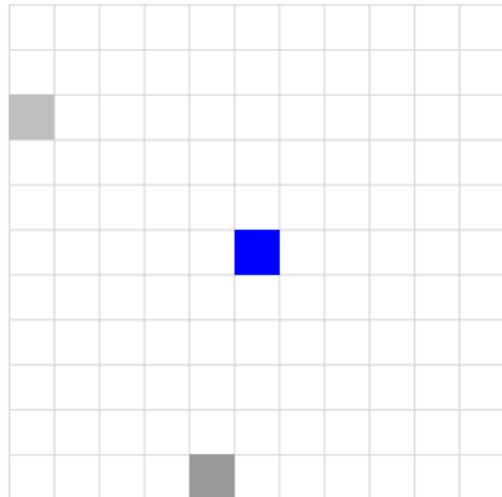
Descripteur BRIEF : Principe



Comparaisons binaires :

- On considère une fenêtre autour d'un point clé, ici le centre $(0, 0)$.
- On choisit aléatoirement un pixel (ex : $(-5, -3)$).

Descripteur BRIEF : Principe

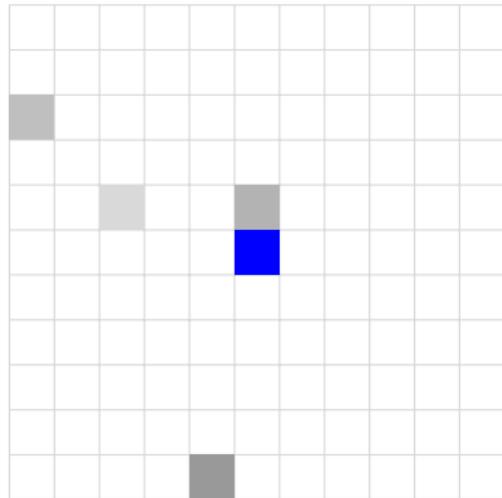


Comparaisons binaires :

$$I(-5, -3) < I(-1, 5) \Rightarrow 1$$

- On considère une fenêtre autour d'un point clé, ici le centre $(0, 0)$.
- On choisit aléatoirement un pixel (ex : $(-5, -3)$).
- On le compare à un autre (ex : $(-1, 5)$) :
 $BRIEF_1 = 1$ si intensité(1er) < intensité(2e)

Descripteur BRIEF : Principe

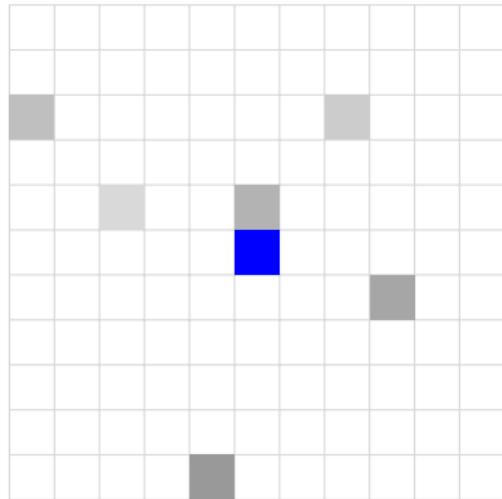


Comparaisons binaires :

$$\begin{aligned} I(-5, -3) &< I(-1, 5) & \Rightarrow 1 \\ I(0, -1) &> I(-3, -1) & \Rightarrow 0 \end{aligned}$$

- On considère une fenêtre autour d'un point clé, ici le centre $(0, 0)$.
- On choisit aléatoirement un pixel (ex : $(-5, -3)$).
- On le compare à un autre (ex : $(-1, 5)$) :
 $BRIEF_1 = 1$ si intensité(1er) < intensité(2e)
- On répète avec d'autres paires...

Descripteur BRIEF : Principe

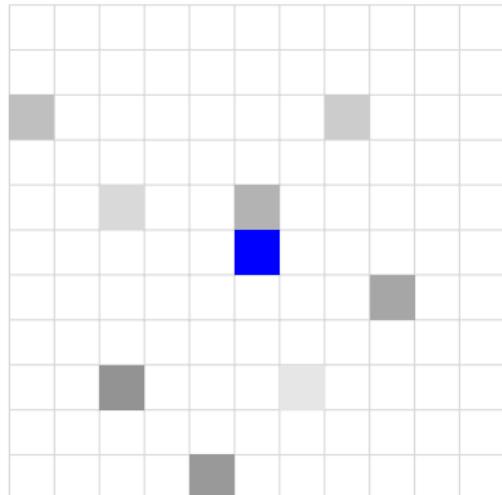


Comparaisons binaires :

$$\begin{aligned} I(-5, -3) < I(-1, 5) &\Rightarrow 1 \\ I(0, -1) > I(-3, -1) &\Rightarrow 0 \\ I(2, -3) < I(3, 1) &\Rightarrow 1 \end{aligned}$$

- On considère une fenêtre autour d'un point clé, ici le centre $(0, 0)$.
- On choisit aléatoirement un pixel (ex : $(-5, -3)$).
- On le compare à un autre (ex : $(-1, 5)$) :
 $BRIEF_1 = 1$ si intensité(1er) < intensité(2e)
- On répète avec d'autres paires...

Descripteur BRIEF : Principe

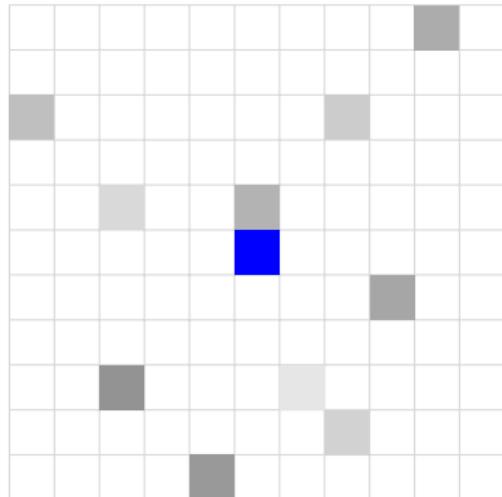


Comparaisons binaires :

$$\begin{array}{ll} I(-5,-3) < I(-1,5) & \Rightarrow 1 \\ I(0,-1) > I(-3,-1) & \Rightarrow 0 \\ I(2,-3) < I(3,1) & \Rightarrow 1 \\ I(-3,3) > I(1,3) & \Rightarrow 0 \end{array}$$

- On considère une fenêtre autour d'un point clé, ici le centre $(0, 0)$.
- On choisit aléatoirement un pixel (ex : $(-5, -3)$).
- On le compare à un autre (ex : $(-1, 5)$) :
 $BRIEF_1 = 1$ si intensité(1er) < intensité(2e)
- On répète avec d'autres paires...

Descripteur BRIEF : Principe

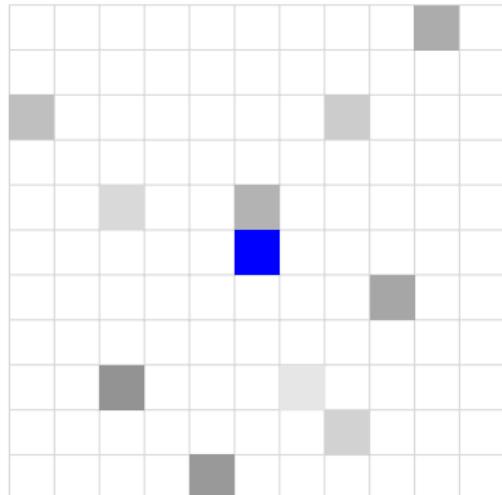


Comparaisons binaires :

$$\begin{aligned} I(-5,-3) < I(-1,5) &\Rightarrow 1 \\ I(0,-1) > I(-3,-1) &\Rightarrow 0 \\ I(2,-3) < I(3,1) &\Rightarrow 1 \\ I(-3,3) > I(1,3) &\Rightarrow 0 \\ I(2,4) < I(4,-5) &\Rightarrow 1 \end{aligned}$$

- On considère une fenêtre autour d'un point clé, ici le centre $(0, 0)$.
- On choisit aléatoirement un pixel (ex : $(-5, -3)$).
- On le compare à un autre (ex : $(-1, 5)$) :
 $BRIEF_1 = 1$ si intensité(1er) < intensité(2e)
- On répète avec d'autres paires...

