segment Manuel de référence

Généré par Doxygen 1.3.7

Thu Jul 1 23 :13 :48 2004

Table des matières

Chapitre 1

segment Index hiérarchique

1.1 segment Hiérarchie des classes

Cette liste d'héritage est classée approximativement par ordre alphabétique :

Chapitre 2

segment Index des classes

2.1 segment Liste des classes

Liste des classes, structures, unions et interfaces avec une brève description :

Chapitre 3

segment Index des fichiers

3.1 segment Liste des fichiers

Liste de tous les fichiers documentés avec une brève description :

Chapitre 4

segment Documentation des classes

4.1 Référence de la classe Channel

Représente un canal de sortie.

`#include <Debug.hpp>`

Fonctions membres publiques

- **Channel** (ostream ∗o=&cout)
  Contructeur avec un flot de sortie comme paramètre par défaut cout.
- void **reset** (ostream ∗o)
  Permet de changer de flot.

Attributs privés

- ostream ∗ **traceFile**
  Flot de sortie.

4.1.1 Description détaillée

Représente un canal de sortie.

Les informations de debug sont dirigée dans un canal : fichier, cout, cerr,... Les canaux sont déclarées au niveau globale.

Définition à la ligne 26 du fichier Debug.hpp.

4.1.2 Documentation des contructeurs et destructeur

4.1.2.1 Channel : :Channel (ostream ∗ *o* = `&cout`) `[inline]`

Contructeur avec un flot de sortie comme paramètre par défaut cout.

Paramètres :
o flot de srotie

Définition à la ligne 33 du fichier Debug.hpp.

Références traceFile.

La documentation de cette classe a été générée à partir du fichier suivant :
– **Debug.hpp**

4.2 Référence de la classe Color

Interface sur une classe de couleur concrète.

`#include <Color.hpp>`

Fonctions membres publiques

- **Color** ()
  Constructeur par défaut, couleur de type : COLOR_EMPTY.
- **Color** (float)
  Constructeur pour une couleur de type : COLOR_GRAY.
- **Color** (float, float, float, bool)
  Constructeur pour une couleur de type : COLOR_COLOR.
- **Color** (const **Color** &)
  Constructeur de copie.
- **Color** & **operator=** (const **Color** &)
  Opérateur d'affectation.
- ~**Color** ()
  Desctructeur.
- **ECOLOR_TYPE getType** () const
  Retourne le type de la couleur.
- string **toString** () const
  Retourne une chaîne de caractères détaillée.
- float **getGray** () const
  Retourne le niveau de gris [0 :1].
- float **getRed** () const
  Retourne la composante rouge [0 :1].
- float **getGreen** () const
  Retourne la composante verte [0 :1].
- float **getBlue** () const
  Retourne la composante bleue [0 :1].
- float **getHue** () const
  Retourne la teinte [0 :1].
- float **getSaturation** () const
  Retourne la saturation [0 :1].
- float **getValue** () const
  Retourne la valeur [0 :1].
- float * **getFloat** (bool) const
  Retourne un table des composantes.

– float **distance** (**Color**) const
  Retourne la distance d'une couleur à une autre.

– bool **operator==** (const **Color**) const
  Opérateur relationnel d'égalité.

– bool **operator<** (const **Color**) const
  Opérateur relationnel d'ordre.

Fonctions membres publiques statiques

– **Color getColor** (int)
  Retourne une couleur prédéfinie.

– **Color getAverageColor** (**Color**, int, **Color**, int)
  Retourne la moyenne pondérée de 2 couleurs.

– bool **inWindow** (float ∗, float, float ∗)
  Teste l'appartenance à une sphère de couleur.

Attributs privés

– **Color_Base ∗ p**
  Pointeur sur une couleur.

4.2.1 Description détaillée

Interface sur une classe de couleur concrète.

Gère un compteur de références La pluspart des méthodes sont redirigées vers une instance héritées de **Color_Base**(p. 12)

Définition à la ligne 33 du fichier Color.hpp.

4.2.2 Documentation des fonctions membres

4.2.2.1 Color Color : :getAverageColor (Color, int, Color, int) `[static]`

Retourne la moyenne pondérée de 2 couleurs.

Paramètres :

Color première couleur

int poids de la première couleur

Color seconde couleur

vint poids de la seconde couleur

Définition à la ligne 108 du fichier Color.cpp.

Références getAverageColor(), getHue(), getSaturation(), et getValue().

Référencé par getAverageColor().

4.2.2.2 Color Color : :getColor (int *i*) `[static]`

Retourne une couleur prédéfinie.

On a une palette de couleurs prédéfinie et on retourne cycliquement les couleurs quelles contient. Les valeur de la palettes sont tirées de la colormap "cool" de Matlab

Définition à la ligne 86 du fichier Color.cpp.

Références getColor().

Référencé par PNG : :getColor(), getColor(), Histogram : :makeClass(), Segment : :makeDisjointSet(), et MInt : :makeDisjointSet().

4.2.2.3 bool Color : :inWindow (float ∗, float, float ∗) `[static]`

Teste l'appartenance à une sphère de couleur.

Paramètres :

float∗ valeur hsv du centre

float rayon de la sphère

float∗ valeur hsv de couleur à tester

Définition à la ligne 120 du fichier Color.cpp.

Références inWindow().

Référencé par inWindow().

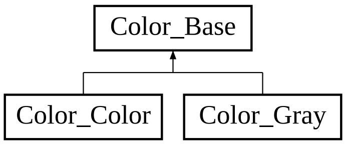
La documentation de cette classe a été générée à partir des fichiers suivants :
- **Color.hpp**
- **Color.cpp**

4.3 Référence de la classe Color_Base

Classe de base abstraite pour les 2 type de couleurs : gris et couleurs.

`#include <Color.hpp>`

Graphe d'héritage de Color_Base : :



Fonctions membres publiques

– **ECOLOR_TYPE getType** () const
  Retourne le type de la couleur.

Fonctions membres protégées

– **Color_Base** (**ECOLOR_TYPE** _type)
  Construit une couleur avec son type et initialise le compteur de références.

– virtual string **toString** () const =0
  Retourne une chaîne de caractères détaillés pour le debug.

– virtual float **getGray** () const =0
  Retourne le niveau de gris [0 :1].

– virtual float **getRed** () const =0
  Retourne la composante rouge [0 :1].

– virtual float **getGreen** () const =0
  Retourne la composante verte [0 :1].

– virtual float **getBlue** () const =0
  Retourne la composante bleue [0 :1].

– virtual float **getHue** () const =0
  Retourne la teinte [0 :1].

– virtual float **getSaturation** () const =0
  Retourne la saturation [0 :1].

– virtual float **getValue** () const =0
  Retourne la valeur [0 :1].

– virtual float **distance** (const **Color_Base** ∗) const =0
  Retourne la distance entre 2 couleurs.

– virtual bool **operator==** (const **Color**) const =0
  Opérateur relationnel d'égalité de couleur.

- virtual bool **operator**< (const **Color**) const =0
  Opérateur relationnel d'ordre sur les couleurs.

Attributs protégés

- **ECOLOR_TYPE type**
  Type.

Attributs privés

- int **use**
  Compteur de références.

4.3.1 Description détaillée

Classe de base abstraite pour les 2 type de couleurs : gris et couleurs.

C'est cette classe qui possède le compteur de référence

Définition à la ligne 94 du fichier Color.hpp.

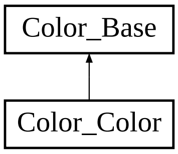
La documentation de cette classe a été générée à partir du fichier suivant :
- **Color.hpp**

4.4 Référence de la classe Color_Color

Classe pour les couleurs.

```
#include <Color.hpp>
```

Graphe d'héritage de Color_Color : :



Fonctions membres privées

- void **_computeHSV** ()
  Fonction de conversion RGB->HSV.
- void **_computeRGB** ()
  Fonction de conversion HSV->RGB.
- **Color_Color** (float, float, float, bool)
  Construit une couleur à partir de 3 valeurs RGB ou 3 valeus HSV (dépend du paramètre booléen).
- string **toString** () const
  Retourne une chaîne de caractères détaillés pour le debug.
- float **getGray** () const
  Retourne le niveau de gris [0 :1] : on fait simplement la moyenne des 3 valeurs.
- float **getRed** () const
  Retourne la composante rouge [0 :1].
- float **getGreen** () const
  Retourne la composante verte [0 :1].
- float **getBlue** () const
  Retourne la composante bleue [0 :1].
- float **getHue** () const
  Retourne la teinte.
- float **getSaturation** () const
  Retourne la saturation.
- float **getValue** () const
  Retourne la valeur.
- float **distance** (const **Color_Base** ∗) const
  Distance entre 2 couleurs.
- bool **operator==** (const **Color**) const
  Opérateur relationnel d'égalité.
- bool **operator<** (const **Color**) const
  Opérateur relationnel d'ordre.

Attributs privés

- float **red**
  Composante rouge.
- float **green**
  Composante verte.
- float **blue**
  Composante bleue.
- float **hue**
  Teinte.
- float **saturation**
  Saturation.
- float **value**
  Valeur.

4.4.1 Description détaillée

Classe pour les couleurs.

Définition à la ligne 164 du fichier Color.hpp.

4.4.2 Documentation des contructeurs et destructeur

4.4.2.1 Color_Color : :Color_Color (float *f1*, float *f2*, float *f3*, bool *rgb*) `[private]`

Construit une couleur à partir de 3 valeurs RGB ou 3 valeus HSV (dépend du paramètre booléen).

Construit une couleur à partir de 3 valeures flottantes Le paramètre rgb indique si la les composantes sont des valeures RGB ou HSV On calcule immédiatement la conversion entre RGB et HSV

Paramètres :

f1,f2,f3 les 3 composantes

rgb indique le type de coordonnées

Définition à la ligne 191 du fichier Color.cpp.

Références _computeHSV(), _computeRGB(), blue, Color_Color(), green, hue, red, saturation, et value.

Référencé par Color_Color().

4.4.3 Documentation des fonctions membres

4.4.3.1 void Color_Color : :_computeHSV () `[private]`

Fonction de conversion RGB->HSV.

Conversion RGB->HSV Algorithme tiré de www.easyrgb.com

Définition à la ligne 240 du fichier Color.cpp.

Références blue, green, hue, red, saturation, et value.

Référencé par Color_Color().

4.4.3.2 void Color_Color : :_computeRGB () [private]

Fonction de conversion HSV->RGB.

Conversion HSV->RGB Algorithme tiré de www.easyrgb.com

Définition à la ligne 283 du fichier Color.cpp.

Références blue, green, hue, red, saturation, et value.

Référencé par Color_Color().

4.4.3.3 float Color_Color : :distance (const Color_Base * *c*) const [private, virtual]

Distance entre 2 couleurs.

Conversion des coordonnées cylindriques en coordonnées cartésiennes et calcule de la norme euclidienne

Implémente **Color_Base** (p. 12).

Définition à la ligne 214 du fichier Color.cpp.

Références distance(), hue, Pi, saturation, et value.

Référencé par distance().

4.4.3.4 bool Color_Color : :operator< (const Color *rhs*) const [private, virtual]

Opérateur relationnel d'ordre.

L'odre utilisé est l'ordre lexicographique : on compare les composantes HSV des 2 couleurs

Implémente **Color_Base** (p. 13).

Définition à la ligne 331 du fichier Color.cpp.

Références Color : :getHue(), Color : :getSaturation(), Color : :getValue(), hue, operator<(), saturation, et value.

Référencé par operator<().

4.4.3.5 bool Color_Color : :operator== (const Color *rhs*) const [private, virtual]

Opérateur relationnel d'égalité.

L'odre utilisé est l'ordre lexicographique : on compare les composantes HSV des 2 couleurs

Implémente **Color_Base** (p. 12).

Définition à la ligne 353 du fichier Color.cpp.

Références Color : :getHue(), Color : :getSaturation(), Color : :getValue(), hue, operator==(), saturation, et value.

Référencé par operator==().

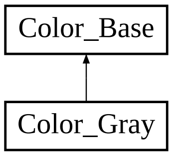
La documentation de cette classe a été générée à partir des fichiers suivants :
- **Color.hpp**
- **Color.cpp**

4.5 Référence de la classe Color_Gray

Classe pour les couleurs en niveaux de gris.

`#include <Color.hpp>`

Graphe d'héritage de Color_Gray : :



Fonctions membres privées

- **Color_Gray** (float _gray)
  Construit une couleur à partir d'un niveau de gris [0 :255.0].
- string **toString** () const
  Retourne une chaîne de caractères détaillée.
- float **getGray** () const
  Retourne le niveau de gris [0 :255].
- float **getRed** () const
  Retourne la composante rouge = niveau de gris [0 :255].
- float **getGreen** () const
  Retourne la composante verte = niveau de gris [0 :255].
- float **getBlue** () const
  Retourne la composante bleue = niveau de gris [0 :255].
- float **getHue** () const
  Retourne la teinte = 0 [0 :1].
- float **getSaturation** () const
  Retourne la saturation teinte = 0 [0 :1].
- float **getValue** () const
  Retourne la valeur = niveau de gris [0 :1].
- float **distance** (const **Color_Base** ∗) const
  Retourn la distance entre 2 niveaux de gris : valeur absoule de la différence.
- bool **operator==** (const **Color**) const
  Oérateur relationnel d'égalité.
- bool **operator<** (const **Color**) const
  opérateur relationnel d'ordre

Attributs privés

- float **gray**

Niveau de gris.

