



CSC-4MI04 – Reconnaissance d'images

Détection et Appariement de Points Caractéristiques

MENESES GAMBOA Carlos
VASQUEZ TORRES Jair

Table des matières

1	Résumé	2
2	Introduction générale	2
3	(2) Format d'images et Convolutions : Q1–Q3	2
3.1	Q1 — Convolution 2D : balayage direct vs <code>cv2.filter2D</code>	2
3.1.1	Objectif	2
3.1.2	Rappels théoriques	2
3.1.3	Implémentations	3
3.1.4	Protocole expérimental	3
3.1.5	Résultats numériques	3
3.1.6	Discussion	4
3.1.7	Figures (emplacements)	4
3.2	Q2 — Interprétation théorique du noyau de rehaussement	5
3.3	Q3 — Gradients I_x , I_y et norme du gradient	6
4	(3) Détecteurs : Q4–Q6	6
4.1	Q4 — Détecteur de Harris : compléments d'implémentation	6
4.2	Q5 — Harris : paramètres, multi-échelles et contrainte de distance	7
4.3	Q6 — Comparaison ORB vs KAZE (détection)	7
5	(4) Descripteurs et Appariement : Q7–Q9	7
5.1	Q7 — Principe des descripteurs ORB/KAZE et invariances	7
5.2	Q8 — Appariement : CrossCheck, RatioTest, FLANN	8
5.3	Q9 — Évaluation quantitative avec transformation connue	8
6	Conclusion	8

1 Résumé

TODO : rédiger un résumé synthétique du rapport, présentant les objectifs, les méthodes utilisées, les résultats clés et les conclusions principales. Le résumé doit être clair et concis, donnant au lecteur une vue d'ensemble rapide du contenu du rapport.

2 Introduction générale

TODO : présenter les objectifs du TP, les étapes prévues, et la structure du rapport. Expliquer brièvement les concepts clés (convolution, détecteurs, descripteurs, appariement) pour contextualiser les questions à venir.

3 (2) Format d'images et Convolutions : Q1–Q3

3.1 Q1 — Convolution 2D : balayage direct vs `cv2.filter2D`

3.1.1 Objectif

Comparer une implémentation directe (double boucle Python) et l'implémentation OpenCV `cv2.filter2D` sur une image en niveaux de gris, puis expliquer les écarts dus aux bords, au clipping et aux conventions d'affichage.

3.1.2 Rappels théoriques

Une image en niveaux de gris est une matrice $I(y, x)$, avec une dynamique typique $[0, 255]$ en `uint8`. La conversion en `float64` permet d'éviter les overflow et de conserver les valeurs négatives ou supérieures à 255.

Le filtrage linéaire spatial s'écrit :

$$I_f(x, y) = \sum_{i=-1}^1 \sum_{j=-1}^1 K(i, j) I(x + i, y + j)$$

Noyau de rehaussement utilisé :

$$K = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Expression locale :

$$\text{val} = 5I(x, y) - I(x - 1, y) - I(x + 1, y) - I(x, y - 1) - I(x, y + 1)$$

Interprétation : amplification des hautes fréquences (bords, détails), pouvant produire overshoot et undershoot.

3.1.3 Implémentations

Méthode directe. Balayage de la zone intérieure ($1..h-2$, $1..w-2$), puis saturation explicite :

$$\text{clip}(\text{val}) = \min(\max(\text{val}, 0), 255)$$

Méthode OpenCV. Dans le code final :

```
img3 = cv2.filter2D(img, -1, kernel, borderType=cv2.BORDER_REPLICATE)
```

La sortie `img3` est RAW en `float64` (pas de clipping). Pour comparaison équitable :

$$\text{img3_clip} = \text{clip}(\text{img3}, 0, 255)$$

Conventions d’affichage.

- OpenCV : les images flottantes sont interprétées dans $[0, 1]$, d’où l’usage de `img3/255.0`.
- Matplotlib : normalisation implicite possible ; pour comparer, fixer `vmin=0`, `vmax=255`.

3.1.4 Protocole expérimental

- Image : `FlowerGarden2.png`, niveaux de gris, $h = 240$, $w = 360$.
- Zone de comparaison : intérieur $[1:h-2, 1:w-2]$, pour éviter les divergences dues aux bords.
- Comparaison A : `img2` (clip) vs `img3` (RAW).
- Comparaison B : `img2` (clip) vs `img3_clip`.

3.1.5 Résultats numériques

Dimensions. 240×360 .