Descripteur BRIEF : Principe



Comparaisons binaires :

$$\begin{array}{lcl} I(-5,-3) < I(-1,5) & \Rightarrow 1 \\ I(0,-1) > I(-3,-1) & \Rightarrow 0 \\ I(2,-3) < I(3,1) & \Rightarrow 1 \\ I(-3,3) > I(1,3) & \Rightarrow 0 \\ I(2,4) < I(4,-5) & \Rightarrow 1 \end{array}$$

Descripteur final :

1 0 1 0 1

- On considère une fenêtre autour d'un point clé, ici le centre $(0, 0)$.
- On choisit aléatoirement un pixel (ex : $(-5, -3)$).
- On le compare à un autre (ex : $(-1, 5)$) :
 $BRIEF_1 = 1$ si intensité(1er) < intensité(2e)
- On répète avec d'autres paires...

Amélioration progressive du descripteur BRIEF

1. BRIEF (intensité)

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

Amélioration progressive du descripteur BRIEF

1. BRIEF (intensité)

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

2. BRIEF RGB

- Comparaison faite indépendamment sur les 3 canaux : R, G, B



(255,0,0)



(255,128,0)



(0,255,0)



(128,255,128)

Rouge vs Orange

Vert foncé vs Vert clair

- → *RGB ne reflète pas toujours 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



(255,0,0)



(255,128,0)



(0,255,0)



(128,255,128)

Rouge vs Orange

Vert foncé vs Vert clair

- → *RGB ne reflète pas toujours la perception humaine*

3. BRIEF Lab

- Comparaison dans l'espace Lab :
 - L : luminosité (luminance)
 - a, b : composantes de couleur perceptuelles

Comparaison de descripteurs BRIEF : distance de Hamming

Descripteur 1 (image gauche)

1 0 1 0 1

Comparaison de descripteurs BRIEF : distance de Hamming

Descripteur 1 (image gauche)

1 0 1 0 1

Descripteur 2 (image droite)

1 1 0 0 1

Comparaison de descripteurs BRIEF : distance de Hamming

Descripteur 1 (image gauche)

1 0 1 0 1

Descripteur 2 (image droite)

1 1 0 0 1

Comparaison bit à bit :

Bit 1	Bit 2	Bit 3	Bit 4	Bit 5
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 :

Bit 1	Bit 2	Bit 3	Bit 4	Bit 5
1	0	1	0	1
1	1	0	0	1
0	1	1	0	0

Distance de Hamming = nombre de bits différents = 2

Pseudo-code : Appariement de points

Algorithme: Appariement

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

Sortie: Liste de correspondances fiables

Pré-tri des points sur image 1

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

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

retirer p

 // suppression non maximale locale

Pseudo-code : Appariement de points

Algorithme: Appariement

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

Sortie: Liste de correspondances fiables

Pré-tri des points sur image 1

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

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

 retirer p

 // suppression non maximale locale

Filtrage épipolaire

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

$I \leftarrow F \cdot p_1$

 // droite épipolaire dans image 2

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

Pseudo-code : Appariement de points

Algorithme: Appariement

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

Sortie: Liste de correspondances fiables

Pré-tri des points sur image 1

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

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

 retirer p

 // suppression non maximale locale

Filtrage épipolaire

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

$I \leftarrow F \cdot p_1$

 // droite épipolaire dans image 2

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

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)

Pseudo-code : Appariement de points

Algorithme: Appariement

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

Sortie: Liste de correspondances fiables

Pré-tri des points sur image 1

pour tout $p \in P_1$ faire

si p n'est pas un coin alors

 retirer p

 // suppression non maximale locale

Filtrage épipolaire

pour tout $p_1 \in P_1$ faire

$I \leftarrow F \cdot p_1$

 // droite épipolaire dans image 2

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

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

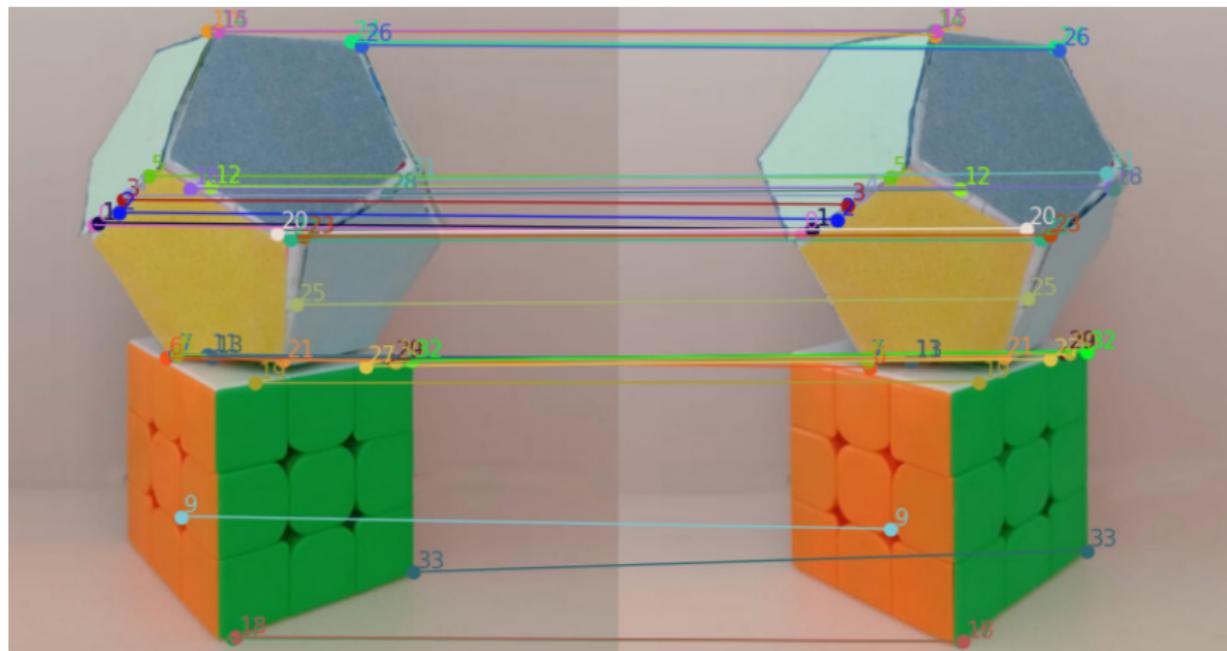
$d_2 \leftarrow \text{BRIEF}(p_2)$

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

 enregistrer (p_1, p_2, h)