4.5.1 Description détaillée

Classe pour les couleurs en niveaux de gris.

Définition à la ligne 132 du fichier Color.hpp.

4.5.2 Documentation des fonctions membres

4.5.2.1 float Color_Gray : :distance (const Color_Base * *c*) const `[private, virtual]`

Retourn la distance entre 2 niveaux de gris : valeur absoule de la différence.

Valeur absolue de la différence des niveaux de gris

Implémente **Color_Base** (p. 12).

Définition à la ligne 154 du fichier Color.cpp.

Références distance().

Référencé par distance().

4.5.2.2 bool Color_Gray : :operator< (const Color *rhs*) const `[private, virtual]`

opérateur relationnel d'ordre

Comparaison avec l'autre niveau de gris

Implémente **Color_Base** (p. 13).

Définition à la ligne 171 du fichier Color.cpp.

Références Color : :getGray(), gray, et operator<().

Référencé par operator<().

4.5.2.3 bool Color_Gray : :operator== (const Color *rhs*) const `[private, virtual]`

Oérateur relationnel d'égalité.

Egalité avec l'autre niveau de gris

Implémente **Color_Base** (p. 12).

Définition à la ligne 177 du fichier Color.cpp.

Références Color : :getGray(), gray, et operator==().

Référencé par operator==().

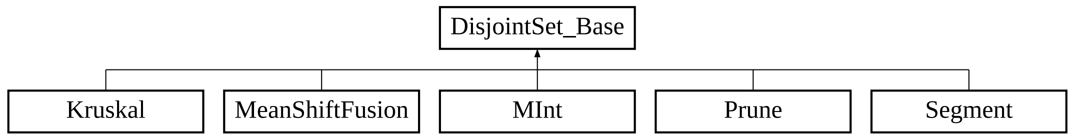
La documentation de cette classe a été générée à partir des fichiers suivants :
- **Color.hpp**
- **Color.cpp**

4.6 Référence de la classe DisjointSet_Base

Classe de base pour les algorithmes utilisant des ensembles disjoints.

`#include <DisjointSet.hpp>`

Graphe d'héritage de DisjointSet_Base : :



Fonctions membres publiques

- **DisjointSet_Base** (const **Node**)
  Constructeur de l'algorithme.
- virtual string **toString** ()
  Retourne une chaîne de caractères détaillée.
- **Node mergeDisjointSet** (**Node**, **Node**)
  Fusionne 2 noeuds.
- virtual **Node makeDisjointSet** ()=0
  Accomplis effectivement un algorithme de clusteringe.
- bool **sameDisjointSet** (**Node**, **Node**)
  Teste l'appartenance au même cluster de 2 points.
- **Node makeNode** (string)
  Retourne sous forme de graphe un le résultat du clustering.
- int **getParentSize** (**Node**)
  Retourne la taille du parent d'un noeud.
- **Node getParent** (**Node**)
  Retourne le cluster d'un noeud.

Fonctions membres protégées

- int **getParentRank** (int)
  retourn le rang d'un parent

Attributs protégés

- **Node graph**
  le graphe à clusteriser
- vector< **Node** > **vNode**
  les noeuds à étudier

– int **nbClass**
  le nombre de cluster en cours

– vector< **Edge** > **edge**
  les arretes du graphe clusterisés

Types privés

– typedef map< int, **Node** > : :iterator **table_iter**
  Itérateur des parents vers les noeuds.

Attributs privés

– vector< int > **parent**
  tableau d'index des parents d'un cluster

– vector< int > **size**
  tableau des tailles d'un cluster

– map< int, **Node** > **table**
  map des parents vers les noeuds

4.6.1 Description détaillée

Classe de base pour les algorithmes utilisant des ensembles disjoints.

Cette classe implémente la compression de chemin

Définition à la ligne 20 du fichier DisjointSet.hpp.

4.6.2 Documentation des contructeurs et destructeur

4.6.2.1 DisjointSet_Base : :DisjointSet_Base (const *Node*)

Constructeur de l'algorithme.

Paramètres :
  Node : le noeud à clusteriser

Définition à la ligne 8 du fichier DisjointSet.cpp.

Références DisjointSet_Base(), Node : :getLeafSize(), Node : :getNodeIterator(), Node : :getNodeSize(), graph, Node_Iterator : :hasNext(), nbClass, Node_Iterator : :next(), parent, Node : :setRank(), size, et vNode.

Référencé par DisjointSet_Base().

4.6.3 Documentation des fonctions membres

4.6.3.1 int DisjointSet_Base : :getParentRank (int *i*) `[protected]`

retourn le rang d'un parent

Retourne l'index du parent Réalise la compression de chemin

Définition à la ligne 31 du fichier DisjointSet.cpp.

Références getParentRank(), et parent.

Référencé par getParent(), getParentRank(), getParentSize(), makeNode(), mergeDisjointSet(), et sameDisjointSet().

4.6.3.2 Node DisjointSet_Base : :makeNode (string *label*)

Retourne sous forme de graphe un le résultat du clustering.

Construit un noeud après la fusion

Définition à la ligne 143 du fichier DisjointSet.cpp.

Références Node : :addEdge(), Node : :addNode(), edge, Edge : :getDST(), getParentRank(), Node : :getRank(), Edge : :getSRC(), Edge : :getWeight(), Edge : :getX0(), Edge : :getX1(), Edge : :getY0(), Edge : :getY1(), makeNode(), parent, Edge : :setCoord(), Node : :setPixel(), Node : :setWeight(), table, table_iter, et vNode.

Référencé par makeNode().

4.6.3.3 Node DisjointSet_Base : :mergeDisjointSet (Node *n1*, Node *n2*)

Fusionne 2 noeuds.

Fusion de 2 ensembles disjoint Réalise l'union par rang

Définition à la ligne 43 du fichier DisjointSet.cpp.

Références Node : :addNode(), getParentRank(), Node : :getRank(), Node : :merge(), mergeDisjointSet(), nbClass, parent, size, table, et table_iter.

Référencé par mergeDisjointSet().

La documentation de cette classe a été générée à partir des fichiers suivants :
- **DisjointSet.hpp**
- **DisjointSet.cpp**

4.7 Référence de la classe Edge

Interface sur une classe **Edge_Base**(p. 25).

`#include <Edge.hpp>`

Fonctions membres publiques

- **Edge** ()
  Constructeur par défaut.
- **Edge** (**Node**, **Node**, float)
  Constructeur à partir de 2 noeuds et poids de l'arrête.
- **Edge** (const **Edge** &edge)
  Constructeur de copie.
- **Edge** & **operator=** (const **Edge** &)
  Operateur d'affectation.
- ~**Edge** ()
  Destructeur.
- string **toString** () const
  Retourne une chaîne de caractères détaillée.
- **Node** **getSRC** () const
  Retourne le noeud source de l'arrête.
- **Node** **getDST** () const
  Retourne le noeud destination de l'arrête.
- double **getWeight** () const
  Retourne le poids de l'arrête.
- void **setCoord** (int, int, int, int)
  Fixe les coordonées de l'arrête.
- int **getX0** () const
  Coordonnée x0.
- int **getY0** () const
  Coordonnée y0.
- int **getX1** () const
  Coordonnée x1.
- int **getY1** () const
  Coordonnée y1.

Attributs privés

- **Edge_Base** ∗ **p**
  Classe de base.

4.7.1 Description détaillée

Interface sur une classe **Edge_Base**(p. 25).

Gère un compteur de références Les méthodes sont redirigées vers une instance de **Edge_-Base**(p. 25)

Définition à la ligne 17 du fichier Edge.hpp.

4.7.2 Documentation des contructeurs et destructeur

4.7.2.1 Edge : :Edge (Node, Node, float)

Constructeur à partir de 2 noeuds et poids de l'arrête.

Paramètres :

Node un noeud

Node un noeud

float poids de l'arrête

Définition à la ligne 7 du fichier Edge.cpp.

Références p.

La documentation de cette classe a été générée à partir des fichiers suivants :
- **Edge.hpp**
- **Edge.cpp**

4.8 Référence de la classe Edge_Base

Arrete entre noeuds.

`#include <Edge.hpp>`

Fonctions membres privées

- **Edge_Base** (**Node**, **Node**, float)
  Constructeur à partir de 2 noeuds et poids de l'arrête.
- string **toString** () const
  Retourne une chaîne de caractères détaillée.
- **Node getSRC** () const
  Retourne le noeud source de l'arrête.
- **Node getDST** () const
  Retourne le noeud destination de l'arrête.
- float **getWeight** () const
  Retourne le poids de l'arrête.
- void **setCoord** (int, int, int, int)
  Fixe les coordonées de l'arrête.
- int **getX0** () const
  Coordonnée x0.
- int **getY0** () const
  Coordonnée y0.
- int **getX1** () const
  Coordonnée x1.
- int **getY1** () const
  Coordonnée y1.

Attributs privés

- int **use**
  Compteur de référence.
- **Node src**
  Noeud source.
- **Node dst**
  Noeud destination.
- float **weight**
  Poids.
- int **x0**
  Position pour le dessin du graphe.

– int **y0**
  Position pour le dessin du graphe.

– int **x1**
  Position pour le dessin du graphe.

– int **y1**
  Position pour le dessin du graphe.

4.8.1 Description détaillée

Arrete entre noeuds.

Définition à la ligne 58 du fichier Edge.hpp.

4.8.2 Documentation des contructeurs et destructeur

4.8.2.1 Edge_Base : :Edge_Base (Node, Node, float) `[private]`

Constructeur à partir de 2 noeuds et poids de l'arrête.

Paramètres :
  Node source
  Node destination
  float poids de l'arrête

Définition à la ligne 58 du fichier Edge.cpp.

Références Edge_Base().

Référencé par Edge_Base().

La documentation de cette classe a été générée à partir des fichiers suivants :
– **Edge.hpp**
– **Edge.cpp**

4.9 Référence de la classe Edge_Iterator

Parcourt les arrêtes d'un graphe.

`#include <Iterator.hpp>`

Fonctions membres publiques

- bool **hasNext** ()
  Y'a-t-il un prochain élément ?
- **Edge next** ()
  Accède au prochain élément.

Types privés

- typedef vector< **Edge** > : :const_iterator **itEdge**
  Itérateur sur le vecteur d'arrête.

Fonctions membres privées

- **Edge_Iterator** (const **Node_Graph** ∗)
  Constructeur à partir d'un noeud graphe.
- **Edge_Iterator** (const **Node_Leaf** ∗)
  Constructeur à partir d'un noeud feuille.

Attributs privés

- **itEdge end**
  itérateur de fin
- **itEdge current**
  itérateur de l'arrête courante

4.9.1 Description détaillée

Parcourt les arrêtes d'un graphe.

Les arrêtes d'un graphe sont stockés dans une liste. Cette classe permet de parcourir les arrêtes d'un graph sans connaissance de cette liste.

Définition à la ligne 41 du fichier Iterator.hpp.

4.9.2 Documentation des fonctions membres

4.9.2.1 bool Edge_Iterator : :hasNext ()

Y'a-t-il un prochain élément ?

Renvoie :
vrai s'il y a encore une arrête

Définition à la ligne 41 du fichier Iterator.cpp.

Références current, et end.

Référencé par Node_Graph : :drawEdgeEPS(), Prune : :makeDisjointSet(), MeanShiftFusion : :makeDisjointSet(), Segment : :makeDisjointSet(), MInt : :makeDisjointSet(), Kruskal : :makeDisjointSet(), et Node_Graph : :toString().

4.9.2.2 Edge Edge_Iterator : :next ()

Accède au prochain élément.

Renvoie :
l'arrête suivante

Définition à la ligne 45 du fichier Iterator.cpp.

Références current.

Référencé par Node_Graph : :drawEdgeEPS(), Prune : :makeDisjointSet(), MeanShiftFusion : :makeDisjointSet(), Segment : :makeDisjointSet(), MInt : :makeDisjointSet(), Kruskal : :makeDisjointSet(), et Node_Graph : :toString().

La documentation de cette classe a été générée à partir des fichiers suivants :
- **Iterator.hpp**
- **Iterator.cpp**

4.10 Référence de la classe EPSWriter

Générateur de fichier PostScript.

`#include <EPSWriter.hpp>`

Fonctions membres publiques

- **EPSWriter** (string, int, int)
  Constructeur.
- void **setScale** (int, int)
  Fixe les échelles.
- void **setColor** (**Color**)
  Fixe la couleur courante.
- void **drawRound** (int, int)
  Dessine un petit rond.
- void **drawLine** (int, int, int, int)
  Dessine une ligne.
- void **write** ()
  Ferme le fichier.

Attributs privés

- int **xScale**
  Facteurs d'échelles horizontales et verticales.
- int **yScale**
  Facteurs d'échelles horizontales et verticales.
- **Color color**
  Couleur courante.
- ofstream **file**
  Fichier PostScript.
- int **width**
  Largeur et hauteur de du fichier.
- int **height**
  Largeur et hauteur de du fichier.

