Takfarinas Kebci

Yann Nebout

Traitement d'image et code

Table des matières

[I. Bibliothèques pour réaliser le traitement d’image 3](#_Toc509569360)

[II. Théorie du traitement d’image 4](#_Toc509569361)

[A. Caractéristique d'une image 4](#_Toc509569362)

[B. Les histogrammes 4](#_Toc509569363)

[C. Filtrage d'une image 5](#_Toc509569364)

[1. Rappel mathématiques de la convolution 5](#_Toc509569365)

[2. Filtres 6](#_Toc509569366)

[D. Détection des contours 6](#_Toc509569367)

[1. Gradient 6](#_Toc509569368)

[2. Laplace 7](#_Toc509569369)

[3. Canny 8](#_Toc509569370)

[E. Points d'intérêts 8](#_Toc509569371)

[1. Détecteur de points d’intérêts 8](#_Toc509569372)

[2. Mise en correspondance 10](#_Toc509569373)

[F. Algorithme SIFT 11](#_Toc509569374)

[III. Explication du code 12](#_Toc509569375)

[A. Header (.hpp) 12](#_Toc509569376)

[1. util.hpp 12](#_Toc509569377)

[2. TraitementImage.hpp 12](#_Toc509569378)

[B. Classe créée 12](#_Toc509569379)

[1. Fonction 12](#_Toc509569380)

[2. Points 12](#_Toc509569381)

[3. Matrice 13](#_Toc509569382)

[4. Image 13](#_Toc509569383)

[5. Filtre 13](#_Toc509569384)

[6. FiltreMoyenneur 14](#_Toc509569385)

[7. FiltreMedian 14](#_Toc509569386)

[8. FiltreGauss 14](#_Toc509569387)

[9. Laplace 14](#_Toc509569388)

[10. Gradient 14](#_Toc509569389)

[11. FiltreCanny 15](#_Toc509569390)

[12. Moravec 15](#_Toc509569391)

[13. Harris 15](#_Toc509569392)

[14. PointInteret 15](#_Toc509569393)

[IV. Table des équations 16](#_Toc509569394)

[V. Bibliographie 17](#_Toc509569395)

# Bibliothèques pour réaliser le traitement d’image

Afin de faire du traitement d’image il est possible d’utiliser différentes bibliothèque. Le plus important étant de récupérer l’image ou le flux vidéo.

La bibliothèque SFML permet de récupérer les données d’une image mais ne permet ni de faire du traitement ni de récupérer un flux vidéo. Cette dernière pourra être utilisée pour créer une interface graphique.

La bibliothèque SDL permet de réaliser les mêmes opérations que SFML.

Ogre3D est une bibliothèque de type SFML et SDL mais permet de traiter la 3D mais tout comme ses sœurs 2D elle ne permet pas de lire les flux vidéo d’une webcam ou autre caméra.

Opencv permet quant à elle de récupérer les données des images et de les afficher. Elle permet aussi d’utiliser un maximum d’outils de traitement d’image, que ce soit la reconnaissance de forme ou le tracking d’objet. Par ailleurs, elle permet de récupérer les données vidéos d’une caméra et Opencv les transformes en une donnée avec laquelle on peut réaliser du traitement de l’image.

Opengl est une bibliothèque permettant de traiter des images 3D et de capturer un flux vidéo.

Les bibliothèques de réalité tel que ARToolKit, Vuforia réalise le traitement d’image sur les marqueurs et sont couplés à des bibliothèques tels qu’Opengl ou Opencv. D’autres bibliothèques comme ARCore de Google permettent de détecter des objets et surface.

Afin d’avoir une bonne compréhension des outils de traitement d’image et devant ces bibliothèques de RA récentes, nous avons décidé de réalisé un programme en c++ avec l’utilisation d’Opencv pour récupérer les images et les vidéos. Cela nous permettra de réaliser un programme pouvant détecter des marqueurs et des objets.

Dans la suite de ce document, nous expliquerons ce qu’est une image et une vidéo, comment on peut traiter une image, détecter les contours et les points d’intérêts afin de pouvoir reconnaître des objets. Par la suite, nous expliquerons comment le code est construit.