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

Résultats appariement



Seuil brief

Comparaison avec Sift

1. Selection

2. Appariement

3. Reconstruction

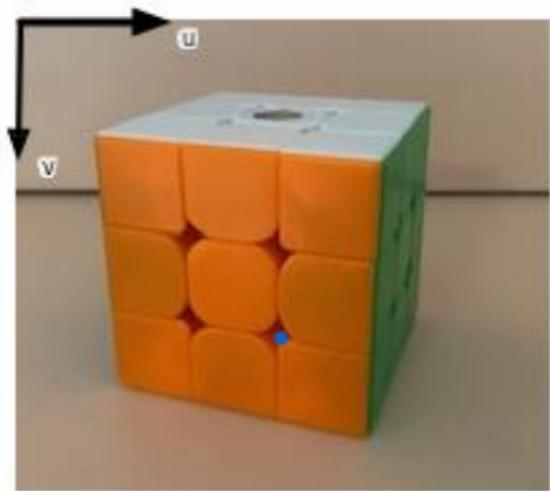
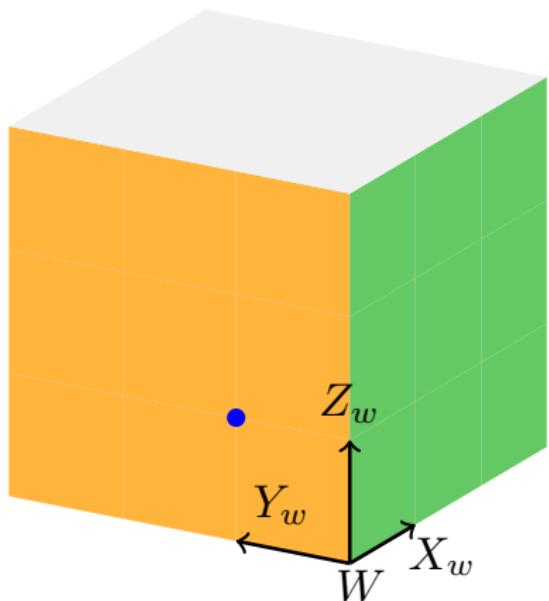
- Détermination du système
- Résolution système
- Implémentation

4. Reconstruction des points

5. Enveloppe Convexe

6. Considérations pratiques et résultats

Les différents repères



Représentation du cube (vue 3D)

Cube sur une image

Modèle de projection — Matrice P

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

Modèle de projection — Matrice P

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

Compléments projections

Modèle de projection — Matrice P

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

Compléments projections

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

Modèle de projection — Matrice P

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

Compléments projections

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

- P est une matrice 3×4 , avec 12 inconnues

Modèle de projection — Matrice P

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

Compléments projections

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

- P est une matrice 3×4 , avec 12 inconnues
- En faisant cela pour 6 points on obtient un système ...

Système matriciel

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

Calibration

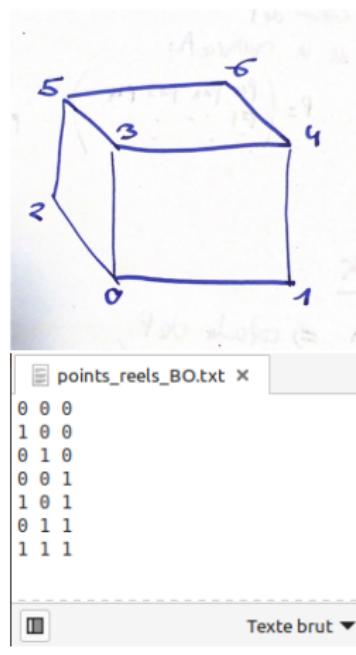


Figure – Interface utilisateur

Résolution système

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

Problème d'optimisation sous contrainte

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

On reformule le problème comme une minimisation sous contrainte :

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

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

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

$$\min \|Ap\| \Rightarrow p = \text{un v.p. associé à la plus petite v.a.p. de } A^T A$$

Décomposition en valeurs singulières (SVD)

- On factorise A :

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

Décomposition en valeurs singulières (SVD)

- On factorise A :

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

- U : matrice orthogonale de $\mathbb{R}^{m \times m}$

Décomposition en valeurs singulières (SVD)

- On factorise A :

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

- U : matrice orthogonale de $\mathbb{R}^{m \times m}$
- Σ : matrice diagonale $\mathbb{R}^{m \times n}$ contenant les valeurs singulières
 $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_n \geq 0$

Décomposition en valeurs singulières (SVD)

- On factorise A :

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

- U : matrice orthogonale de $\mathbb{R}^{m \times m}$
- Σ : matrice diagonale $\mathbb{R}^{m \times n}$ contenant les valeurs singulières
 $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_n \geq 0$
- V : matrice orthogonale de $\mathbb{R}^{n \times n}$, ses colonnes sont les vecteurs propres de $A^T A$, tels que :

$$A^T A v_i = \sigma_i^2 v_i$$

Algorithme QR : principe de convergence

Décomposition QR pour l'itération :

- Si $A = QR$, avec :
 - Q une matrice orthogonale ($Q^T Q = I$),
 - R une matrice triangulaire supérieure,

Algorithme QR : principe de convergence

Décomposition QR pour l'itération :

- Si $A = QR$, avec :
 - Q une matrice orthogonale ($Q^T Q = I$),
 - R une matrice triangulaire supérieure,

alors on définit :

$$A' = RQ$$

Algorithme QR : principe de convergence

Décomposition QR pour l'itération :

- Si $A = QR$, avec :
 - Q une matrice orthogonale ($Q^T Q = I$),
 - R une matrice triangulaire supérieure,

alors on définit :

$$A' = RQ$$

- A' est **semblable** à A , car :

$$Q^T A Q = Q^T Q R Q = R Q = A'$$

Algorithme QR : principe de convergence

Décomposition QR pour l'itération :

- Si $A = QR$, avec :
 - Q une matrice orthogonale ($Q^T Q = I$),
 - R une matrice triangulaire supérieure,

alors on définit :

$$A' = RQ$$

- A' est **semblable** à A , car :

$$Q^T A Q = Q^T Q R Q = R Q = A'$$

Construction de la suite :

$$\begin{cases} A_0 = A \\ \text{À chaque itération : } A_k = Q_k R_k \\ A_{k+1} = R_k Q_k \end{cases}$$

Algorithme QR : principe de convergence

Décomposition QR pour l'itération :

- Si $A = QR$, avec :
 - Q une matrice orthogonale ($Q^T Q = I$),
 - R une matrice triangulaire supérieure,

alors on définit :

$$A' = RQ$$

- A' est **semblable** à A , car :

$$Q^T A Q = Q^T Q R Q = R Q = A'$$

Construction de la suite :

$$\begin{cases} A_0 = A \\ \text{À chaque itération : } A_k = Q_k R_k \\ A_{k+1} = R_k Q_k \end{cases}$$

- Cette suite (A_k) converge vers une matrice triangulaire.
- Les valeurs propres de A_k sont identiques à celles de A .

Résolution avec SVD

Pseudo-code détaillé

- Produit symétrique : $A^T A \in \mathbb{R}^{n \times n}$

Résolution avec SVD

Pseudo-code détaillé

- Produit symétrique : $A^T A \in \mathbb{R}^{n \times n}$
- Décomposition QR :

$$A^T A = Q_{k-1} \cdots Q_0 A_k Q_0^T \cdots Q_{k-1}^T$$

Résolution avec SVD

Pseudo-code détaillé

- Produit symétrique : $A^T A \in \mathbb{R}^{n \times n}$
- Décomposition QR :

$$A^T A = Q_{k-1} \cdots Q_0 A_k Q_0^T \cdots Q_{k-1}^T$$

- Matrice des vecteurs propres :

$$V = Q_0 Q_1 \cdots Q_{k-1}$$

Résolution avec SVD

Pseudo-code détaillé

- Produit symétrique : $A^T A \in \mathbb{R}^{n \times n}$
- Décomposition QR :

$$A^T A = Q_{k-1} \cdots Q_0 A_k Q_0^T \cdots Q_{k-1}^T$$

- Matrice des vecteurs propres :

$$V = Q_0 Q_1 \cdots Q_{k-1}$$

- Extraction du vecteur propre associé à la plus petite valeur propre non nulle :

$$v_{i_0} \in V$$

1. Selection
2. Appariement
3. Reconstruction
- 4. Reconstruction des points**
5. Enveloppe Convexe
6. Considérations pratiques et résultats