4.10.1 Description détaillée

Générateur de fichier PostScript.

Définition à la ligne 15 du fichier EPSWriter.hpp.

4.10.2 Documentation des contructeurs et destructeur

4.10.2.1 EPSWriter : :EPSWriter (string, int, int)

Constructeur.

Paramètres :

string le nom du fichier

int largeur du dessin

int hauteur du dessin

Définition à la ligne 6 du fichier EPSWriter.cpp.

Références EPSWriter(), file, height, width, xScale, et yScale.

Référencé par EPSWriter().

4.10.3 Documentation des fonctions membres

4.10.3.1 void EPSWriter : :drawLine (int, int, int, int)

Dessine une ligne.

Paramètres :

int coordonnée X0

int coordonnée Y0

int coordonnée X1

int coordonnée Y1

Définition à la ligne 35 du fichier EPSWriter.cpp.

Références drawLine(), file, xScale, et yScale.

Référencé par Node_Graph : :drawEdgeEPS(), et drawLine().

4.10.3.2 void EPSWriter : :drawRound (int, int)

Dessine un petit rond.

Paramètres :

int coordonnée X

int coordonnée Y

Définition à la ligne 27 du fichier EPSWriter.cpp.

Références color, drawRound(), file, Color : :getBlue(), Color : :getGreen(), Color : :getRed(), xScale, et yScale.

Référencé par Node_Leaf : :drawNodeEPS(), et drawRound().

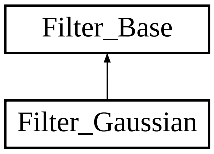
La documentation de cette classe a été générée à partir des fichiers suivants :
- **EPSWriter.hpp**
- **EPSWriter.cpp**

4.11 Référence de la classe Filter_Base

Classe de base pour les filtres.

`#include <Filter.hpp>`

Graphe d'héritage de Filter_Base : :



Fonctions membres publiques

- **Filter_Base** (int)
  Constructeur.
- virtual string **toString** ()=0
  Retourne une chaîne de caractères détaillée.
- **PNG apply** (**PNG**)
  Applique le fitre à une image.

Fonctions membres protégées

- virtual void **computeMask** ()=0
  Méthode à redéfinir pour chaque filtre.
- void **normalise** ()
  Normalise les valeurs du filtre.
- float **getMask** (int, int)
  Valeur du filtre en un pointe.
- void **setMask** (int, int, float)
  Fixe la valeur du filtre en un pointe.

Attributs protégés

- int **size**
  Dimension du filtre.
- float ∗ **mask**
  Poids du filtre.

4.11.1 Description détaillée

Classe de base pour les filtres.

Définition à la ligne 19 du fichier Filter.hpp.

4.11.2 Documentation des contructeurs et destructeur

4.11.2.1 Filter_Base : :Filter_Base (int)

Constructeur.

Paramètres :
 size Dimension du filtre

Définition à la ligne 7 du fichier Filter.cpp.

Références Filter_Base().

Référencé par Filter_Base().

4.11.3 Documentation des fonctions membres

4.11.3.1 PNG Filter_Base : :apply (PNG)

Applique le fitre à une image.

Paramètres :
 image source

Renvoie :
 image filtrée

Définition à la ligne 34 du fichier Filter.cpp.

Références apply(), Color : :getBlue(), PNG : :getColor(), Color : :getGray(), Color : :getGreen(), PNG : :getHeight(), getMask(), Color : :getRed(), Color : :getType(), PNG : :getType(), PNG : :getWidth(), PNG_COLOR, PNG_GRAY, PNG : :setColor(), et size.

Référencé par apply().

4.11.3.2 void Filter_Base : :normalise () `[protected]`

Normalise les valeurs du filtre.

Sommation des valeurs du filtres et division de chaque coefficients par cette somme.

Définition à la ligne 17 du fichier Filter.cpp.

Références getMask(), setMask(), et size.

Référencé par Filter_Gaussian : :Filter_Gaussian().

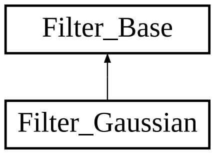
La documentation de cette classe a été générée à partir des fichiers suivants :
- **Filter.hpp**
- **Filter.cpp**

4.12 Référence de la classe Filter_Gaussian

Filtre gaussien.

`#include <Filter.hpp>`

Graphe d'héritage de Filter_Gaussian : :



Fonctions membres publiques

- **Filter_Gaussian** (int, float, float)
  Filtre gaussien.
- string **toString** ()
  Retourne une chaîne de caractères détaillée.

Fonctions membres privées

- void **computeMask** ()
  Calcul les valeurs du filtres.

Attributs privés

- float **sigma**
  Sigma.
- float **mu**
  Mue.

4.12.1 Description détaillée

Filtre gaussien.

Définition à la ligne 47 du fichier Filter.hpp.

4.12.2 Documentation des contructeurs et destructeur

4.12.2.1 Filter_Gaussian : :Filter_Gaussian (int, float, float)

Filtre gaussien.

Paramètres :
size dimension du filtre

sigma
mu

Définition à la ligne 77 du fichier Filter.cpp.

Références computeMask(), Filter_Gaussian(), mu, Filter_Base : :normalise(), et sigma.

Référencé par Filter_Gaussian().

4.12.3 Documentation des fonctions membres

4.12.3.1 void Filter_Gaussian : :computeMask () `[private, virtual]`

Calcul les valeurs du filtres.

Calcul du filtre gaussien

Implémente **Filter_Base** (p. 31).

Définition à la ligne 86 du fichier Filter.cpp.

Références mu, et sigma.

Référencé par Filter_Gaussian().

La documentation de cette classe a été générée à partir des fichiers suivants :
- **Filter.hpp**
- **Filter.cpp**

4.13 Référence de la classe FVector

Interface sur un **FVector_Base**(p. 38).

`#include <FVector.hpp>`

Fonctions membres publiques

- **FVector** ()
  Constructeur par défaut.
- **FVector** (float, float, float, float, float)
  Constructeur à partir des coordonnées dans l'espace des Features.
- string **toString** () const
  Retourne une chaîne de caractères détaillée.
- **FVector** (const **FVector** &)
  Constructeur de copie.
- **FVector** & **operator=** (const **FVector** &)
  Opérateur d'affectation.
- ~**FVector** ()
  Destructeur.
- int **getX** () const
  retourne la coordonnée x entière
- int **getY** () const
  retourne la coordonnée y entière
- float **operator[]** (int) const
  retourne la ième coordonnée
- **FVector** & **operator+=** (const **FVector** &)
  Opérateur d'addition de vecteur.
- **FVector** & **operator/=** (const float &)
  Opérateur de division par un scalaire.
- bool **operator<** (const **FVector**) const
  Opérateur relationnel d'ordre.

Attributs privés

- **FVector_Base** * **p**
  Pointeur sur un vecteur.

Amis

- float **norm** (const **FVector** &, const **FVector** &)
  Calcule la norme euclidienne de 2 vecteurs.

– float **normColor** (const **FVector** &, const **FVector** &)
Calcule la norme euclidienne entre les couleurs de 2 vecteur.

4.13.1 Description détaillée

Interface sur un **FVector_Base**(p. 38).

Gère un compteur de références La pluspart des méthodes sont redirigées vers une instance de FVector

Définition à la ligne 17 du fichier FVector.hpp.

4.13.2 Documentation des contructeurs et destructeur

4.13.2.1 FVector : :FVector (float, float, float, float, float)

Constructeur à partir des coordonnées dans l'espace des Features.

Paramètres :
float coordonnée x
float coordonnée y
float coordonnée h
float coordonnée s
float coordonnée v

Définition à la ligne 10 du fichier FVector.cpp.

Références p.

4.13.3 Documentation des fonctions membres

4.13.3.1 float FVector : :operator[ ] (int) const

retourne la ième coordonnée

Paramètres :
int axe

Définition à la ligne 39 du fichier FVector.cpp.

Références operator[ ](), et p.

Référencé par operator[ ]().

4.13.4 Documentation des fonctions amies et associées

4.13.4.1 float norm (const FVector &, const FVector &) `[friend]`

Calcule la norme euclidienne de 2 vecteurs.

Paramètres :
FVector& 1er vecteur

FVector& 2ème vecteur

Définition à la ligne 54 du fichier FVector.cpp.

4.13.4.2 float normColor (const FVector &, const FVector &) `[friend]`

Calcule la norme euclidienne entre les couleurs de 2 vecteur.

Paramètres :

FVector& 1er vecteur

FVector& 2ème vecteur

Définition à la ligne 57 du fichier FVector.cpp.

La documentation de cette classe a été générée à partir des fichiers suivants :
- **FVector.hpp**
- **FVector.cpp**

4.14 Référence de la classe FVector_Base

Classe de base abstraite pour les **FVector**(p. 35).

`#include <FVector.hpp>`

Fonctions membres publiques

- float **operator[]** (int i) const
  retourne la ième coordonnée

Attributs publics statiques

- const int **dim** = 5
  Dimension de l'espace.

Fonctions membres privées

- **FVector_Base** ()
  Constructeur par défaut.
- **FVector_Base** (float, float, float, float, float)
  Constructeur à partir des coordonnées dans l'espace des Features.
- ~**FVector_Base** ()
  Destructeur.
- int **getX** () const
  retourne la coordonnée x entière
- int **getY** () const
  retourne la coordonnée y entière
- **FVector_Base** & **operator+=** (const **FVector_Base** *)
  Opérateur d'addition de vecteur.
- **FVector_Base** & **operator/=** (const float &)
  Opérateur de division par un scalaire.
- bool **operator<** (const **FVector_Base** *) const
  Opérateur relationnel : utilise l'ordre lexicographique.
- string **toString** () const
  Affiche une chaîne de caractères détaillée.

Attributs privés

- int **use**
  Compteur de référence.
- float * **data**
  Données.

4.14.1 Description détaillée

Classe de base abstraite pour les **FVector**(p. 35).

C'est cette classe qui possède le compteur de référence

Définition à la ligne 67 du fichier FVector.hpp.

4.14.2 Documentation des contructeurs et destructeur

4.14.2.1 FVector_Base : :FVector_Base (float, float, float, float, float) `[private]`

Constructeur à partir des coordonnées dans l'espace des Features.

Paramètres :

float coordonnée x

float coordonnée y

float coordonnée h

float coordonnée s

float coordonnée v

Définition à la ligne 70 du fichier FVector.cpp.

Références data, et dim.

4.14.3 Documentation des fonctions membres

4.14.3.1 float FVector_Base : :operator[] (int *i*) const

retourne la ième coordonnée

Paramètres :

int axe

Définition à la ligne 95 du fichier FVector.cpp.

Références operator[]().

Référencé par operator[]().

La documentation de cette classe a été générée à partir des fichiers suivants :
- **FVector.hpp**
- **FVector.cpp**

4.15 Référence de la classe Histogram

Histogram.

`#include <Histogram.hpp>`

Fonctions membres publiques

- **Histogram** ()
  Constructeur par défaut.
- string **toString** () const
  Retourne une chaîne de caractères détaillée.
- void **addPixel** (**Pixel**)
  Ajoute un pixel à l'histogramme.
- void **makeCumul** ()
  Calcule l'histogramme cumulé.
- void **makeClass** (int classNumber)
  Sépare l'ensemble des couleurs en un nombre classe.
- **Pixel getElementAt** (const int) const
  Retourne le pixel représentant la classe du paramètre.
- int **getSize** () const
  Retourne le nombre de classe.
- **Color getColor** (int) const
  Retourne la couleur de la ième classe.

Types privés

- typedef map< **Color**, **HistogramClass** > : :iterator **map_iter**
  Itérateur.
- typedef map< **Color**, **HistogramClass** > : :const_iterator **const_map_iter**
  Itérateur constant.

Attributs privés

- int **pixelNumber**
  Nombre total de pixel.
- map< **Color**, **HistogramClass** > **data**
  Une hashtable faisant la correspondance.
- vector< **Color** > **classColor**

4.15.1 Description détaillée

Histogram.

Met en correspondance une couleur et sa classe

Définition à la ligne 49 du fichier Histogram.hpp.

4.15.2 Documentation des fonctions membres

4.15.2.1 void Histogram : :addPixel (Pixel *pixel*)

Ajoute un pixel à l'histogramme.

Met en correspondance une couleur et sa classe

Définition à la ligne 37 du fichier Histogram.cpp.

Références HistogramClass : :addPixel(), addPixel(), data, Pixel : :getColor(), map_iter, et pixelNumber.

Référencé par addPixel(), et PNG_Base : :getHistogram().

4.15.2.2 Pixel Histogram : :getElementAt (const int *i*) const

Retourne le pixel représentant la classe du paramètre.

Parcourt chaque classe jusqu'à arriver au rang demandé

Définition à la ligne 114 du fichier Histogram.cpp.

Références getElementAt().

Référencé par getElementAt(), et makeClass().

4.15.2.3 void Histogram : :makeClass (int *classNumber*)

Sépare l'ensemble des couleurs en un nombre classe.

Sépare l'ensemble des couleurs en un nombre classe. Le but de cette méthode est de fournir de bonnes valeurs de départ à l'algorithme des **KMean**(p. 43)

Définition à la ligne 78 du fichier Histogram.cpp.