Notre programme est codé en c++ car Opencv ne peut être utilisé qu’avec du C++ ou du Python. Nous avons créé différentes classes (Image, Filtre …) sous forme de .cpp et .hpp ainsi que des Headers(.hpp) contenant de simple fonction.

# Théorie du traitement d’image

## Caractéristique d'une image

Une image est en réalité un échantillonnage spatial qui quantifie l'intensité lumineuse.Toute image est caractérisée par une hauteur et une largeur (notée nx et ny). (ensta)

L'intensité lumineuse peut être définie par une échelle de variation de gris (nuance de gris), de couleur(RGB), ou en HSV (teinte : 0->2π, saturation : 0->1, intensité lumineuse 0->1).

Il existe des calculs permettant de passer d'une échelle à une autre :

RGB(Ir,Ib,Ig)->nuance de gris(Indg) :

Équation 1: transformation image RGB en image en nuance de gris

RGB(Ir,Ib,Ig)->HSV(Ih,Is,Iv) :

Équation 2 : transformation image RGB en image HSV

## Les histogrammes

Les histogrammes permettent de caractériser une image.

Un histogramme correspond à un graphique de la mesure du nombre de pixels pour chaque niveau de couleur.

Il est possible de réaliser un histogramme cumulé qui correspond au graphique qui mesure le nombre de pixel ayant un valeur de niveau de gris qui soit inférieur ou égale à une certaine valeur.

Chacun de ces histogrammes peut être utilisé pour réaliser des filtrages ou des rehaussements d’intensité de chaque pixel, par exemple.

## Filtrage d'une image

### Rappel mathématiques de la convolution

De manière générale, le produit de convolution s’écrit :

Équation 3: Produit de convolution

Il a pour propriété :

Équation 4: Propriété du produit de convolution

Pour une image :

Équation 5: Produit de convolution discret pour une image

Où I représente l’image qui sera traitée.

### Filtres

Le principe du filtrage permet de réduire ou d'ajouter du flou (n : taille de l’image)

(Philippeau)(Philippeau) (Bonnin)

|  |  |  |  |
| --- | --- | --- | --- |
| Nom | Formule | Avantages | Inconvénients |
| Moyenneur | X | - Facile à coder  - Rapide  - Plus n est grand plus le bruit est éliminé | - Peu robuste  - Plus n est grand plus les contours  - Plus n est grand plus les détails fins disparaissent |
| Médian | X | - Très efficace pour enlever le bruit impulsionnel (poivre et sel) | - Supprime les détails  - Enlève les points isolés non désiré  - Très coûteux en temps des calculs |
| Gaussien | Équation 6: Calcul de la valeur des cases du masque gaussien | - Réglage simple  - S’adapte au problème | - Complexité du calcul |



Attention le masque gaussien créé doit être normalisé en faisant la somme des éléments du masque et divisé chaque élément par cette somme.

## Détection des contours

Le principe de détection de contour est de détecter les discontinuités de nuances de couleur dans une image. En effet, un contour correspond à une forte discontinuité dans l'image. Il s’agit de filtre particulier.

### Gradient

Le contour correspond à une pente élevé de transition (maxima locaux).

2 paramètres : le module et l'orientation.

Basé sur le principe de dérivé première selon chacun des axes.

Équation 7: Calcul du gradient d'une image

Où dx et dy sont des masques.

Pour afficher les contours il faut calculer le module du gradient (Noté G), on peut aussi utiliser l'orientation (Noté Φ) :

Équation 8: Norme du gradient

Équation 9: Phase du gradient

Deux masques sont généralement utilisés :

Équation 10: Noyau de Prewitt

,

Équation 11: Noyau de Sobel

,

### Laplace

Le filtre de Laplace est basé sur la dérivée seconde et la matrice Hessienne.

La matrice Hessienne (Noté H) est une matrice regroupant les dérivées secondes d’un pixel.

Équation 12: Matrice Hessienne

Le Laplacien correspond à la trace de la matrice Hessienne (Formule F13) :

Équation 13: Laplacien de l'image

Généralement, on utilise deux masques :

Équation 14: Noyaux Laplacien

### Canny

La première étape est de réduire le bruit par un filtre gaussien.