4- Reconstruction des points

Reconstruction : retrouver les coordonnées 3D

- On connaît les matrices de projection P_1 et P_2

4- Reconstruction des points

Reconstruction : retrouver les coordonnées 3D

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

$$m_1 = (u_1, v_1), \quad m_2 = (u_2, v_2)$$

4- Reconstruction des points

Reconstruction : retrouver les coordonnées 3D

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

$$m_1 = (u_1, v_1), \quad m_2 = (u_2, v_2)$$

- Point 3D inconnu : $M = (X, Y, Z)$

Reconstruction : retrouver les coordonnées 3D

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

$$m_1 = (u_1, v_1), \quad m_2 = (u_2, v_2)$$

- Point 3D inconnu : $M = (X, Y, Z)$

- On a :

$$\lambda_1 m_1 = P_1 M, \quad \lambda_2 m_2 = P_2 M$$

Reconstruction : retrouver les coordonnées 3D

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

$$m_1 = (u_1, v_1), \quad m_2 = (u_2, v_2)$$

- Point 3D inconnu : $M = (X, Y, Z)$
- On a :
$$\lambda_1 m_1 = P_1 M, \quad \lambda_2 m_2 = P_2 M$$
- 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 :

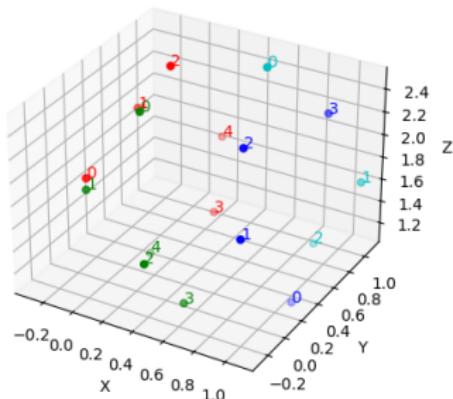
$$m_1 = (u_1, v_1), \quad m_2 = (u_2, v_2)$$

- Point 3D inconnu : $M = (X, Y, Z)$
- On a :
$$\lambda_1 m_1 = P_1 M, \quad \lambda_2 m_2 = P_2 M$$
- Système de 6 équations linéaires homogènes
- Résolution par SVD

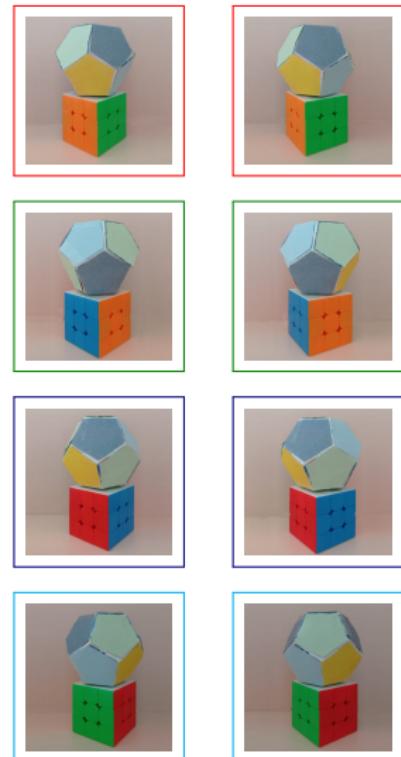
4- Reconstruction des points

Reconstruction 3D multi-vues

Points 3D

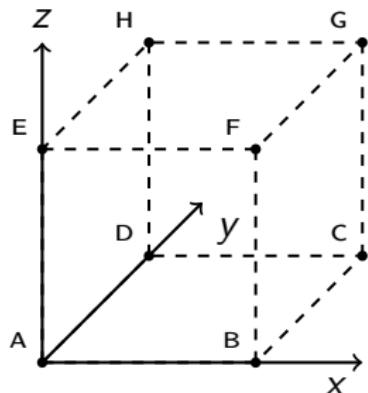


Nuage de points



1. Selection
2. Appariement
3. Reconstruction
4. Reconstruction des points
- 5. Enveloppe Convexe**
6. Considérations pratiques et résultats

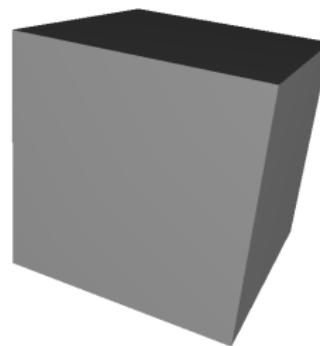
Triangulation



5- Enveloppe Convexe

Le format STL

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



fichier STL visualisé avec viewstl.com

1. Selection
2. Appariement
3. Reconstruction
4. Reconstruction des points
5. Enveloppe Convexe
6. Considérations pratiques et résultats
 - En pratique
 - Quelques exemples
 - Des pistes d'amélioration
 - Limite de la méthode

Implémentation

- Cœur du projet en **C** : traitement, calculs, reconstruction.

Implémentation

- Cœur du projet en C : traitement, calculs, reconstruction.

Fichier	Rôle
matrice.c	Implémente les opérations sur les matrices.
selection.c	Sélectionne des points d'intérêt.
trouve_coin.c	Raffine les points détectés.
ransac.c	Second raffinement des points par RANSAC.
appariement.c	Calcule le descripteur BRIEF et la distance à la droite épipolaire.
detection.c	Regroupe détection, tri et appariement pour un couple d'images.
camera_calibration.c	Calibre la caméra à partir des correspondances 2D.
SVD.c	Fonctions pour la décomposition SVD et résolution de systèmes.
reconstruction.c	Reconstruit les coordonnées 3D à partir des points 2D.
triangles.c	Détermine les triangles pour reconstruire l'enveloppe 3D.
manipulation_fichier.c	Utilitaires de lecture/écriture de matrices.
constante.c	Gère les paramètres de reconstruction.

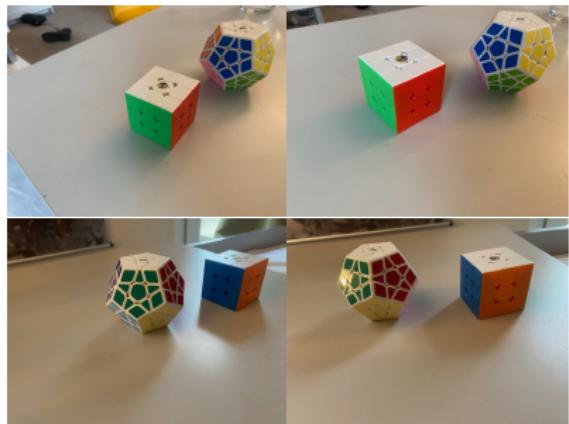
Implémentation

- Cœur du projet en **C** : traitement, calculs, reconstruction.

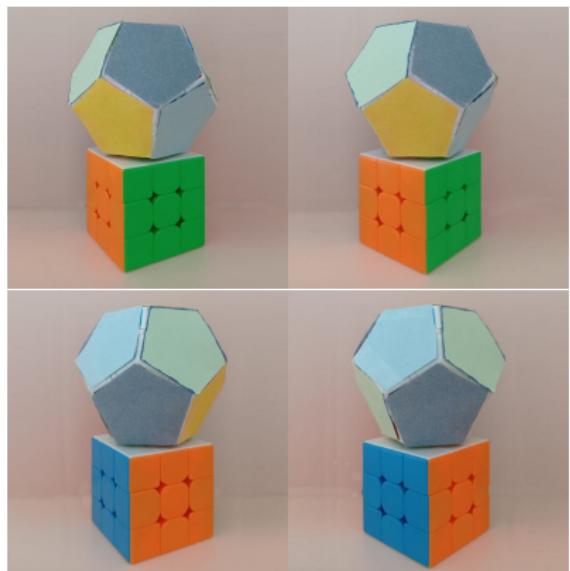
Fichier	Rôle
matrice.c	Implémente les opérations sur les matrices.
selection.c	Sélectionne des points d'intérêt.
trouve_coin.c	Raffine les points détectés.
ransac.c	Second raffinement des points par RANSAC.
appariement.c	Calcule le descripteur BRIEF et la distance à la droite épipolaire.
detection.c	Regroupe détection, tri et appariement pour un couple d'images.
camera_calibration.c	Calibre la caméra à partir des correspondances 2D.
SVD.c	Fonctions pour la décomposition SVD et résolution de systèmes.
reconstruction.c	Reconstruit les coordonnées 3D à partir des points 2D.
triangles.c	Détermine les triangles pour reconstruire l'enveloppe 3D.
manipulation_fichier.c	Utilitaires de lecture/écriture de matrices.
constante.c	Gère les paramètres de reconstruction.