Références Color : :getColor(), Pixel : :getColor(), getElementAt(), makeClass(), et pixelNumber.

Référencé par KMean : :init(), et makeClass().

4.15.2.4 void Histogram : :makeCumul ()

Calcule l'histogramme cumulé.

Calcul de l'histogramme cumulé. On parcourt chaque classe en mettant à jour le total de la suivante

Définition à la ligne 64 du fichier Histogram.cpp.

Références data, et map_iter.

Référencé par KMean : :init().

La documentation de cette classe a été générée à partir des fichiers suivants :
- **Histogram.hpp**
- **Histogram.cpp**

4.16 Référence de la classe HistogramClass

HistogramClass.

`#include <Histogram.hpp>`

Fonctions membres publiques

- **HistogramClass** (**Color**)
  Classe de cette couleur.
- string **toString** () const
  Retourne une chaîne de caractères détaillée.
- void **addPixel** (**Pixel**)
  Ajoute un pixel à la classe.
- int **getSize** () const
  Retourne le nombre de pixel de cette classe.
- void **setTotal** (int)
  Fixe le nombre de pixel pour l'histogramme cumulé.
- int **getTotal** () const
  Retourne le nombre de pixel pour l'histogramme cumulé.
- **Pixel getElementAt** (const int) const
  Retourne le ième pixel de la classe.

Attributs privés

- **Color color**
  Couleur de la classe.
- vector< **Pixel** > **data**
  Pixels de cette même couleur.
- int **total**
  Nombre de pixel pour l'histogramme cumulé.

4.16.1 Description détaillée

HistogramClass.

Comptabilise les pixels d'une image d'une certaine couleur

Définition à la ligne 21 du fichier Histogram.hpp.

La documentation de cette classe a été générée à partir des fichiers suivants :
- **Histogram.hpp**
- **Histogram.cpp**

4.17 Référence de la classe KMean

Implémente l'algorithmes des K-Means.

`#include <KMean.hpp>`

Fonctions membres publiques

- **KMean** (**PNG**, int)
  Constructeur avec une image et le nombre de classe.
- string **toString** () const
  Retourne une chaîne de caractères détaillée.
- void **doKMean** ()
  Execute l'algorithme.
- void **output** (string) const
  Production d'un fichier **PNG**(p. 75).
- **Node makeNode** () const
  Construit un graphe des composantes connexes.

Fonctions membres protégées

- void **init** ()
  Initialise les centres.

Attributs privés

- int **k**
  Nombre de centre.
- **PNG image**
  Image à traiter.
- vector< **Pixel** > **pixel**
  Pixel(p. 69) *de l'image.*
- vector< **KMeanClass** > **center**
  Les classes.

4.17.1 Description détaillée

Implémente l'algorithmes des K-Means.

Définition à la ligne 48 du fichier KMean.hpp.

4.17.2 Documentation des contructeurs et destructeur

4.17.2.1 KMean : :KMean (PNG, int)

Constructeur avec une image et le nombre de classe.

Paramètres :
image l'image à traîter
k le nombre de centre

Définition à la ligne 112 du fichier KMean.cpp.

Références init(), et KMean().

Référencé par KMean().

4.17.3 Documentation des fonctions membres

4.17.3.1 void KMean : :doKMean ()

Execute l'algorithme.

Exécution de l'algorithme des KMean. Tant que des pixels changent de centre et que le nombre d'itération est plus petit que MAX_ITER, on continue l'algorithme

Définition à la ligne 152 du fichier KMean.cpp.

Références center, Color : :distance(), Pixel : :getColor(), k, pixel, et Trace : :print().

4.17.3.2 void KMean : :init () `[protected]`

Initialise les centres.

Initialisation de la class KMean. Pour avoir une bonne répartition des points dans chaque classe, on se sert de l'histogramme cumulé de l'image.

Définition à la ligne 120 du fichier KMean.cpp.

Références center, Histogram : :getColor(), PNG : :getColor(), PNG : :getHeight(), PNG : :getHistogram(), Histogram : :getSize(), PNG : :getWidth(), image, k, Histogram : :makeClass(), Histogram : :makeCumul(), pixel, et KMeanClass : :setColor().

Référencé par KMean().

La documentation de cette classe a été générée à partir des fichiers suivants :
- **KMean.hpp**
- **KMean.cpp**

4.18 Référence de la classe KMeanClass

KMeanClass : représente un centroïde.

`#include <KMean.hpp>`

Fonctions membres publiques

- **KMeanClass** ()
  Constructeur par défaut.
- string **toString** () const
  Retourne une chaîne de caractères détaillée.
- void **setColor** (**Color**)
  Fixe le centre de la classe.
- **Color getColor** () const
  Retourne le centre de la classe.
- void **addPixel** (**Pixel**)
  Ajoute un pixel à la classe.
- void **computeMean** ()
  Calcule la moyenne de la classe.
- void **output** (**PNG**) const
  Méthode auxiliaire pour produire un fichier **PNG**(p. 75).
- void **getContent** (**PixelMap** &) const
  Méthode auxiliaire pour obtenir les composantes connexes.

Attributs privés

- **Color color**
  Centre de la classe.
- vector< **Pixel** > **pixel**
  Pixels associés à ce centre.

4.18.1 Description détaillée

KMeanClass : représente un centroïde.

Définition à la ligne 19 du fichier KMean.hpp.

4.18.2 Documentation des fonctions membres

4.18.2.1 void KMeanClass : :computeMean ()

Calcule la moyenne de la classe.

Calcule de la moyenne d'un centre. Pour les images en niveaux de gris on calcule la moyenne des niveaux. Pour les images couleurs on fait la moyenne sur les composantes HSV indépendament.

Définition à la ligne 30 du fichier KMean.cpp.

Références color, Color : :distance(), getColor(), Color : :getGray(), Color : :getHue(), Color : :getSaturation(), Color : :getType(), Color : :getValue(), et pixel.

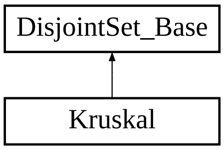
La documentation de cette classe a été générée à partir des fichiers suivants :
- **KMean.hpp**
- **KMean.cpp**

4.19 Référence de la classe Kruskal

Kruskal : classe de test.

#include <DisjointSet.hpp>

Graphe d'héritage de Kruskal : :



Fonctions membres publiques

- **Kruskal** (**Node**)
  Prend comme argument le graphe dont on veut le MST.
- **Node makeDisjointSet** ()
  Implémentation de l'algorithme de Kruskal.

4.19.1 Description détaillée

Kruskal : classe de test.

Implemente l'algorithme de Kruskal pour trouver le MST d'un graphe

Définition à la ligne 64 du fichier DisjointSet.hpp.

La documentation de cette classe a été générée à partir des fichiers suivants :
- **DisjointSet.hpp**
- **DisjointSet.cpp**

4.20 Référence de la classe Lattice

Représente une zone rectangulaire de vecteurs de Features.

`#include <MeanShift.hpp>`

Fonctions membres publiques

- **Lattice** (**PNG**)
  Crée à partir d'une image.
- int **getWidth** () const
  Retourne la largeur.
- int **getHeight** () const
  Retourne la hauteur.
- **FVector getFVector** (const int, const int)
  Retourne le vecteur (i, j).
- **FVector getOutputFVector** (const int, const int) const
  Retourne le résultat du filtre (i, j).
- void **setOutputFVector** (const int, const int, const **FVector**)
  Fixe le résultat du filtre (i, j).
- **ELATTICE_STATE getState** (int, int) const
  Retourne l'état du vecteur (i, j).
- void **setState** (const int, const int, const **ELATTICE_STATE**)
  Fixe l'état du vecteur (i, j).

Attributs privés

- **PNG image**
  Image représentée.
- int **width**
  Largeur, hauteur et taille de l'imisate.
- int **height**
  Largeur, hauteur et taille de l'imisate.
- int **size**
  Largeur, hauteur et taille de l'imisate.
- vector< int > **visited**
  Cache des points explorés.
- vector< **FVector** > **input**
  Valeurs d'entrée.
- vector< **FVector** > **output**
  Valeurs de sortie.

– vector< **ELATTICE_STATE** > **state**
  Etat des point explorés.

4.20.1 Description détaillée

Représente une zone rectangulaire de vecteurs de Features.

Données du filtre Meanshift

Définition à la ligne 27 du fichier MeanShift.hpp.

La documentation de cette classe a été générée à partir des fichiers suivants :
– **MeanShift.hpp**
– **MeanShift.cpp**

4.21 Référence de la classe MeanShift

Calcul du filtre Meanshift.

`#include <MeanShift.hpp>`

Fonctions membres publiques

- **MeanShift** (int, float, **PNG**, float=0.1, int=30)
  Algorithme du filtre Meanshift.
- string **toString** () const
  Affiche une chaîne de caractères détaillé.
- void **findMode** (const int, const int)
  Trouve un mode.
- void **doFilter** ()
  Calcul le filtre Meanshift.
- **Node makeNode** ()
  Construit un graphe d'adjacence des modes calculés.

Attributs privés

- **Lattice lattice**
  Données à explorer.
- **MeanShiftWindow window**
  Fenêtre d'exploration.
- int **size**
  Dimension spatial.
- float **radius**
  Dimension dans l'espace des couleurs.
- int **maxIter**
  Limite pour le calcul d'un mode.
- float **tol**
  Limite pour le speedup.

4.21.1 Description détaillée

Calcul du filtre Meanshift.

Définition à la ligne 96 du fichier MeanShift.hpp.

4.21.2 Documentation des contructeurs et destructeur

4.21.2.1 MeanShift : :MeanShift (int, float, PNG, float = `0.1`, int = `30`)

Algorithme du filtre Meanshift.

Paramètres :
- ***int*** dimension spatial
- ***float*** dimension dans l'espace des couleurs
- ***PNG*** image à filtrer
- ***float*** tolérance pour le speedup
- ***int*** maximum pour le calcul d'un mode

Définition à la ligne 132 du fichier MeanShift.cpp.

Références MeanShift().

Référencé par MeanShift().

4.21.3 Documentation des fonctions membres

4.21.3.1 void MeanShift : :findMode (const *int*, const *int*)

Trouve un mode.

Paramètres :
- ***int*** coordonnée x du mode à chercher
- ***int*** coordonnée y du mode à chercher

Définition à la ligne 147 du fichier MeanShift.cpp.

Références MeanShiftWindow : :computeMean(), ELATTICE_STATE, findMode(), MeanShiftWindow : :findPixel(), MeanShiftWindow : :getCenter(), Lattice : :getFVector(), Lattice : :getState(), FVector : :getX(), FVector : :getY(), lattice, norm(), normColor(), MeanShiftWindow : :setCenter(), Lattice : :setOutputFVector(), Lattice : :setState(), tol, et window.

Référencé par doFilter(), et findMode().

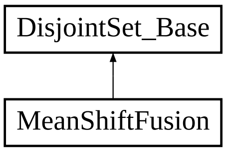
La documentation de cette classe a été générée à partir des fichiers suivants :
- **MeanShift.hpp**
- **MeanShift.cpp**

4.22 Référence de la classe MeanShiftFusion

Meanshift.

#include <DisjointSet.hpp>

Graphe d'héritage de MeanShiftFusion : :



Fonctions membres publiques

– **MeanShiftFusion** (**Node**, float)
  Constructeur.

– **Node makeDisjointSet** ()
  Implémente l'algorithme.

Attributs privés

– float **radius**
  Compteur de fusion.

4.22.1 Description détaillée

Meanshift.

Fusion des régions pour le **MeanShift**(p. 50)

Définition à la ligne 116 du fichier DisjointSet.hpp.

4.22.2 Documentation des contructeurs et destructeur

4.22.2.1 MeanShiftFusion : :MeanShiftFusion (Node, float)

Constructeur.

Paramètres :
  le graphe de l'image à segmenter

Définition à la ligne 486 du fichier DisjointSet.cpp.

Références MeanShiftFusion().

Référencé par MeanShiftFusion().

La documentation de cette classe a été générée à partir des fichiers suivants :
– **DisjointSet.hpp**
– **DisjointSet.cpp**

4.23 Référence de la classe MeanShiftWindow

Représente la fenêtre du Meanshift.

`#include <MeanShift.hpp>`

Fonctions membres publiques

- **MeanShiftWindow** (int, float, **Lattice** &)
  Construit une fenêtre pour le filtre Meanshift.
- string **toString** () const
  Retourne une chaîne de caractères détaillée.
- void **setCenter** (const int, const int)
  Fixe le centre de la fenêtre.
- void **findPixel** ()
  Cherche les données correspondant au centre.
- float **computeMean** ()
  Calcule la moyenne.
- **FVector getCenter** () const
  Retourne le centre de la fenêtre.

Attributs privés

- **FVector center**
  Centre de la fenêtre.
- int **size**
  Dimension spatial de la fenêtre.
- float **radius**
  Dimension dans l'espace des couleurs.
- **Lattice** & **lattice**
  Données de à explorer.
- vector< **FVector** > **data**
  Données de la fenêtre.

4.23.1 Description détaillée

Représente la fenêtre du Meanshift.

Définition à la ligne 62 du fichier MeanShift.hpp.