La seconde étape est de détecter les contours par le gradient.

La troisième étape consiste à supprimer les non maxima. Si la valeur du gradient du pixel est inférieure à celle d’un des deux voisins alors cette dernière est mise à zéro.

La quatrième étape consiste à réaliser un seuillage par hystérésis. Pour cela, on définit deux seuils : - un seuil haut qui dit que si la valeur du gradient est supérieure à ce seuil alors le pixel appartient au contour.

- un seuil bas qui dit que si la valeur du gradient est inférieure à ce seuil alors le pixel n’appartient pas au contour.

- dans le cas où la valeur du gradient est comprise entre ces deux seuils alors on test la valeur des huit pixels qui l’entoure et si un de ces pixels appartient au contour alors le pixel appartient au contour.

## Points d'intérêts

### Détecteur de points d’intérêts

#### Détecteur de Moravec

Le détecteur de Moravec permet de détecter les coins dans une image en analysant les variations entre 2 pixels.

Plus exactement, il faut faire varier la position de la fenêtre en x et y et calculer la valeur de la dissemblance (noté E).

Le calcul de la dissemblance correspond à la somme de la valeur absolue de la différence au carré du masque décalé avec le masque d’origine :

Équation 15: Dissemblance de Moravec

Où F correspond au masque

Si la dissemblance est nettement supérieure à 0 on considère que le pixel est un coin sinon il s'agit d'un contour ou d'une zone sans variation. Néanmoins, différents problèmes sont présents :

Tout d'abord, pour faire les calculs on utilise une fenêtre qui est binaire et peut apporter du bruit.

Ensuite, seuls les décalages à 45° sont considérés

Enfin, sa sensibilité aux contours est importante car seule le minimum de E est considéré.

#### Détecteur de Harris

Le détecteur de Harris permet de résoudre en grande partie des problèmes du filtre de Moravec :

La fenêtre n'est pas binaire mais gaussienne.

Un développement de Taylor est utilisé.

Utilisation des valeurs propre de la matrice hessienne de l'image.

L’objectif de ce filtre est de réaliser un filtrage Laplacien d’une gaussienne (aussi appelé LoG).

La matrice Hessienne (ici notée M) est la suivante :

Équation 16: Matrice de Harris d'une gaussienne

Où <x> représente le filtrage de x par un filtre gaussien.

Le calcul de la dissemblance est donc le suivant :

Équation 17: Dissemblance de Harris

Nous savons que si la valeur de E est grande alors il s’agit d’un coin. D’après Harris :

- Si λ1≈0 et λ2≈0 le pixel est dans une zone homogène

- Si λ1≈0 et λ2 grande (ou réciproquement) le pixel appartient au contour

- Si λ1 et λ2 sont grande le pixel est un coin

Donc Harris calcule la valeur H de la manière suivante :

Équation 18: Paramètre de Harris

Où det(M) est le déterminant de M, Tr(M) est la trace de M et c une constante généralement égale à 0,05.

Donc H est grand seulement près d’un coin.

### Mise en correspondance

Une fois que les points d'intérêts dans l'image sélectionnée on les compare à une base de référence. Plusieurs comparaisons sont possibles :

u et v sont un ensemble de pixel selon les deux directions, et I1 et I2 sont les images de référence et testée, I1 et I2 sont les moyennes des masques.

La somme des différences au carrées ou SSD (Sum of Squared Difference) :

- Il s’agit de calculer le carré de la distance euclidienne :

Équation 19: Somme des différences au carrées

Plus la valeur est grande plus les points sont différents. Si la valeur est 0 alors les points sont identiques

- Il existe une variante appelée SSD centré ou ZSSD (Zero mean Sum of Squared Difference) :

Équation 20: Somme des différences au carrées centrée

La somme de la valeur absolue des différences ou SAD (Sum of Absolute Difference) :

Elle est l’une des plus utilisées.