- Interface utilisateur et affichage en **Python**.

Shooting photo : importance de la prise de vue

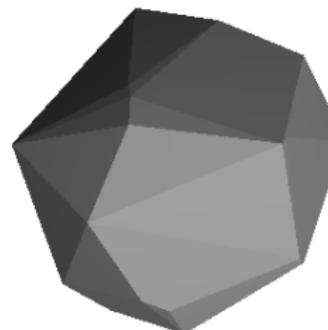
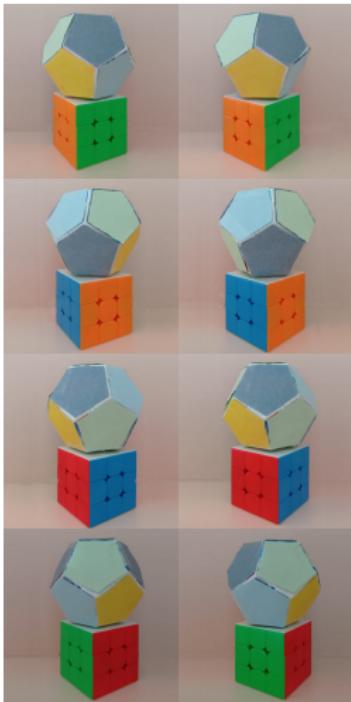


Vues initiales



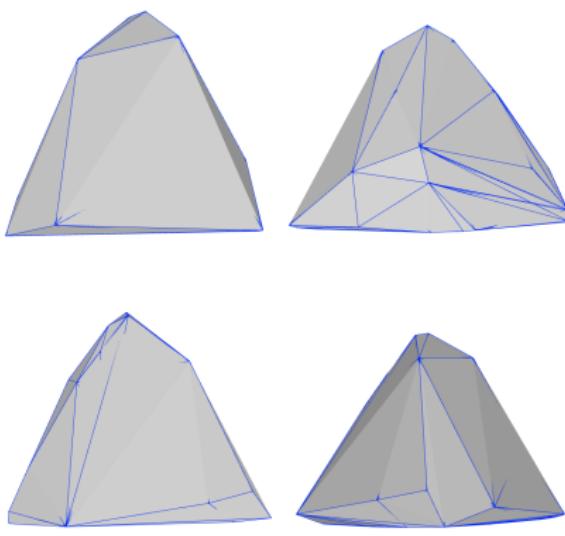
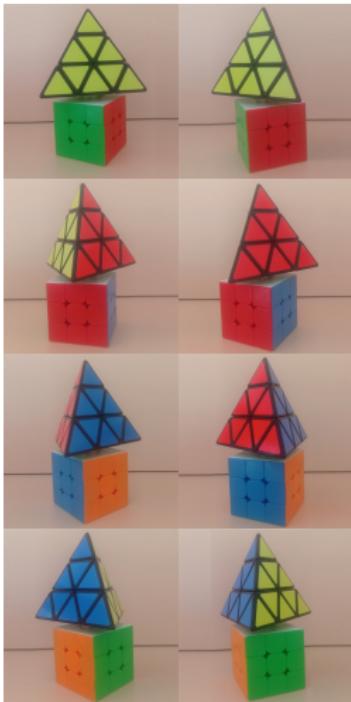
Vues améliorées