4.23.2 Documentation des contructeurs et destructeur

4.23.2.1 MeanShiftWindow : :MeanShiftWindow (int, float, Lattice &)

Construit une fenêtre pour le filtre Meanshift.

Paramètres :
- ***int*** dimesion spatial
- ***float*** dimension dans l'espace des couleurs
- ***Lattice&*** données à explorer

Définition à la ligne 57 du fichier MeanShift.cpp.

Références MeanShiftWindow().

Référencé par MeanShiftWindow().

4.23.3 Documentation des fonctions membres

4.23.3.1 float MeanShiftWindow : :computeMean ()

Calcule la moyenne.

Renvoie :
float norme entre le centre et la moyenne calculée

Définition à la ligne 113 du fichier MeanShift.cpp.

Références center, data, et norm().

Référencé par MeanShift : :findMode().

4.23.3.2 void MeanShiftWindow : :setCenter (const *int*, const *int*)

Fixe le centre de la fenêtre.

Paramètres :
- ***int*** coordonnée spatial x
- ***int*** coordonnée spatial y

Définition à la ligne 70 du fichier MeanShift.cpp.

Références center, Lattice : :getFVector(), lattice, et setCenter().

Référencé par MeanShift : :findMode(), et setCenter().

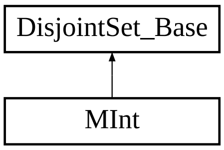
La documentation de cette classe a été générée à partir des fichiers suivants :
- **MeanShift.hpp**
- **MeanShift.cpp**

4.24 Référence de la classe MInt

MInt : classe de l'article.

`#include <DisjointSet.hpp>`

Graphe d'héritage de MInt : :



Fonctions membres publiques

- **MInt** (**Node**, float)
  Constructeur.
- **Node makeDisjointSet** ()
  Implémente l'algorithme.

Attributs privés

- float **tau**
  Paramètre de l'algorithme.

4.24.1 Description détaillée

MInt : classe de l'article.

Implemente l'algorithme de l'article pour segmenter une image

Définition à la ligne 76 du fichier DisjointSet.hpp.

4.24.2 Documentation des contructeurs et destructeur

4.24.2.1 MInt : :MInt (Node, float)

Constructeur.

Paramètres :

le graphe de l'image à segmenter

le paramètre de l'algorithme

mode bavard

fichier de résultat

Définition à la ligne 270 du fichier DisjointSet.cpp.

Références MInt().

Référencé par MInt().

La documentation de cette classe a été générée à partir des fichiers suivants :
- **DisjointSet.hpp**
- **DisjointSet.cpp**

4.25 Référence de la classe Node

Interface sur une classe de noeud.

`#include <Node.hpp>`

Fonctions membres publiques

- **Node** ()
  Constructeur par défaut.
- **Node** (const **Node** &)
  Constructeur de copie.
- **Node** & **operator=** (const **Node** &)
  Opérateur d'affectation.
- ~**Node** ()
  Destructeur.
- string **toString** (int, bool child=false, bool edge=false) const
  Retourne une chaîne de caractères détaillée.
- int **getRank** () const
  Retourne le rang du noeud.
- void **setRank** (int)
  Fixe le rang du noeud.
- **Pixel getPixel** () const
  Retourne le pixel du noeud.
- void **setPixel** (**Pixel**)
  Fixe le pixel du noeud.
- double **getWeight** ()
  Retourne le poids du noeuds.
- void **setWeight** (double)
  Fixe le poids du noeuds.
- **PixelMap** * **makeMap** (int) const
  Produit une carte d'un noeud.
- void **drawPNG** (string, int) const
  Produit un fichier **PNG**(p. 75) *du graphe.*
- void **drawBorderPNG** (string, int) const
  Produit un fichier **PNG**(p. 75) *des bords du graphe.*
- void **drawEPS** (string, int) const
  Produit un fichier PS du graphe.
- int **getNodeSize** ()
  Retourne le nombre d'enfants.
- int **getLeafSize** ()

Retourne le nombre de feuilles.

- int **getEdgeSize** ()
  Retourne le nombre d'arrêtes.
- void **sortEdge** ()
  Trie les arrêtes par leur poids.
- void **sortEdge** (float &min, float &max)
  Trie les arrêtes par leur poids et récupère les valeurs extrêmes.
- void **addNode** (**Node**)
  Ajoute un noeud.
- void **addEdge** (**Edge**)
  Ajoute une arrête.
- **Node_Iterator getNodeIterator** ()
  Retourne un itérateur des noeuds.
- **Edge_Iterator getEdgeIterator** ()
  Retourne un itérateur des arrêtes.
- void **merge** (**Node**)
  Fusionne avec un autre noeud.

Fonctions membres publiques statiques

- **Node CreateLeaf** (string="")
  Crée un noeud feuille.
- **Node CreateGraph** (string="")
  Crée un noeud graphe.
- **Node CreateGridGraph** (**PNG**, string="")
  Crée un grid graphe à partir d'une image.

Fonctions membres privées

- **Node** (**Node_Leaf** ∗)
  Constructeur privé.

Attributs privés

- **Node_Leaf** ∗ **p**
  Pointeur sur un noeud.

4.25.1 Description détaillée

Interface sur une classe de noeud.

Gère un compteur de références La pluspart des méthodes sont redirigées vers une instance héritées de Node_Base

Définition à la ligne 33 du fichier Node.hpp.

4.25.2 Documentation des fonctions membres

4.25.2.1 Node Node : :CreateGridGraph (PNG *image*, string *label* = "") `[static]`

Crée un grid graphe à partir d'une image.

Crée un grid graphe à partir d'une image

Définition à la ligne 24 du fichier Node.cpp.

Références addEdge(), addNode(), CreateGraph(), CreateGridGraph(), CreateLeaf(), Color : :distance(), Pixel : :getColor(), PNG : :getColor(), PNG : :getHeight(), getPixel(), PNG : :getWidth(), Pixel : :getX(), Pixel : :getY(), Edge : :setCoord(), setPixel(), et setWeight().

Référencé par CreateGridGraph().

4.25.2.2 void Node : :drawBorderPNG (string *fileName*, int *level*) const

Produit un fichier **PNG**(p. 75) des bords du graphe.

Produit un fichier **PNG**(p. 75) des bords du graphe

Définition à la ligne 185 du fichier Node.cpp.

Références PixelMap : :drawBorderPNG(), et drawBorderPNG().

Référencé par drawBorderPNG().

4.25.2.3 void Node : :drawEPS (string *fileName*, int *level*) const

Produit un fichier PS du graphe.

Produit un fichier PS C'est ici qu'on démarre la récursion

Définition à la ligne 196 du fichier Node.cpp.

Références drawEPS(), et EPSWriter : :write().

Référencé par drawEPS().

4.25.2.4 void Node : :drawPNG (string *fileName*, int *level*) const

Produit un fichier **PNG**(p. 75) du graphe.

Produit un fichier **PNG**(p. 75) C'est ici qu'on démarre la récursion

Définition à la ligne 169 du fichier Node.cpp.

Références drawPNG(), PNG_COLOR, et PNG : :write().

Référencé par drawPNG().

4.25.2.5 PixelMap $*$ Node : :makeMap (int *level*) const

Produit une carte d'un noeud.

Produit une carte

Définition à la ligne 150 du fichier Node.cpp.

Références makeMap().

Référencé par makeMap().

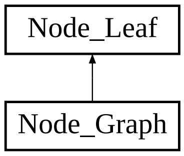
La documentation de cette classe a été générée à partir des fichiers suivants :
- **Node.hpp**
- **Node.cpp**

4.26 Référence de la classe Node_Graph

Noeud de type feuille.

#include <Node.hpp>

Graphe d'héritage de Node_Graph : :



Fonctions membres privées

- **Node_Graph** ()
  Constructeur privé.
- string **toString** (int, bool, bool) const
  Retourne une chaîne de caractères détaillée.
- void **_size** (int &, int &, int &, int &)
  Fonction auxilliaire pour le dessin.
- **PixelMap** ∗ **makeMap** (**PixelMap** ∗, int, int, int) const
  Produit une carte d'un noeud.
- void **drawPNG** (**PNG**, int, **Color**, int)
  Produit un fichier **PNG**(p. 75) *d'un noeud.*
- void **drawNodeEPS** (**EPSWriter** &, int, **Color**, int, int)
  Produit les noeuds pour un fichier PS.
- void **drawEdgeEPS** (**EPSWriter** &, int)
  Produit les arrêtes pour un fichier PS.
- int **getNodeSize** () const
  Retourne le nombre d'enfants.
- int **getLeafSize** () const
  Retourne le nombre de feuilles.
- int **getEdgeSize** () const
  Retourne le nombre d'arrête.
- void **sortEdge** ()
  Trie les arrête.
- void **sortEdge** (float &, float &)
  Trie les arrêtes et récupère les valeurs extrêmes.
- void **addNode** (**Node**)
  Ajoute un noeud.
- void **addEdge** (**Edge**)

Ajoute une arrête.

- void **merge** (**Node**)
  Fusionnne avec un autre noeud.
- **Node_Iterator getNodeIterator** () const
  Retourne un itérateur des noeuds.
- **Edge_Iterator getEdgeIterator** () const
  Retourne un itérateur des arrêtes.

Attributs privés

- vector< **Node** > **vNode**
  Les noeuds du graphe.
- vector< **Edge** > **vEdge**
  Les arrêtes du graphe.

4.26.1 Description détaillée

Noeud de type feuille.

Classe de base avec compteur de references

Définition à la ligne 174 du fichier Node.hpp.

4.26.2 Documentation des fonctions membres

4.26.2.1 void Node_Graph : :drawEdgeEPS (EPSWriter & *writer*, int *height*) `[private, virtual]`

Produit les arrêtes pour un fichier PS.

Production des arrêtes pour un fichier PS

Réimplémentée à partir de **Node_Leaf** (p. 66).

Définition à la ligne 447 du fichier Node.cpp.

Références drawEdgeEPS(), EPSWriter : :drawLine(), Edge : :getDST(), getEdgeIterator(), getEdgeSize(), Edge : :getSRC(), Edge : :getX0(), Edge : :getX1(), Edge : :getY0(), Edge : :getY1(), Edge_Iterator : :hasNext(), et Edge_Iterator : :next().

Référencé par drawEdgeEPS(), et drawNodeEPS().

4.26.2.2 void Node_Graph : :drawNodeEPS (EPSWriter & *writer*, int *level*, Color *c*, int *depth*, int *height*) `[private, virtual]`

Produit les noeuds pour un fichier PS.

On poursuit la récursion et on détermine la couleur du cluster

Réimplémentée à partir de **Node_Leaf** (p. 68).

Définition à la ligne 428 du fichier Node.cpp.

Références drawEdgeEPS(), Node_Leaf : :drawNodeEPS(), drawNodeEPS(), Pixel : :getColor(), getNodeIterator(), Node : :getPixel(), Node_Iterator : :hasNext(), Node_Iterator : :next(), et Node : :p.

Référencé par drawNodeEPS().

4.26.2.3 void Node_Graph : :drawPNG (PNG *image*, int *level*, Color *c*, int *depth*) [private, virtual]

Produit un fichier **PNG**(p. 75) d'un noeud.

On poursuit la récursion et on détermine la couleur du cluster

Réimplémentée à partir de **Node_Leaf** (p. 68).

Définition à la ligne 412 du fichier Node.cpp.

Références Node_Leaf : :drawPNG(), drawPNG(), Pixel : :getColor(), getNodeIterator(), Node : :getPixel(), Node_Iterator : :hasNext(), Node_Iterator : :next(), et Node : :p.

Référencé par drawPNG().

4.26.2.4 string Node_Graph : :toString (int *l*, bool *child*, bool *edge*) const [private, virtual]

Retourne une chaîne de caractères détaillée.

asdfjasdjféasd asdfljlasjdfléasd

Réimplémentée à partir de **Node_Leaf** (p. 66).

Définition à la ligne 324 du fichier Node.cpp.

Références Edge_Iterator : :hasNext(), Node_Iterator : :hasNext(), Edge_Iterator : :next(), Node_Iterator : :next(), Edge : :toString(), Node : :toString(), et toString().

Référencé par toString().

La documentation de cette classe a été générée à partir des fichiers suivants :
- **Node.hpp**
- **Node.cpp**

4.27 Référence de la classe Node_Iterator

Parcourt les noeuds d'un graphe.

`#include <Iterator.hpp>`

Fonctions membres publiques

- bool **hasNext** ()
  Y'a-t-il un prochain élément ?
- **Node next** ()
  Accède au prochain élément.

Types privés

- typedef vector< **Node** > : :const_iterator **itNode**
  Itérateur sur le vecteur de noeuds.

Fonctions membres privées

- **Node_Iterator** (const **Node_Graph** *)
  Constructeur à partir d'un noeud graphe.
- **Node_Iterator** (const **Node_Leaf** *)
  Constructeur à partir d'un noeud feuille.

Attributs privés

- **itNode current**
  itérateur du noeud courant
- **itNode end**
  itérateur de fin

4.27.1 Description détaillée

Parcourt les noeuds d'un graphe.