Équation 21: Somme de la valeur absolue des différences

- Il existe une variante appelée SAD centré ou ZSAD (Zero mean Sum of Absolute Difference) :

Équation 22: Somme de la valeur absolue des différences centrée

La Corrélation croisée normalisée ou NCC (Normalized Cross Correlation) :

Équation 23: Corrélation croisée normalisée

- Il existe une variante appelée NCC centré ou ZNCC (Zero mean Normalized Cross Correlation) :

Équation 24: Corrélation croisée normalisée centrée

La valeur se situe dans l’intervalle [-1, 1] et est d’autant plus grande que les deux points sont corrélés. Cette méthode est plus robuste au changement de luminosité.

## Algorithme SIFT

**Suite au Prochaine Épisode ...**

(Poublang)

# Explication du code

Au départ, nous avons voulu, tout simplement, utiliser la bibliothèque Opencv pour le traitement d’image. Mais, nous nous sommes confrontés à la documentation qui n’expliquait pas le fonctionnement exacte des outils que nous devions utilisés. Afin de palier à ce problème de compréhension des fonctions nous avons décidé de créer nos propres fonctions/Classes à partir de cours/thèses/sites trouvés sur internet pour comprendre la totalité du fonctionnement du traitement d’image. Opencv ne sera utilisé que pour récupérer le flux vidéo et le transformer en image

## Header (.hpp)

### util.hpp

Ce fichier contient toutes les bibliothèques de base du langage c++ et de la librairie opensource Opencv.

Les variables globales définies par des « #define » sont aussi présents dans ce fichier.

### TraitementImage.hpp

Ce fichier contient toute la totalité des headers créés. Ils correspondent aux fichiers contenant la déclaration des fonctions et classes créées. Cela permet de ne d’écrire qu’un seul « include » contenant toutes les bibliothèques utiles et créées.

## Classe créée

### Fonction

Dans ce fichier se trouve le calcul de convolution d’une image par un masque selon l’équation 3.

Il contient aussi une fonction qui copie les valeurs d’un tableau source dans un tableau de destination et une fonction qui tri les valeurs d’un tableau.

### Points

La classe Points contient seulement les coordonnées du point et permet de lire et d’écrire ces coordonnées.

### Matrice

La classe Matrice est constituée des dimensions d’un tableau à deux dimensions contenant les valeurs de chaque case de ma matrice. Elle contient aussi deux variables correspondant à la trace et au déterminant de la matrice si celle-ci est carrée. Enfin, une variable définie si la matrice est carrée.

Dans cette classe, il y a la fonction *estCarre()* qui vérifie si la matrice est carrée ou non pour savoir si on peut calculer la trace et le déterminant de la matrice. Si la matrice est carrée alors la fonction *calculTrace()* réalisera le calcul de la trace et la fonction *calculDeterminant()* réalise le calcul du déterminant d’une matrice 2x2 sinon elle appelle la fonction *copieMatrice()* qui découpera la matrice jusqu’à ce que le calcul du déterminant soit celui d’une matrice 2x2.

**Une fonction de produit matricielle sera peut être créée dans le prochaine épisode.**

### Image

La classe Image est constituée des dimensions de l’image, de deux tableaux à respectivement 3 et 2 dimensions contenant l’image en couleur et en nuance de gris et d’un tableau à deux dimensions correspondant à l’histogramme d’une couleur de l’image choisie.

Cette classe récupérera les informations d’une image par l’utilisation de la classe Mat de la librairie Opencv. Elle peut calculer la valeur de chaque pixel d’une image couleur pour donner la valeur de la nuance de gris selon la Formule F1. Cette valeur sera affecterer au pixel correspondant dans le tableau correspondant à l’image en nuance de gris par la fonction *RGBtoGRAY()*. Les fonctions *recupeRGBvalue()* et *recupeGRAYvalue()* permettent de récupérer les valeurs des pixels d’une image de type Mat. La fonction *histo()* calcule l’histogramme d’une couleur de l’image.

### Filtre

La classe Filtre est une classe qui décrit la base de tous les filtres. Elle est constituée du nom du filtre, de l’image initiale et de l’image filtrée (les 2 en nuance de gris), de la taille du noyau de convolution et du masque extrait de l’image qui est un nombre impair, du noyau du filtre (si nécessaire) et du masque correspondant à un morceau de l’image.