Temps d’exécution et accélération.

TABLE 1 – Q1 – Temps et accélération.

Méthode	Temps (s)	Facteur
Méthode directe	0.167926867	$1.00 \times$
<code>cv2.filter2D</code>	0.006553004	$25.63 \times$

$$S = \frac{0.167926867}{0.006553004} \approx 25.63$$

Sortie RAW (`filter2D`).

$$\min(\text{img3}) = -609.0, \quad \max(\text{img3}) = 877.0$$

Sur la zone intérieure (85204 pixels) :

- pixels < 0 : 13141,
- pixels > 255 : 8304,

— total hors plage : 21445 ($\approx 25.17\%$).

Différence intérieure : `img2 (clip)` vs `img3 (RAW)`.

$\text{max} = 622.0, \quad \text{mean} = 23.888702408337636, \quad \text{std} = 61.92829613986976$

Différence intérieure : `img2 (clip)` vs `img3_clip`.

$\text{max} = 0.0, \quad \text{mean} = 0.0, \quad \text{std} = 0.0$

3.1.6 Discussion

L'égalité parfaite entre `img2` et `img3_clip` confirme que les deux méthodes appliquent la même opération linéaire (même noyau) sur la zone intérieure. L'écart avec `img3 RAW` provient uniquement du clipping. Le maximum 622 est cohérent avec un dépassement positif : $877 - 255 = 622$.

La fonction `filter2D` est nettement plus rapide car implémentée en C/C++ optimisé, tandis que la double boucle Python subit un overhead interprété élevé. La comparaison sur l'intérieur évite de confondre ces effets avec le traitement des bords.

3.1.7 Figures (emplacements)



FIGURE 1 – Fig Q1.1 – Image originale.

FIGURE 2 – Fig Q1.2 – Résultat méthode directe (`img2`).FIGURE 3 – Fig Q1.3 – Résultat `filter2D` après clipping (`img3_clip`).

3.2 Q2 — Interprétation théorique du noyau de rehaussement

Rappel de l'énoncé : justifier théoriquement pourquoi le noyau utilisé agit comme un filtre de sharpening, et relier son effet aux hautes fréquences.

TODO – Éléments à compléter (sans résultats inventés)

— Exécuter/adapter `Convolutions.py` pour produire des exemples illustratifs.

- Expliquer le lien $\text{noyau} = \text{identité} + \text{terme de type Laplacien}$ (interprétation fréquentielle).
- Explorer des paramètres/variantes de noyaux pour comparer l'intensité du rehaussement.
- Produire des figures (image d'entrée, image filtrée, zoom sur contours).
- Définir les métriques à rapporter (variation de contraste local, histogrammes, énergie des gradients).
- Rédiger la discussion théorique (effets attendus, limites, artefacts).

Statut : section préparée, aucun résultat chiffré ajouté.

3.3 Q3 — Gradients I_x , I_y et norme du gradient

Rappel de l'énoncé : modifier l'approche pour calculer les gradients horizontal et vertical, puis la norme du gradient, avec précautions de type et d'affichage.

TODO – Éléments à compléter (sans résultats inventés)

- Modifier `Convolutions.py` pour calculer I_x , I_y et $\|\nabla I\|$.
- Définir les opérateurs/convolutions utilisés pour I_x et I_y .
- Vérifier les conventions d'affichage (valeurs négatives, normalisation, conversion de type).
- Produire les figures dédiées : I_x , I_y , norme, comparaison visuelle.
- Choisir les métriques à rapporter (min/max, distributions, sensibilité au bruit).
- Expliquer les points théoriques (direction du gradient, amplitude, interprétation contour).

Statut : section préparée, aucun résultat chiffré ajouté.

4 (3) Détecteurs : Q4–Q6

4.1 Q4 — Détecteur de Harris : compléments d'implémentation

Rappel de l'énoncé : compléter `Harris.py`, calculer la réponse Θ , utiliser une fenêtre W et extraire des maxima locaux par dilatation morphologique.

TODO – Éléments à compléter (sans résultats inventés)

- Exécuter et compléter `Harris.py` (calcul de Θ , fenêtre W).
- Implémenter/valider la détection de maxima locaux par dilatation morphologique.
- Explorer les paramètres principaux (taille de fenêtre, seuil, lissage).
- Produire les figures : carte de réponse et points détectés sur image.
- Définir les métriques à rapporter (nombre de points, stabilité visuelle, distribution spatiale).
- Expliquer le mécanisme Harris (matrice des moments, coin vs bord vs zone plate).