Le dodécahèdre



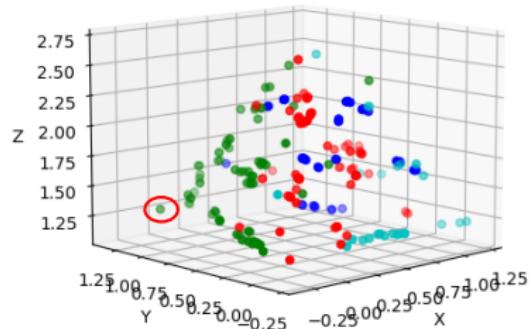
visualisation sur viewstl.com

La pyramide

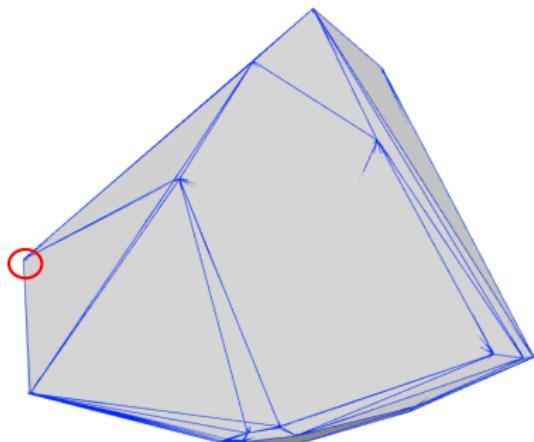


visualisation sur 3dviewer.net

Une importante sensibilité aux erreurs



points localisés par le programme

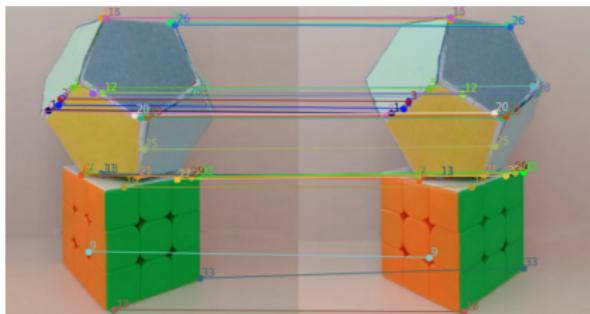


*enveloppe convexe visualisé avec
3dview.net*

Fin

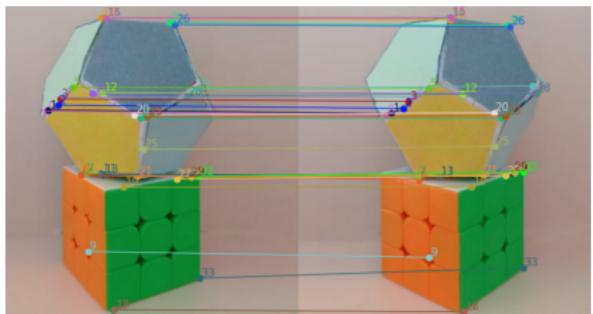
Merci pour votre attention

comparaison avec SIFT opencv

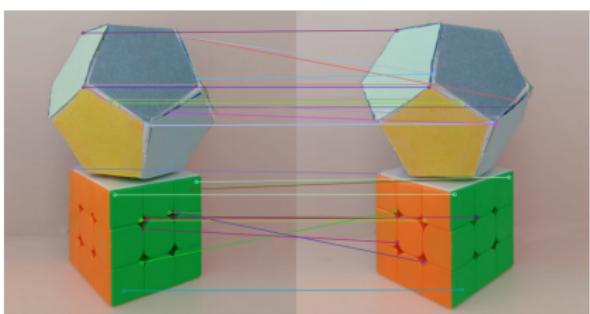


Algorithme personnalisé

comparaison avec SIFT opencv

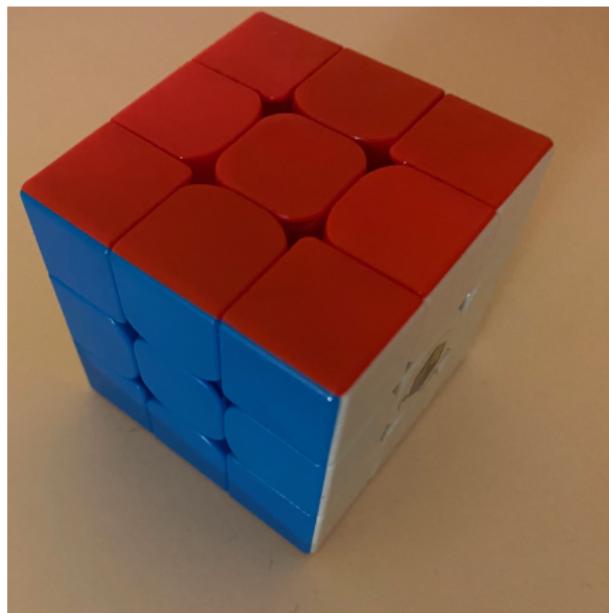


Algorithm personnalisé

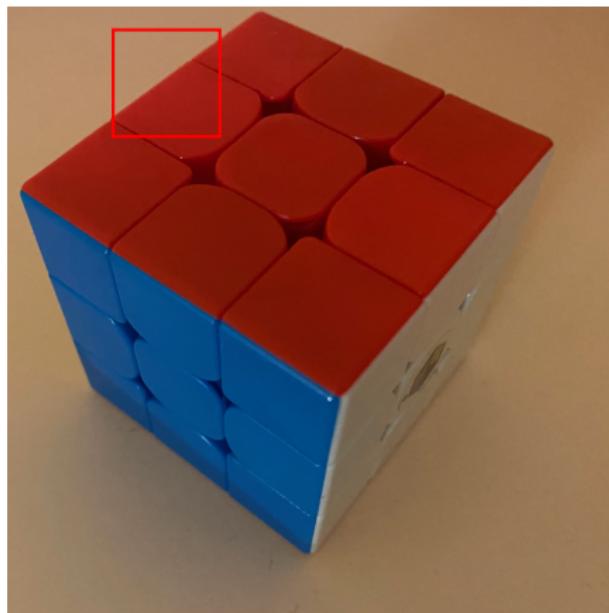


SIFT

Exemple Moravec



Exemple Moravec



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

$w = 2$

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

$i = -2$

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

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

$i = -1$

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

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

$$i = 2$$

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

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

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

$$w = 2$$

$$T = 300$$

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

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

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

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

$$S = 386.8$$

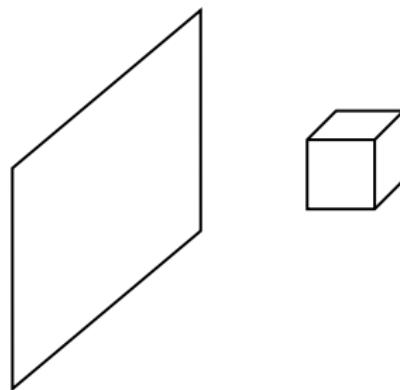
$S > T \implies$ point d'intérêt

7- • 7.1 Compléments

Voir le schéma détaillé

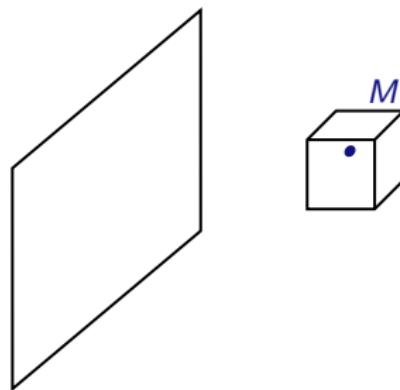
7- • 7.1 Compléments

Les différents repères

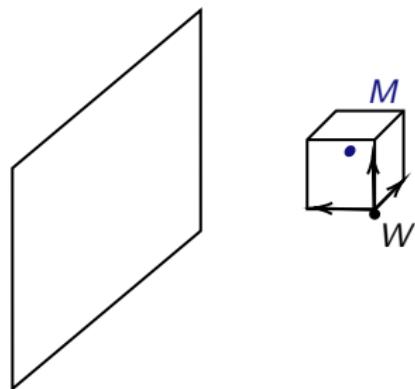


Les différents repères

- M : point réel

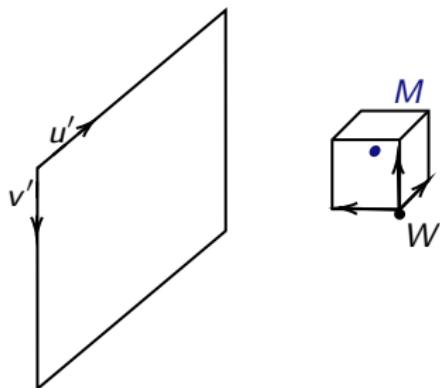


Les différents repères



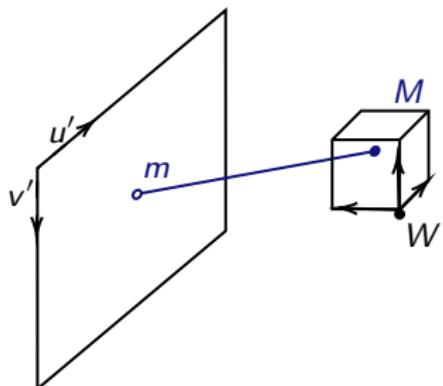
- M : point réel
- W : origine du repère du monde

Les différents repères



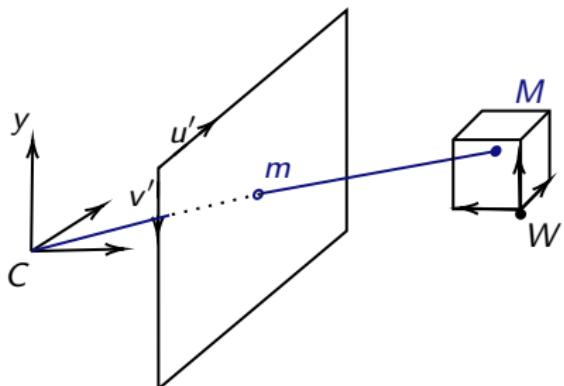
- M : point réel
- W : origine du repère du monde
- (u', v') : coordonnées dans le plan image en pixels

Les différents repères



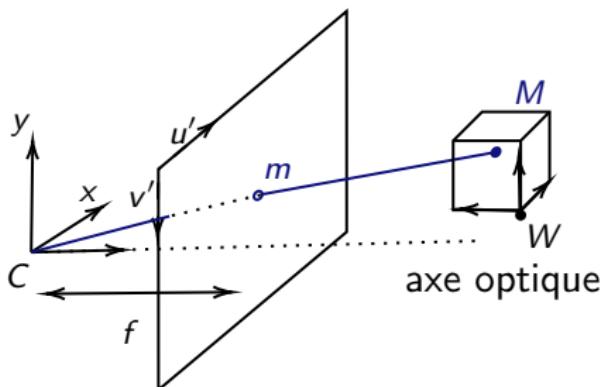
- M : point réel
- W : origine du repère du monde
- (u', v') : coordonnées dans le plan image en pixels
- m : projection de M dans le plan image

Les différents repères



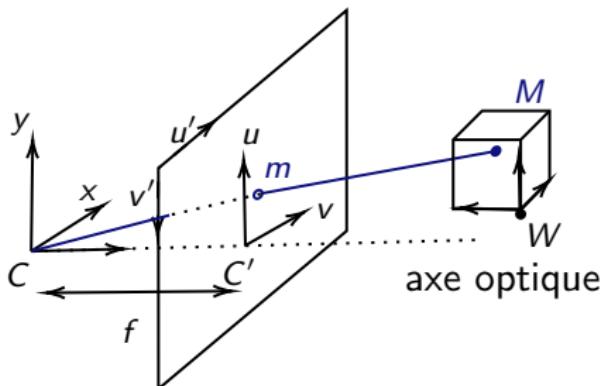
- M : point réel
- W : origine du repère du monde
- (u', v') : coordonnées dans le plan image en pixels
- m : projection de M dans le plan image
- C : origine du repère de la caméra