Cette classe définie la méthode d’extraction du morceau de l’image qui sera convolué. La méthode a été de découper en trois fonctions distinctes :

*extractionMask()*: qui appellera les deux autres fonctions si le pixel central du masque n’est pas à l’emplacement de coordonnée (n-1)/2,(m-1)/2 au minimum et n-(n-1)/2,m-(m-1)/2 au maximum si la taille est de taille n, m.

*coin()*: cette fonction est utilisée si et seulement si au moins un pixel du masque selon la hauteur et un pixel du masque selon la largeur se trouve à l’extérieur de l’image.

*bord()*: cette fonction est utilisée si et seulement si au moins un pixel du masque selon la hauteur ou un pixel du masque selon la largeur se trouve à l’extérieur de l’image.

### FiltreMoyenneur

Cette classe hérite de la classe filtre. Le seul paramètre supplémentaire de cette classe est la variable moyenne qui correspond à 1/(n\*n) où n est la taille du noyau.

La fonction *noyauMoyen()* calcule le noyau moyen. La fonction *filtrage()* réalise le filtrage en extrayant les morceaux de l’image en les convoluant avec le noyau moyen.

### FiltreMedian

Cette classe hérite de la classe filtre.La fonction *filtreMedian()* réalise le filtrage en extrayant les morceaux de l’image. Ces morceaux seront ensuite copiés et les valeurs triées. Enfin la fonction *calculMedianne()* trouvera la valeur médiane.

### FiltreGauss

Cette classe contient l’écart type, σ, d’une gaussienne.

La fonction *ecartType()* permet de vérifié que l’écart type de la gaussienne n’est pas égale à zéro. La fonction *noyauGauss()* calcule le noyau gaussien avec l’équation 6. La fonction *filtrage()* permet de réaliser le filtrage gaussien.

### Laplace

La classe contient une valeur seuil qui permet de binariser l’image pour afficher les contours.

La fonction *choixNoyau()* permet de choisir le noyau laplacien entre les deux cités dans la partie précédente (équation 14). Elle appelle les fonctions *noyauLaplace1()* et *noyauLaplace2()*. La fonction *filtrage()* réalise le filtrage laplacien. La fonction *seuillage()* réalise l’opération de binarisation.

### Gradient

La classe Gradient contient un noyau supplémentaire car la détection de contour par le gradient nécessite un masque orienté selon la hauteur et un autre selon la largeur, d’un seuil pour réaliser la binarisation et d’un tableau à deux dimensions contenant la valeur des arguments de chaque gradient.

La fonction *initArg()* permet que le tableau d’argument soit de la même taille que l’image. La fonction *choixNoyau()* permet de choisir entre les noyaux de Sobel (équation 11) et les noyaux de Prewitt (équation 10). La fonction *seuillage()* permet de binariser l’image résultante. La fonction *filtreGradient()* permet de calculer le gradient (équation 8) et l’argument du gradient (équation 9).

### FiltreCanny

La classe est constituée d’un filtre gaussien, d’une image qui sera le résultat du filtre gaussien, une image qui sera issue d’un filtre gradient, un tableau d’arguments, un écart type pour le filtre gaussien et deux seuils, un seuil haut et un seuil bas, pour le seuillage par hystérésis.

La fonction *canny()* réalise le filtrage, la fonction *nonMaxima()* permet d’éliminer les pixels non maxima, la fonction *hystérésis()* réalise l’opération de seuillage par hystérésis expliquée dans la partie précédente. Les fonctions *normArg(), comp0/45/90/135()* permettent de ramener les arguments de chaque gradient à un multiple de 45° positif entre 0° et 135°. La fonction *comparaisonNorme()* permet de comparer les normes du gradient d’un pixel avec ses deux voisins dans sa direction. Enfin, les fonctions *hysteresisCoin/Bord/Autre()* permettent de réaliser l’hystérésis en fonction de la position du pixel.

### Moravec

La classe contient un seuil pour déterminer si le pixel étudié est un pixel d’intérêt, un image, un masque et un tableau de pixel d’intérêts.