Statut : section préparée, aucun résultat chiffré ajouté.

4.2 Q5 — Harris : paramètres, multi-échelles et contrainte de distance

Rappel de l'énoncé : analyser l'effet des paramètres, du multi-échelles et imposer une contrainte de distance minimale r entre maxima.

TODO – Éléments à compléter (sans résultats inventés)

- Étendre `Harris.py` pour explorer taille de fenêtre, α , seuils.
- Ajouter l'analyse multi-échelles (pyramide ou variation d'échelle).
- Implémenter la sélection de maxima avec contrainte distance $\geq r$.
- Produire les figures comparatives par réglage/échelle.
- Définir les métriques (densité de points, répétabilité qualitative, couverture spatiale).
- Discuter compromis robustesse/localisation et sensibilité aux paramètres.

Statut : section préparée, aucun résultat chiffré ajouté.

4.3 Q6 — Comparaison ORB vs KAZE (détection)

Rappel de l'énoncé : comparer ORB et KAZE en détection de points d'intérêt, avec paramètres explicites et comparaison visuelle sur une paire d'images.

TODO – Éléments à compléter (sans résultats inventés)

- Exécuter `Features_Detect.py` pour ORB et KAZE.
- Documenter les paramètres utilisés pour chaque détecteur.
- Produire les figures sur une paire d'images (points détectés superposés).
- Définir les métriques à rapporter (nombre de points, répartition, robustesse visuelle).
- Vérifier la répétabilité qualitative sous changements de vue/éclairage.
- Expliquer les différences de comportement ORB vs KAZE.

Statut : section préparée, aucun résultat chiffré ajouté.

5 (4) Descripteurs et Appariement : Q7–Q9

5.1 Q7 — Principe des descripteurs ORB/KAZE et invariances

Rappel de l'énoncé : présenter le principe des descripteurs ORB/KAZE et distinguer invariances du détecteur et du descripteur (échelle, rotation).

TODO – Éléments à compléter (sans résultats inventés)

- Décrire le pipeline détecteur + descripteur pour ORB et KAZE.
- Préciser les invariances (échelle/rotation) et à quel niveau elles interviennent.
- Préparer des illustrations locales (patches/points correspondants) à extraire.

- Lister les scripts/fonctions à exécuter (`Features_Detect.py` et scripts de description).
- Définir les critères d'analyse (discriminabilité, robustesse qualitative).
- Ajouter une discussion théorique claire détecteur vs descripteur.

Statut : section préparée, aucun résultat chiffré ajouté.

5.2 Q8 — Appariement : CrossCheck, RatioTest, FLANN

Rappel de l'énoncé : comparer plusieurs stratégies d'appariement et expliquer les différences de distances selon le type de descripteur.

TODO – Éléments à compléter (sans résultats inventés)

- Exécuter les scripts d'appariement (CrossCheck, RatioTest, FLANN).
- Documenter les paramètres de matching (seuil ratio, nombre de voisins, index FLANN).
- Produire les figures de correspondances pour chaque stratégie.
- Définir les métriques à rapporter (nombre de matches, taux d'outliers, qualité visuelle).
- Expliquer pourquoi les distances diffèrent selon descripteurs binaires vs flottants.
- Discuter compromis précision/robustesse/coût de calcul.

Statut : section préparée, aucun résultat chiffré ajouté.

5.3 Q9 — Évaluation quantitative avec transformation connue

Rappel de l'énoncé : mettre en place une évaluation quantitative à partir d'une transformation géométrique connue (par ex. `cv2.warpAffine`) pour mesurer précision/rappel des appariements.

TODO – Éléments à compléter (sans résultats inventés)

- Générer des paires image-transformée avec transformation connue (`cv2.warpAffine`).
- Définir un protocole d'évaluation des correspondances (vrai/faux match via géométrie).
- Calculer les métriques cibles (précision, rappel, éventuellement courbes).
- Comparer ORB/KAZE et les stratégies de matching dans le même protocole.
- Produire les figures de synthèse (matches corrects/incorrects, visualisation d'erreurs).
- Discuter les limites du protocole et les biais possibles.

Statut : section préparée, aucun résultat chiffré ajouté.

6 Conclusion

La Q1 est finalisée avec une comparaison complète, reproductible et chiffrée entre balayage direct et `cv2.filter2D`. Les Q2 à Q9 sont structurées selon l'énoncé avec des checklists opérationnelles pour la suite du TP, sans ajout de résultats non vérifiés.