Les noeuds d'un graphe sont stockés dans une liste. Cette classe permet de parcourir les noeuds d'un graph sans connaissance de cette liste.

Définition à la ligne 14 du fichier Iterator.hpp.

4.27.2 Documentation des fonctions membres

4.27.2.1 bool Node_Iterator : :hasNext ()

Y'a-t-il un prochain élément ?

Renvoie :
vrai s'il y a encore un noeud

Définition à la ligne 17 du fichier Iterator.cpp.

Références current, et end.

Référencé par Node_Graph : :_size(), DisjointSet_Base : :DisjointSet_Base(), Node_-Graph : :drawNodeEPS(), Node_Graph : :drawPNG(), Node_Graph : :getLeafSize(), Node_-Graph : :makeMap(), et Node_Graph : :toString().

4.27.2.2 Node Node_Iterator : :next ()

Accède au prochain élément.

Renvoie :
le noeud suivant

Définition à la ligne 21 du fichier Iterator.cpp.

Références current.

Référencé par Node_Graph : :_size(), DisjointSet_Base : :DisjointSet_Base(), Node_-Graph : :drawNodeEPS(), Node_Graph : :drawPNG(), Node_Graph : :getLeafSize(), Node_-Graph : :makeMap(), et Node_Graph : :toString().

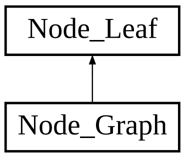
La documentation de cette classe a été générée à partir des fichiers suivants :
- **Iterator.hpp**
- **Iterator.cpp**

4.28 Référence de la classe Node_Leaf

Noeud de type feuille.

`#include <Node.hpp>`

Graphe d'héritage de Node_Leaf : :



Fonctions membres protégées

– virtual string **toString** (int level, bool child, bool edge) const
  Retourne une chaîne de caractères détaillée.

– int **getRank** () const
  Retourne le rang du noeud.

– void **setRank** (int)
  Fixe le rang du noeud.

– double **getWeight** () const
  Retourne le poids du noeud.

– void **setWeight** (double)
  Fixe le poids du noeud.

– **Pixel** **getPixel** () const
  Retourne le pixel du noeud.

– void **setPixel** (**Pixel**)
  Fixe le pixel du noeud.

– virtual **PixelMap** ∗ **makeMap** (**PixelMap** ∗, int, int, int) const
  Produit une carte d'un noeud.

– virtual void **drawPNG** (**PNG**, int, **Color**, int)
  Dessin des noeuds du graphe pour un fichier **PNG**(p. 75).

– virtual void **drawNodeEPS** (**EPSWriter** &, int, **Color**, int, int)
  Dessin des noeuds du graphe pour un fichier PS.

– virtual void **drawEdgeEPS** (**EPSWriter** &, int)
  Dessin des arrêtes d'une feuille pour un fichier PS.

– virtual int **getNodeSize** () const
  Retourne le nombre d'enfant : 1.

– virtual int **getLeafSize** () const
  Retourne le nombre de feuilles : 1.

– virtual int **getEdgeSize** () const

Retourne le nombre d'arrête : 0.

- virtual void **addNode** (**Node**)
  Ajoute un noeud : ne fait rien.
- virtual void **addEdge** (**Edge**)
  Ajoute une arrête : ne fait rien.
- virtual void **sortEdge** ()
  Trie les arrête : ne fait rien.
- virtual void **sortEdge** (float &, float &)
  Trie les arrête : ne fait rien.
- virtual void **merge** (**Node**)
  Fusionne avec un autre noeud : ne fait rien.
- virtual **Node_Iterator getNodeIterator** () const
  Retourne un itérateur des noeuds.
- virtual **Edge_Iterator getEdgeIterator** () const
  Retourne un itérateur des arrêtes.
- virtual void **_size** (int &, int &, int &, int &)
  Fonction auxilliaire pour le dessin.

Attributs protégés

- int **use**
  Compteur de références.
- int **rank**
  Rang.
- double **weight**
  Poids du noeud.
- **Pixel pixel**
  Pixel(p. 69).

Fonctions membres privées

- **Node_Leaf** ()
  Node_Leaf.

4.28.1 Description détaillée

Noeud de type feuille.

Classe de base avec compteur de references

Définition à la ligne 108 du fichier Node.hpp.

4.28.2 Documentation des fonctions membres

4.28.2.1 void Node_Leaf : :drawNodeEPS (EPSWriter & *writer*, int *level*, Color *c*, int, int *height*) `[protected, virtual]`

Dessin des noeuds du graphe pour un fichier PS.

Pour un noeud feuille on produit le pixel avec la couleur correspondante

Réimplémentée dans **Node_Graph** (p. 62).

Définition à la ligne 303 du fichier Node.cpp.

Références drawNodeEPS(), EPSWriter : :drawRound(), Pixel : :getColor(), Pixel : :getX(), Pixel : :getY(), pixel, et EPSWriter : :setColor().

Référencé par Node_Graph : :drawNodeEPS(), et drawNodeEPS().

4.28.2.2 void Node_Leaf : :drawPNG (PNG *image*, int *level*, Color *c*, int) `[protected, virtual]`

Dessin des noeuds du graphe pour un fichier **PNG**(p. 75).

Pour un noeud feuille on produit le pixel avec la couleur correspondante

Réimplémentée dans **Node_Graph** (p. 63).

Définition à la ligne 293 du fichier Node.cpp.

Références drawPNG(), Pixel : :getColor(), Pixel : :getX(), Pixel : :getY(), pixel, et PNG : :setColor().

Référencé par Node_Graph : :drawPNG(), et drawPNG().

La documentation de cette classe a été générée à partir des fichiers suivants :
- **Node.hpp**
- **Node.cpp**

4.29 Référence de la classe Pixel

Interface sur une classe **Pixel_Base**(p. 71).

`#include <Pixel.hpp>`

Fonctions membres publiques

- **Pixel** ()
  Constructeur par défaut.
- **Pixel** (int, int, **Color**)
  Constructeur à partir des coordonnées du pixel et sa couleur.
- **Pixel** (const **Pixel** &)
  Constructeur de copie.
- **Pixel** & **operator=** (const **Pixel** &)
  Operateur d'affectation.
- ~**Pixel** ()
  Destructeur.
- string **toString** () const
  Retourne une chaîne de caractères détaillée.
- int **getX** () const
  Retourne la coordonnée horizontale du pixel.
- int **getY** () const
  Retourne la coordonnée verticale du pixel.
- void **setColor** (**Color**)
  Fixe la couleur du pixel.
- **Color getColor** () const
  Retourne la couleur du pixel.

Attributs privés

- **Pixel_Base * p**
  Classe de base.

4.29.1 Description détaillée

Interface sur une classe **Pixel_Base**(p. 71).

Gère un compteur de références Les méthodes sont redirigées vers une instance de **Pixel_-Base**(p. 71)

Définition à la ligne 21 du fichier Pixel.hpp.

4.29.2 Documentation des contructeurs et destructeur

4.29.2.1 Pixel : :Pixel (int, int, Color)

Constructeur à partir des coordonnées du pixel et sa couleur.

Paramètres :

int coordonnée horizontale

int coordonnée verticale

Color couleur

Définition à la ligne 10 du fichier Pixel.cpp.

Références p.

La documentation de cette classe a été générée à partir des fichiers suivants :
- **Pixel.hpp**
- **Pixel.cpp**

4.30 Référence de la classe Pixel_Base

Un pixel.

`#include <Pixel.hpp>`

Fonctions membres privées

- **Pixel_Base** ()
  Constructeur par défaut.
- **Pixel_Base** (int _x, int _y, **Color** _color)
  Constructeur avec.
- string **toString** () const
  Retourne une chaîne de caractères détaillée.
- int **getX** () const
  Retourne la coordonnée horizontale du pixel.
- int **getY** () const
  Retourne la coordonnée verticale du pixel.
- void **setColor** (**Color** _color)
  Fixe la couleur du pixel.
- **Color getColor** () const
  Retourne la couleur du pixel.

Attributs privés

- int **use**
  Compteur de référence.
- **Color color**
  Couleur du pixel.
- int **x**
  Coordonnées du pixel.
- int **y**
  Coordonnées du pixel.

4.30.1 Description détaillée

Un pixel.

Représente un pixel d'une image avec ses coordonnées et sa couleur

Définition à la ligne 54 du fichier Pixel.hpp.

4.30.2 Documentation des contructeurs et destructeur

4.30.2.1 Pixel_Base : :Pixel_Base (int _ *x*, int _ *y*, Color _ *color*) `[inline, private]`

Constructeur avec.

Paramètres :

_ ***x*** coordonnée horizontale

_ ***y*** coordonnée verticale

_ ***color*** couleur du pixel

Définition à la ligne 68 du fichier Pixel.hpp.

La documentation de cette classe a été générée à partir des fichiers suivants :
- **Pixel.hpp**
- **Pixel.cpp**

4.31 Référence de la classe PixelMap

Labelise un tableau de pixel.

`#include <PixelMap.hpp>`

Fonctions membres publiques

- **PixelMap** (int, int)
  Constructeur à partir des dimensions.
- string **toString** ()
  Retourne une chaîne de caractères détaillée.
- void **addPixel** (**Pixel**)
  Ajoute un pixel.
- void **addPixel** (**Pixel**, int)
  Ajoute un pixel avec son label.
- **Node makeComponent** ()
  Produit un graphe à partir du tableau.
- void **drawBorderPNG** (string)
  Produit un fichier **PNG**(p. 75) *des bords des composant.*
- void **parentToFile** (string) const

Fonctions membres protégées

- void **findBorder** ()

Attributs privés

- vector< **Pixel** > **data**
  Les pixels.
- vector< int > **parent**
  Les labels.
- vector< bool > **border**
  Les bords.
- int **width**
  Largeur.
- int **height**
  Hauteur.

4.31.1 Description détaillée

Labelise un tableau de pixel.

Définition à la ligne 17 du fichier PixelMap.hpp.

4.31.2 Documentation des contructeurs et destructeur

4.31.2.1 PixelMap : :PixelMap (int, int)

Constructeur à partir des dimensions.

Paramètres :
 int largeur
 int hauteur

Définition à la ligne 7 du fichier PixelMap.cpp.

Références border, data, height, parent, PixelMap(), et width.

Référencé par PixelMap().

4.31.3 Documentation des fonctions membres

4.31.3.1 Node PixelMap : :makeComponent ()

Produit un graphe à partir du tableau.

Produit un graphe à partir d'un tableau de pixel On crée le graphe en effectuant une recherche en largeur

Définition à la ligne 31 du fichier PixelMap.cpp.

Références Node : :addEdge(), Node : :addNode(), data, Color : :distance(), Pixel : :getColor(), Node : :getEdgeSize(), Node : :getNodeSize(), Pixel : :getX(), Pixel : :getY(), height, parent, Trace : :print(), Edge : :setCoord(), Node : :setPixel(), et width.

Référencé par MeanShift : :makeNode(), et KMean : :makeNode().

La documentation de cette classe a été générée à partir des fichiers suivants :
- **PixelMap.hpp**
- **PixelMap.cpp**

4.32 Référence de la classe PNG

Interface sur une classe PNG concrète.

`#include <PNG.hpp>`

Fonctions membres publiques

- **PNG** (string)
  Ouvre un fichier PNG en lecture.
- **PNG** (int, int, **EPNG_TYPE**, string)
  Crée un fichier PNG.
- **PNG** (const **PNG** &)
  Constructeur de copie.
- **PNG** & **operator=** (const **PNG** &)
  Opérateur d'affectation.
- ~**PNG** ()
  Destructeur.
- **EPNG_TYPE getType** () const
  Retourne le type du fichier.
- string **toString** () const
  Retourne une chaîne de caractères détaillées.
- int **getWidth** () const
  Retourne la largeur de l'image en pixel.
- int **getHeight** () const
  Retourne la hauteur de l'image en pixel.
- **Color getColor** (int, int) const
  Retourne la couleur du pixel (i, j).
- void **setColor** (int, int, **Color**)
  Fixe la couleur du pixel (i, j).
- void **setColor** (int, int, float)
  Fixe le niveau de gris du pixel (i, j).
- void **setColor** (int, int, float, float, float)
  Fixe la couleur du pixel (i, j).
- void **read** ()
  Lit le fichier.
- void **write** ()
  Ecrit le fichier ;.
- **Histogram** * **getHistogram** () const
  Retourne une référence sur l'histogramme de l'image.

Attributs privés

– **PNG_Base * p**
  Pointeur sur l'image.

4.32.1 Description détaillée

Interface sur une classe PNG concrète.

Gère un compteur de références La pluspart des méthodes sont redirigées vers une instance héritées de **PNG_Base**(p. 77)

Définition à la ligne 45 du fichier PNG.hpp.

4.32.2 Documentation des contructeurs et destructeur

4.32.2.1 PNG : :PNG (int, int, EPNG_TYPE, string)

Crée un fichier PNG.

Paramètres :
  int largeur en pixel
  int hauteur en pixel
  type type de fichier
  string nom du fichier

Définition à la ligne 10 du fichier PNG.cpp.

Références p, et PNG().

4.32.3 Documentation des fonctions membres

4.32.3.1 Histogram * PNG : :getHistogram () const

Retourne une référence sur l'histogramme de l'image.