La fonction *extractionMask()* permet d’extraire un morceau de l’image. La fonction *calculE()* permet de réaliser le calcul de E selon l’équation 15. La fonction *detection()* permet de réaliser la détection de points d’intérêts et la fonction *estInterressant()* permet de dire si le point est un point d’intérêt.

### Harris

La classe contient les mêmes paramètres que Moravec. En plus de ces paramètres, la classe contient une matrice hessienne et un paramètre c.

Les fonctions *extractionMask(), detection()* et *estInterressant()* sont identiques que celles de Moravec. La fonction *calculH()* réalise le calcul de l’équation 18. La fonction *hessienne()*, quant à elle, permet de calculer la matrice hessienne de chaque pixel.

### PointInteret

La classe contient deux images (l’image de référence et l’image étudiée), deux masques (un pour chaque image), deux tableaux de points d’intérêts (un pour chaque image) et un tableau contenant les points appariés deux à deux.

La fonction *assigneMasque()* permet d’assigner chaque masque à son image pour un extraire un morceau, la fonction *moyenne()* calcul la moyenne de chaque masque, la fonction *choixType()* permet de choisir le type de corrélation pour apparier les points de chaque image, appariement() réalise l’appariement en appelant la fonction *correlation()*. Les fonctions *norme(), SSD(), ZSSD(), SAD(), ZSAD(), NCC(), ZNNC()* et *somme()* permettent de réaliser les différents moyens de corréler à partir des équations 19 à 24.

# Table des équations

[Équation 1: transformation image RGB en image en nuance de gris 4](#_Toc509568451)

[Équation 2 : transformation image RGB en image HSV 4](#_Toc509568452)

[Équation 3: Produit de convolution 5](#_Toc509568453)

[Équation 4: Propriété du produit de convolution 5](#_Toc509568454)

[Équation 5: Produit de convolution discret pour une image 5](#_Toc509568455)

[Équation 6: Calcul de la valeur des cases du masque gaussien 6](#_Toc509568456)

[Équation 7: Calcul du gradient d'une image 6](#_Toc509568457)

[Équation 8: Norme du gradient 7](#_Toc509568458)

[Équation 9: Phase du gradient 7](#_Toc509568459)

[Équation 10: Noyau de Prewitt 7](#_Toc509568460)

[Équation 11: Noyau de Sobel 7](#_Toc509568461)

[Équation 12: Matrice Hessienne 7](#_Toc509568462)

[Équation 13: Laplacien de l'image 7](#_Toc509568463)

[Équation 14: Noyaux Laplacien 8](#_Toc509568464)

[Équation 15: Dissemblance de Moravec 8](#_Toc509568465)

[Équation 16: Matrice Hessienne d'une gaussienne 9](#_Toc509568466)

[Équation 17: Dissemblance de Harris 9](#_Toc509568467)

[Équation 18: Paramètre de Harris 9](#_Toc509568468)

[Équation 19: Somme des différences au carrées 10](#_Toc509568469)

[Équation 20: Somme des différences au carrées centrée 10](#_Toc509568470)

[Équation 21: Somme de la valeur absolue des différences 10](#_Toc509568471)

[Équation 22: Somme de la valeur absolue des différences centrée 10](#_Toc509568472)

[Équation 23: Corrélation croisée normalisée 11](#_Toc509568473)

[Équation 24: Corrélation croisée normalisée centrée 11](#_Toc509568474)

# Bibliographie

**Bonnin Patrick** http://patrick-bonnin.developpez.com/cours/vision/apprendre-bases-traitement-image/partie-1-introduction/ [En ligne].

**ensta** http://perso.ensta-paristech.fr/~manzaner/Support\_Cours.html [En ligne].

http://patrick-bonnin.developpez.com/cours/vision/apprendre-bases-traitement-image/partie-1-introduction/ [En ligne].

http://perso.ensta-paristech.fr/~manzaner/Support\_Cours.html [En ligne].

**Philippeau Xavier** http://xphilipp.developpez.com/articles/filtres/ [En ligne].

**Philippeau Xavier** http://xphilipp.developpez.com/articles/segmentation/regions/ [En ligne].

**Poublang Philippe** https://sites.google.com/site/poublangsift/l-algorithme-des-sift [En ligne].

Cours PDF disponible sur le drive d’astek.