Les différents repères



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

Les différents repères



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

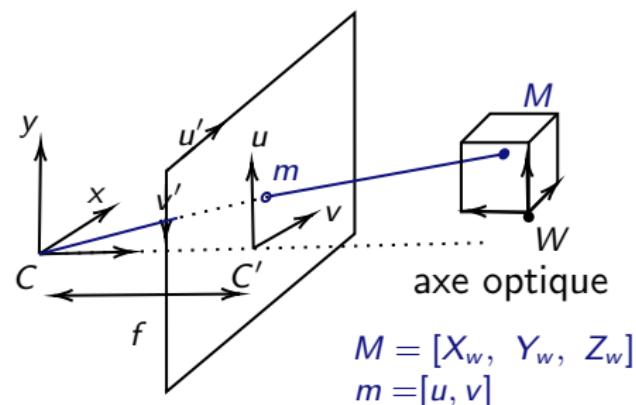
Projection d'un point 3D sur le plan image

Par le théorème de Thalès
(projection perspective)

$$u = fx_c$$

$$v = fy_c$$

$$w = z_c$$



$$M = [X_w, Y_w, Z_w]$$
$$m = [u, v]$$

Projection d'un point 3D sur le plan image

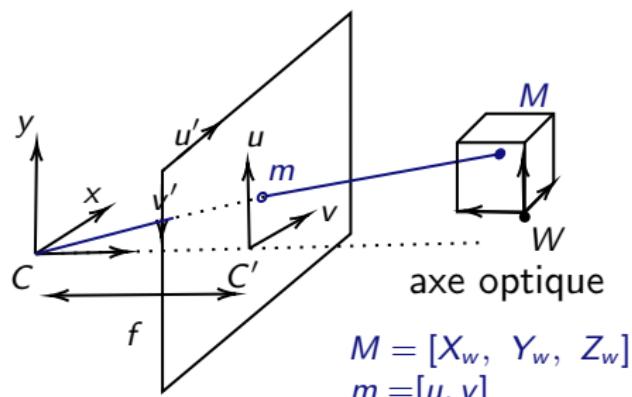
Par le théorème de Thalès
 (projection perspective)

$$u = fx_c$$

$$v = fy_c$$

$$w = z_c$$

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

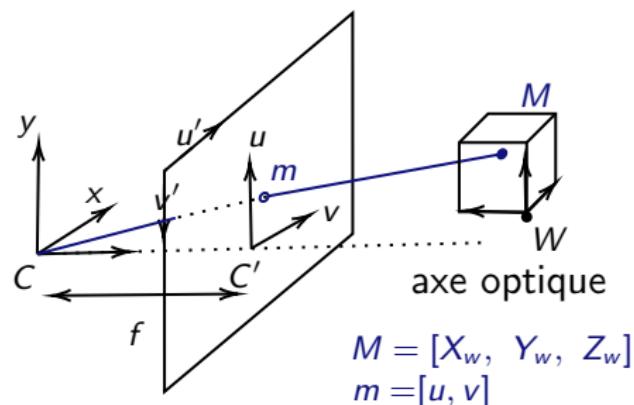


Projection d'un point 3D sur le plan image

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

$$\begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} = [R \quad T] \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

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



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

$$m = [u, v]$$

Projection d'un point 3D sur le plan image

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

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

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

Système d'optimisation à contrainte unitaire

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

Système d'optimisation à contrainte unitaire

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

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

et reformuler le système comme un problème d'optimisation :

Système d'optimisation à contrainte unitaire

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

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

et reformuler le système comme un problème d'optimisation :

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

Système d'optimisation à contrainte unitaire

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

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

et reformuler le système comme un problème d'optimisation :

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

On introduit les fonctions :

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

Système d'optimisation à contrainte unitaire

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

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

et reformuler le système comme un problème d'optimisation :

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

On introduit les fonctions :

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

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

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

Lien avec les valeurs propres

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

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

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 :

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

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 :

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

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 $A^T A$
- λ est la valeur propre associée

Triangulation : formulation du système

- P_1 et P_2 déterminées

Triangulation : formulation du système

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

Triangulation : formulation du système

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

Triangulation : formulation du système

- P_1 et P_2 déterminées
- On cherche les coordonnées $X = (x_C, y_C, z_C, 1)^T$
- Pour chaque paire (x_1, x_2) de projections
- On élimine λ_1, λ_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 λ_1, λ_2 et on écrit un système homogène
- Système sous la forme $AX = 0$

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

Algorithme: Décomposition QR via Gram-Schmidt

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

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

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

$v_j \leftarrow A_{:,j}$

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

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

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

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

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

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

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

sinon

$Q_{:,j} \leftarrow 0$

retourner Q, R

Algorithme: algorithme QR

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

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

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

(*Accumule les produits de Q *)

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

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

$Q, R \leftarrow$ décomposition QR de B

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

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

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

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

$k \leftarrow k + 1$

pour $i = 1$ à n **faire**

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

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

sinon

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

retourner Σ^2, Q_{acc}

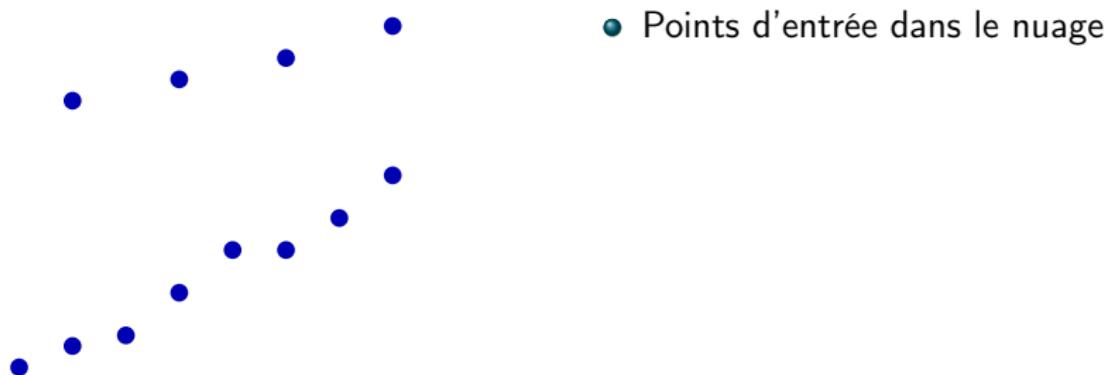
Algorithme: SVD via algorithme QR sur $A^T A$

Entrée: $A \in \mathbb{R}^{m \times n}$ **Sortie:** U, Σ, 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($A^T A, \Sigma^2, V$) (* Σ^2 diagonale, V orthogonale*)**pour** $i \leftarrow 1$ **to** n **faire** $\sigma^2 \leftarrow \Sigma^2[i, i]$ **si** $\sigma^2 < 10^{-12}$ **alors** **continuer**

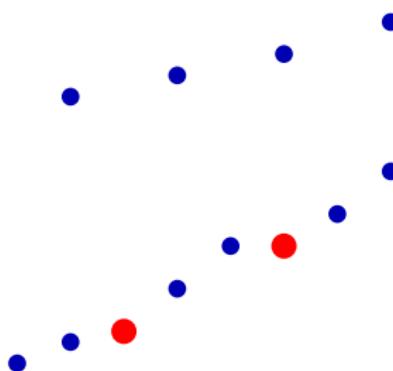
(*Ignorer valeur singulière nulle*)

 $\sigma \leftarrow \sqrt{\sigma^2}$ $\Sigma[i, i] \leftarrow \sigma$ (*Met à jour la vraie valeur singulière*) $v_i \leftarrow i^{\text{e}} \text{ colonne de } V$ $u_i \leftarrow A \cdot v_i$ (* u_i non normalisé*) $u_i \leftarrow u_i / \sigma$ normaliser u_i insérer u_i comme i^{e} colonne de U