Paramètres :
  int le nombre classes à construir (**KMean**(p. 43))

Définition à la ligne 66 du fichier PNG.cpp.

Références PNG_Base : :getHistogram(), et p.

Référencé par KMean : :init().

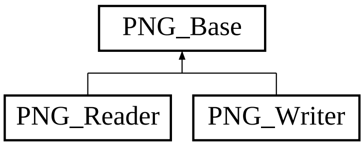
La documentation de cette classe a été générée à partir des fichiers suivants :
– **PNG.hpp**
– **PNG.cpp**

4.33 Référence de la classe PNG_Base

Classe de base abstraite pour les 2 type de **PNG**(p. 75) : lecture et écriture.

`#include <PNG.hpp>`

Graphe d'héritage de PNG_Base : :



Fonctions membres protégées

– **PNG_Base** ()
  Constructeur.

– ~**PNG_Base** ()
  Desctructeur.

– string **toString** () const
  Retourne une chaîne de caractères détaillées.

– int **getWidth** () const
  Retourne la largeur de l'image.

– int **getHeight** () const
  Retourne la hauteur de l'image.

– **EPNG_TYPE getType** () const
  Retourne le type de l'image.

– virtual **Color getColor** (int, int) const =0
  Retourne la couleur du pixel (i, j).

– virtual void **setColor** (int, int, **Color**)=0
  Fixe la couleur du pixel (i, j).

– virtual void **setColor** (int, int, float)=0
  Fixe le niveau de gris du pixel (i, j).

– virtual void **setColor** (int, int, float, float, float)=0
  Fixe la couleur du pixel (i, j).

– virtual void **read** ()=0
  Lit le fichier.

– virtual void **write** ()=0
  Ecrit le fichier.

– **Histogram** ∗ **getHistogram** () const
  Retourne l'histogramme de l'image.

Attributs protégés

– int **width**
Largeur et hauteur de l'image.

– int **height**
Largeur et hauteur de l'image.

– **EPNG_TYPE type**
Mode du format **PNG**(p. 75).

– string **fileName**
Nom du fichier.

– FILE * **fp**
Descripteur du fichier.

– png_structp **png_ptr**
Structure **PNG**(p. 75).

– png_infop **info_ptr**
Info **PNG**(p. 75).

– png_byte **color_type**
Mode de couleur.

– png_byte **bit_depth**
Profondeur.

– int **rowbytes**
Largeur d'une lign.

– png_bytep * **pData**
Données.

Attributs privés

– int **use**
Compteur de références.

4.33.1 Description détaillée

Classe de base abstraite pour les 2 type de **PNG**(p. 75) : lecture et écriture.

C'est cette classe qui possède le compteur de référence

Définition à la ligne 92 du fichier PNG.hpp.

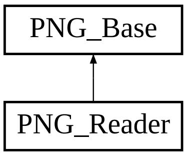
La documentation de cette classe a été générée à partir des fichiers suivants :
– **PNG.hpp**
– **PNG.cpp**

4.34 Référence de la classe PNG_Reader

Lecture d'une image au format **PNG**(p. 75).

`#include <PNG.hpp>`

Graphe d'héritage de PNG_Reader : :



Fonctions membres privées

- **PNG_Reader** (string _fileName)
  Constructeur.
- **Color getColor** (int, int) const
  Retourne la couleur du pixel (i, j).
- void **setColor** (int, int, **Color**)
  Fixe la couleur du pixel (i, j).
- void **setColor** (int, int, float)
  Fixe le niveau de gris du pixel (i, j).
- void **setColor** (int, int, float, float, float)
  Fixe la couleur du pixel (i, j).
- void **read** ()
  Lit le fichier.
- void **write** ()
  Ecrit le fichier.

4.34.1 Description détaillée

Lecture d'une image au format **PNG**(p. 75).

Définition à la ligne 150 du fichier PNG.hpp.

4.34.2 Documentation des contructeurs et destructeur

4.34.2.1 PNG_Reader : :PNG_Reader (string _*fileName*) `[inline, private]`

Constructeur.

Paramètres :
string le nom du fichier

Définition à la ligne 154 du fichier PNG.hpp.

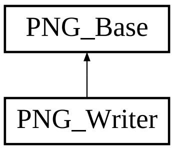
La documentation de cette classe a été générée à partir des fichiers suivants :
- **PNG.hpp**
- **PNG.cpp**

4.35 Référence de la classe PNG_Writer

Création d'un fichier au format **PNG**(p. 75).

```
#include <PNG.hpp>
```

Graphe d'héritage de PNG_Writer : :



Fonctions membres privées

- png_bytep * **make_pixels** ()
  Allocation de mémoire.
- void **_setColor** (int, int, float)
  Fixe le niveau de gris du pixel (i, j).
- void **_setColor** (int, int, float, float, float)
  Fixe la couleur du pixel (i, j).
- void **_setColor** (int, int, **Color**)
  Ecriture du buffer vers l'image.
- **PNG_Writer** (int, int, **EPNG_TYPE**, string)
  Crée un fichier **PNG**(p. 75).
- **Color getColor** (int, int) const
  Retourne la couleur du buffer en (i, j).
- void **setColor** (int, int, **Color**)
  Fixe la couleur du buffer en (i, j).
- void **setColor** (int, int, float)
  Fixe le niveau de gris du pixel (i, j).
- void **setColor** (int, int, float, float, float)
  Fixe la couleur du pixel (i, j).
- void **read** ()
  Lit le fichier.
- void **write** ()
  Ecrit le fichier.

Attributs privés

- vector< **Color** > **buffer**
  Buffer de couleurs.

4.35.1 Description détaillée

Création d'un fichier au format **PNG**(p. 75).

Définition à la ligne 170 du fichier PNG.hpp.

4.35.2 Documentation des contructeurs et destructeur

4.35.2.1 PNG_Writer : :PNG_Writer (int, int, EPNG_TYPE, string) `[private]`

Crée un fichier **PNG**(p. 75).

Paramètres :

int largeur en pixel

int hauteur en pixel

type type de fichier

string nom du fichier

Définition à la ligne 195 du fichier PNG.cpp.

Références buffer, make_pixels(), PNG_COLOR, PNG_GRAY, et PNG_Writer().

Référencé par PNG_Writer().

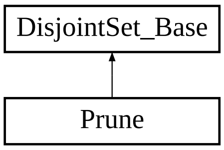
La documentation de cette classe a été générée à partir des fichiers suivants :
- **PNG.hpp**
- **PNG.cpp**

4.36 Référence de la classe Prune

Elagage d'un graphe.

`#include <DisjointSet.hpp>`

Graphe d'héritage de Prune : :



Fonctions membres publiques

- **Prune** (**Node**, int)
  Constructeur.
- **Node makeDisjointSet** ()
  Implémente l'algorithme.

Attributs privés

- int **area**
  Compteur de fusion.

4.36.1 Description détaillée

Elagage d'un graphe.

Fusion des régions plus petites qu'un nombre

Définition à la ligne 131 du fichier DisjointSet.hpp.

4.36.2 Documentation des contructeurs et destructeur

4.36.2.1 Prune : :Prune (Node, int)

Constructeur.

Paramètres :
Node le graphe de l'image à segmenter
int taille minimale

Définition à la ligne 540 du fichier DisjointSet.cpp.

Références Prune().

Référencé par Prune().

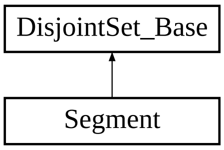
La documentation de cette classe a été générée à partir des fichiers suivants :
- **DisjointSet.hpp**
- **DisjointSet.cpp**

4.37 Référence de la classe Segment

Segment : résultat de plusieurs test.

```
#include <DisjointSet.hpp>
```

Graphe d'héritage de Segment : :



Fonctions membres publiques

- **Segment** (**Node**, float)
  Constructeur.
- **Node makeDisjointSet** ()
  Implémente l'algorithme.

Attributs privés

- float **k**
  Paramètre de l'algorithme.
- float **edgeWeightMin**
  Arrête de poids minimal.
- float **edgeWeightMax**
  Arrête de poids maximal.
- float **edgeWeightDelta**
  Différence maximal.

4.37.1 Description détaillée

Segment : résultat de plusieurs test.

Variation de l'algorithme de l'article pour segmenter une image

Définition à la ligne 94 du fichier DisjointSet.hpp.

4.37.2 Documentation des contructeurs et destructeur

4.37.2.1 Segment : :Segment (Node, float)

Constructeur.

Paramètres :
Node graphe de l'image à segmenter

float paramètre de l'algorithme

Définition à la ligne 339 du fichier DisjointSet.cpp.

Références Segment().

Référencé par Segment().

La documentation de cette classe a été générée à partir des fichiers suivants :
- **DisjointSet.hpp**
- **DisjointSet.cpp**

4.38 Référence de la classe Trace

Représente une trace.

`#include <Debug.hpp>`

Fonctions membres publiques

- **Trace** (const char ∗_name, **Channel** &_pChannel, bool _display=true)
  Constructeur d'une trace.
- void **print** (const char ∗txt)
  Affiche un message dans la trace (C style).
- void **print** (const string txt)
  Affiche un message dans la trace (STL style).

Attributs privés

- **Channel ∗ pChannel**
  Flot de sortie.
- const char ∗ **name**
  Nom de la trace.
- bool **display**
  Affiche les messages.
- time_t **start**
  Chronomètre.
- time_t **end**
  Chronomètre.

4.38.1 Description détaillée

Représente une trace.

L'utilisation typique est de créer une instance de Trace à l'entrée d'une fonction et loguer des informations dans cette trace.

Définition à la ligne 48 du fichier Debug.hpp.

4.38.2 Documentation des contructeurs et destructeur

4.38.2.1 Trace : :Trace (const char ∗ *_name*, Channel & *_pChannel*, bool *_display* = `true`) `[inline]`

Constructeur d'une trace.

Paramètres :
  _name : nom à afficher avec un message

_pChannel : un flot de sortie

_display : affiche les messages ou non

Définition à la ligne 62 du fichier Debug.hpp.

Références display, name, pChannel, start, et Channel : :traceFile.

La documentation de cette classe a été générée à partir du fichier suivant :
– **Debug.hpp**

Chapitre 5

segment Documentation des fichiers

5.1 Référence du fichier Color.cpp

Implémentation des classes utiles pour la gestion des couleurs.

`#include "Color.hpp"`

5.1.1 Description détaillée

Implémentation des classes utiles pour la gestion des couleurs.

Définition dans le fichier **Color.cpp**.

5.2 Référence du fichier Color.hpp

Définition des classes utiles pour la gestion des couleurs.

#include <string>

#include <sstream>

#include <cmath>

#include <stdexcept>

#include "Debug.hpp"

Classes

- class **Color**
  Interface sur une classe de couleur concrète.
- class **Color_Base**
  Classe de base abstraite pour les 2 type de couleurs : gris et couleurs.
- class **Color_Gray**
  Classe pour les couleurs en niveaux de gris.
- class **Color_Color**
  Classe pour les couleurs.

Énumérations

- enum **ECOLOR_TYPE** { **COLOR_EMPTY**, **COLOR_GRAY**, **COLOR_COLOR** }
  Type de couleur.

Variables

- const double **Pi** = 3.141592
  Une constante bien connue...

5.2.1 Description détaillée

Définition des classes utiles pour la gestion des couleurs.

Définition dans le fichier **Color.hpp**.

5.2.2 Documentation du type de l'énumération

5.2.2.1 enum ECOLOR_TYPE

Type de couleur.

3 type de couleurs : vide pour des besoins d'implémenation, gris et couleur

Définition à la ligne 20 du fichier Color.hpp.

Référencé par Color : :getType().

5.3 Référence du fichier Debug.hpp

Définition des classes utiles pour le debug.

```
#include <iostream>
#include <fstream>
#include <string>
#include <iomanip>
#include <time.h>
```

Namespaces

– namespace **std**

Classes

– class **Channel**
Représente un canal de sortie.

– class **Trace**
Représente une trace.

5.3.1 Description détaillée

Définition des classes utiles pour le debug.

Définition dans le fichier **Debug.hpp**.

5.4 Référence du fichier DisjointSet.cpp

Implémentation des classes utiles pour le clustering de classes disjointes.

```
#include "DisjointSet.hpp"
```

5.4.1 Description détaillée

Implémentation des classes utiles pour le clustering de classes disjointes.

Définition dans le fichier **DisjointSet.cpp**.

5.5 Référence du fichier DisjointSet.hpp

Définition des classes utiles pour le clustering de classes disjointes.

`#include <vector>`

`#include "Debug.hpp"`

`#include "Node.hpp"`

`#include <map>`

Classes

- class **DisjointSet_Base**
  Classe de base pour les algorithmes utilisant des ensembles disjoints.
- class **Kruskal**
  Kruskal : classe de test.
- class **MInt**
  MInt : classe de l'article.
- class **Segment**
  Segment : résultat de plusieurs test.
- class **MeanShiftFusion**
  Meanshift.
- class **Prune**
  Elagage d'un graphe.

5.5.1 Description détaillée

Définition des classes utiles pour le clustering de classes disjointes.

Définition dans le fichier **DisjointSet.hpp**.