Ransac

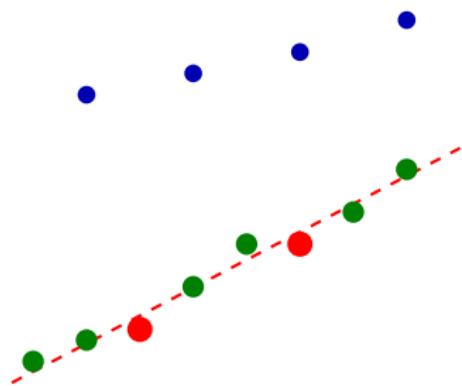


Ransac



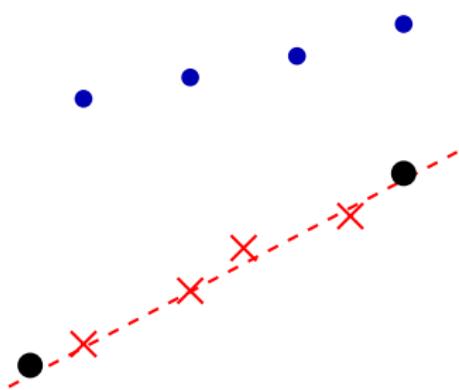
- Points d'entrée dans le nuage
- Sélection de deux points pour estimer une droite

Ransac



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

Ransac

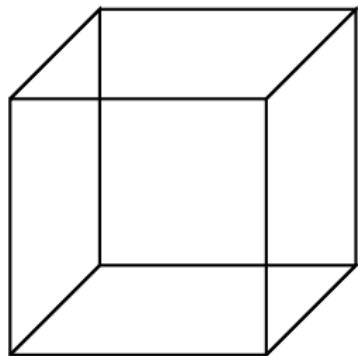


- Points d'entrée dans le nuage
- Sélection de deux points pour estimer une droite
- Inliers détectés à proximité de la droite
- Suppression des points internes, on garde les extrémités

⇒ Nettoyage effectué : seuls les segments visibles sont conservés

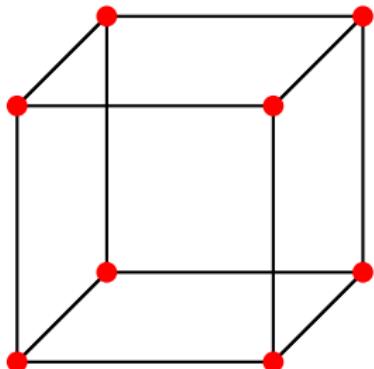
Points d'intérêts

Points d'intérêts sur un cube



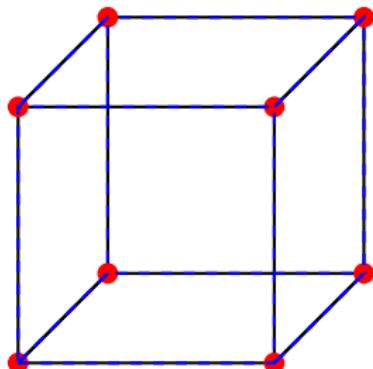
Points d'intérêts

Points d'intérêts sur un cube



Points d'intérêts

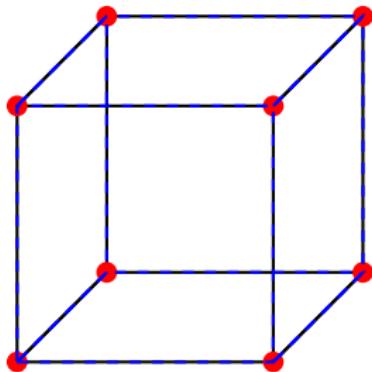
Points d'intérêts sur un cube



Enveloppe convexe

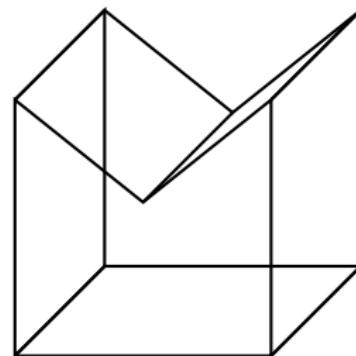
Points d'intérêts

Points d'intérêts sur un cube



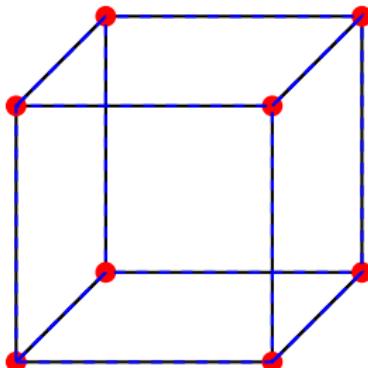
Enveloppe convexe

Problème si non convexe



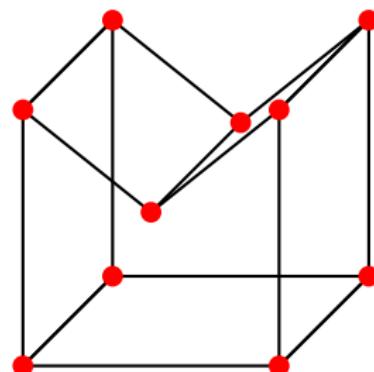
Points d'intérêts

Points d'intérêts sur un cube



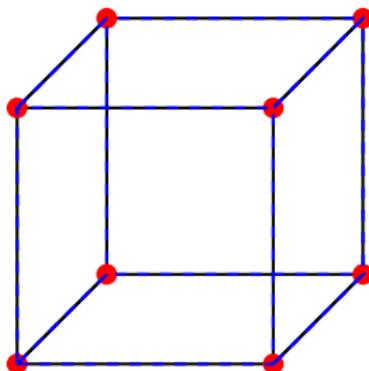
Enveloppe convexe

Problème si non convexe



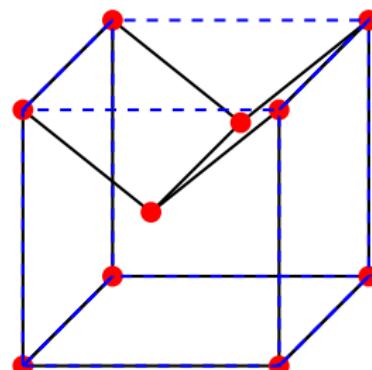
Points d'intérêts

Points d'intérêts sur un cube



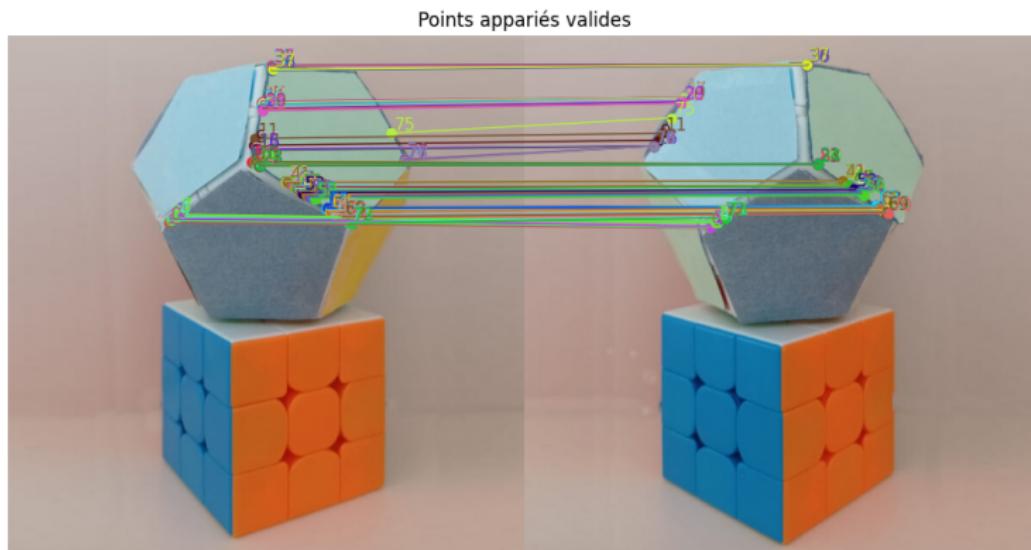
Enveloppe convexe

Problème si non convexe



Enveloppe convexe

Réultat : Filtrage epipolaire



Resultat : droite epipolaire + BRIEF lab