5.6 Référence du fichier Edge.cpp

Implémentation de la classe **Edge**(p. 23).

```
#include "Edge.hpp"
```

Fonctions

- bool **compareEdge** (const **Edge** a, const **Edge** b)
  Compare le poids de 2 arrêtes.

5.6.1 Description détaillée

Implémentation de la classe **Edge**(p. 23).

Classe : **Edge**(p. 23)

Définition dans le fichier **Edge.cpp**.

5.6.2 Documentation des fonctions

5.6.2.1 bool compareEdge (const Edge *a*, const Edge *b*)

Compare le poids de 2 arrêtes.

Paramètres :
  a une arrête
  b une arrête

Renvoie :
  vrai si le poids de a est plus petit que celui de b

Définition à la ligne 109 du fichier Edge.cpp.

Références Edge : :getWeight().

Référencé par Node_Graph : :sortEdge().

5.7 Référence du fichier Edge.hpp

Définition de la classe **Edge**(p. 23).

`#include "Node.hpp"`

Classes

- class **Edge**
  Interface sur une classe **Edge_Base**(p. 25).
- class **Edge_Base**
  Arrete entre noeuds.

Fonctions

- bool **compareEdge** (const **Edge**, const **Edge**)
  Compare le poids de 2 arrêtes.

5.7.1 Description détaillée

Définition de la classe **Edge**(p. 23).

Définition dans le fichier **Edge.hpp**.

5.7.2 Documentation des fonctions

5.7.2.1 bool compareEdge (const Edge *a*, const Edge *b*)

Compare le poids de 2 arrêtes.

Paramètres :
  a une arrête
  b une arrête

Renvoie :
  vrai si le poids de a est plus petit que celui de b

Définition à la ligne 109 du fichier Edge.cpp.

Références Edge : :getWeight().

Référencé par Node_Graph : :sortEdge().

5.8 Référence du fichier EPSWriter.cpp

Implémentation de la classe **EPSWriter**(p. 29) pour générer des fichiers PostScript.

```
#include "EPSWriter.hpp"
```

5.8.1 Description détaillée

Implémentation de la classe **EPSWriter**(p. 29) pour générer des fichiers PostScript.

Définition dans le fichier **EPSWriter.cpp**.

5.9 Référence du fichier EPSWriter.hpp

Définition de la classe **EPSWriter**(p. 29) pour générer des fichiers PostScript.

```
#include <string>
#include <fstream>
#include "Color.hpp"
```

Classes

- class **EPSWriter**
  Générateur de fichier PostScript.

5.9.1 Description détaillée

Définition de la classe **EPSWriter**(p. 29) pour générer des fichiers PostScript.

Définition dans le fichier **EPSWriter.hpp**.

5.10 Référence du fichier Filter.cpp

Implémentation des classes filtres.

```
#include "Filter.hpp"
```

5.10.1 Description détaillée

Implémentation des classes filtres.

Définition dans le fichier **Filter.cpp**.

5.11 Référence du fichier Filter.hpp

Définition des classes filtres.

```
#include <math.h>
#include <string>
#include <iostream>
#include <sstream>
#include "Debug.hpp"
#include "PNG.hpp"
```

Classes

- class **Filter_Base**
  Classe de base pour les filtres.
- class **Filter_Gaussian**
  Filtre gaussien.

5.11.1 Description détaillée

Définition des classes filtres.

Définition dans le fichier **Filter.hpp**.

5.12 Référence du fichier FVector.cpp

Implémentation de la classe **FVector**(p. 35).

`#include "FVector.hpp"`

Fonctions

- float **norm** (const **FVector** &a, const **FVector** &b)
  Calcule la norme euclidienne de 2 vecteurs.
- float **normColor** (const **FVector** &a, const **FVector** &b)
  Calcule la norme euclidienne entre les couleurs de 2 vecteur.
- float **norm** (const **FVector_Base** &a, const **FVector_Base** &b)
  Calcule la norme euclidienne de 2 vecteurs.
- float **normColor** (const **FVector_Base** &a, const **FVector_Base** &b)
  Calcule la norme euclidienne entre les couleurs de 2 vecteur.

5.12.1 Description détaillée

Implémentation de la classe **FVector**(p. 35).

Définition dans le fichier **FVector.cpp**.

5.12.2 Documentation des fonctions

5.12.2.1 float norm (const FVector_Base &, const FVector_Base &)

Calcule la norme euclidienne de 2 vecteurs.

Paramètres :

FVector_Base& 1er vecteur

FVector_Base& 2ème vecteur

Définition à la ligne 127 du fichier FVector.cpp.

Référencé par MeanShiftWindow : :computeMean(), et norm().

5.12.2.2 float norm (const FVector &, const FVector &)

Calcule la norme euclidienne de 2 vecteurs.

Paramètres :

FVector& 1er vecteur

FVector& 2ème vecteur

Définition à la ligne 54 du fichier FVector.cpp.

Référencé par MeanShift : :findMode().

5.12.2.3 float normColor (const FVector_Base &, const FVector_Base &)

Calcule la norme euclidienne entre les couleurs de 2 vecteur.

Paramètres :
- ***FVector_Base*** 1er vecteur
- ***FVector_Base*** 2ème vecteur

Définition à la ligne 135 du fichier FVector.cpp.

Référencé par MeanShift : :findMode(), et normColor().

5.12.2.4 float normColor (const FVector &, const FVector &)

Calcule la norme euclidienne entre les couleurs de 2 vecteur.

Paramètres :
- ***FVector&*** 1er vecteur
- ***FVector&*** 2ème vecteur

Définition à la ligne 57 du fichier FVector.cpp.

5.13 Référence du fichier FVector.hpp

Définition de la classe **FVector**(p. 35).

`#include "Debug.hpp"`

`#include "Pixel.hpp"`

`#include <vector>`

Classes

- class **FVector**
  Interface sur un **FVector_Base**(p. 38).
- class **FVector_Base**
  Classe de base abstraite pour les **FVector**(p. 35).

Fonctions

- float **norm** (const **FVector** &, const **FVector** &)
  Calcule la norme euclidienne de 2 vecteurs.
- float **normColor** (const **FVector** &, const **FVector** &)
  Calcule la norme euclidienne entre les couleurs de 2 vecteur.
- float **norm** (const **FVector_Base** &, const **FVector_Base** &)
  Calcule la norme euclidienne de 2 vecteurs.
- float **normColor** (const **FVector_Base** &, const **FVector_Base** &)
  Calcule la norme euclidienne entre les couleurs de 2 vecteur.

5.13.1 Description détaillée

Définition de la classe **FVector**(p. 35).

Définition dans le fichier **FVector.hpp**.

5.13.2 Documentation des fonctions

5.13.2.1 float norm (const FVector_Base &, const FVector_Base &)

Calcule la norme euclidienne de 2 vecteurs.

Paramètres :

FVector_Base& 1er vecteur

FVector_Base& 2ème vecteur

Définition à la ligne 127 du fichier FVector.cpp.

Référencé par MeanShiftWindow : :computeMean(), et norm().

5.13.2.2 float norm (const FVector &, const FVector &)

Calcule la norme euclidienne de 2 vecteurs.

Paramètres :

FVector& 1er vecteur

FVector& 2ème vecteur

Définition à la ligne 54 du fichier FVector.cpp.

Références norm(), et FVector : :p.

5.13.2.3 float normColor (const FVector_Base &, const FVector_Base &)

Calcule la norme euclidienne entre les couleurs de 2 vecteur.

Paramètres :

FVector_Base 1er vecteur

FVector_Base 2ème vecteur

Définition à la ligne 135 du fichier FVector.cpp.

Référencé par MeanShift : :findMode(), et normColor().

5.13.2.4 float normColor (const FVector &, const FVector &)

Calcule la norme euclidienne entre les couleurs de 2 vecteur.

Paramètres :

FVector& 1er vecteur

FVector& 2ème vecteur

Définition à la ligne 57 du fichier FVector.cpp.

Références normColor(), et FVector : :p.

5.14 Référence du fichier Histogram.cpp

Définition de la classe **Histogram**(p. 40).

```
#include "Histogram.hpp"
```

5.14.1 Description détaillée

Définition de la classe **Histogram**(p. 40).

Définition dans le fichier **Histogram.cpp**.

5.15 Référence du fichier Histogram.hpp

Définition de la classe **Histogram**(p. 40).

```
#include <vector>
#include <map>
#include <sstream>
#include "Debug.hpp"
#include "Pixel.hpp"
#include "Color.hpp"
```

Classes

- class **HistogramClass**
  HistogramClass.
- class **Histogram**
  Histogram.

5.15.1 Description détaillée

Définition de la classe **Histogram**(p. 40).

Définition dans le fichier **Histogram.hpp**.

5.16 Référence du fichier Iterator.cpp

Implémentation des classe **Node_Iterator**(p. 64) et **Edge_Iterator**(p. 27).

```
#include "Iterator.hpp"
```

5.16.1 Description détaillée

Implémentation des classe **Node_Iterator**(p. 64) et **Edge_Iterator**(p. 27).

Classe : **Node_Iterator**(p. 64)

Définition dans le fichier **Iterator.cpp**.

5.17 Référence du fichier Iterator.hpp

Définition des classe **Node_Iterator**(p. 64) et **Edge_Iterator**(p. 27).

`#include "Node.hpp"`

Classes

- class **Node_Iterator**
  Parcourt les noeuds d'un graphe.
- class **Edge_Iterator**
  Parcourt les arrêtes d'un graphe.

5.17.1 Description détaillée

Définition des classe **Node_Iterator**(p. 64) et **Edge_Iterator**(p. 27).

Définition dans le fichier **Iterator.hpp**.

5.18 Référence du fichier KMean.cpp

Implémentation des classes pour les **KMean**(p. 43).

`#include "KMean.hpp"`

5.18.1 Description détaillée

Implémentation des classes pour les **KMean**(p. 43).

KMeanClass(p. 45) : représente un centroïde

Définition dans le fichier **KMean.cpp**.

5.19 Référence du fichier KMean.hpp

Définition des classes pour les **KMean**(p. 43).

```
#include <vector>
#include <list>
#include "Debug.hpp"
#include "PNG.hpp"
#include "PixelMap.hpp"
```

Classes

- class **KMeanClass**
  KMeanClass : représente un centroïde.
- class **KMean**
  Implémente l'algorithmes des K-Means.

5.19.1 Description détaillée

Définition des classes pour les **KMean**(p. 43).

Définition dans le fichier **KMean.hpp**.

5.20 Référence du fichier MeanShift.cpp

Implémentation des classes pour le Meanshift.

```
#include "MeanShift.hpp"
```

5.20.1 Description détaillée

Implémentation des classes pour le Meanshift.

Définition dans le fichier **MeanShift.cpp**.

5.21 Référence du fichier MeanShift.hpp

Définition des classes pour le Meanshift.

```
#include "Debug.hpp"
#include "Pixel.hpp"
#include "PNG.hpp"
#include "PixelMap.hpp"
#include "FVector.hpp"
```

Classes

- class **Lattice**
  Représente une zone rectangulaire de vecteurs de Features.
- class **MeanShiftWindow**
  Représente la fenêtre du Meanshift.
- class **MeanShift**
  Calcul du filtre Meanshift.

Énumérations

- enum **ELATTICE_STATE** { **ELATTICE_NO_MODE**, **ELATTICE_HAS_MODE**, **ELATTICE_CANDIDATE_MODE** }
  Etat des points visités par le filtre Meanshift.

5.21.1 Description détaillée

Définition des classes pour le Meanshift.

Définition dans le fichier **MeanShift.hpp**.

5.22 Référence du fichier Node.cpp

Implémentation de la classe **Node**(p. 57).

```
#include "Node.hpp"
```

5.22.1 Description détaillée

Implémentation de la classe **Node**(p. 57).

Définition dans le fichier **Node.cpp**.

5.23 Référence du fichier Node.hpp

Définition de la classe **Node**(p. 57).

```
#include <string>
#include <list>
#include <iostream>
#include <sstream>
#include <iomanip>
#include <fstream>
#include <assert.h>
#include "Debug.hpp"
#include "PNG.hpp"
#include "EPSWriter.hpp"
#include "Edge.hpp"
#include "Iterator.hpp"
#include "PixelMap.hpp"
```

Classes

– class **Node**
  Interface sur une classe de noeud.

– class **Node_Leaf**
  Noeud de type feuille.

– class **Node_Graph**
  Noeud de type feuille.

5.23.1 Description détaillée

Définition de la classe **Node**(p. 57).

Définition dans le fichier **Node.hpp**.

5.24 Référence du fichier Pixel.cpp

Définition de la classe **Pixel**(p. 69).

```
#include "Pixel.hpp"
```

5.24.1 Description détaillée

Définition de la classe **Pixel**(p. 69).

Définition dans le fichier **Pixel.cpp**.

5.25 Référence du fichier Pixel.hpp

Définition de la classe **Pixel**(p. 69).

```
#include <string>
#include "Debug.hpp"
#include "Color.hpp"
```

Classes

- class **Pixel**
  Interface sur une classe **Pixel_Base**(p. 71).
- class **Pixel_Base**
  Un pixel.

5.25.1 Description détaillée

Définition de la classe **Pixel**(p. 69).

Définition dans le fichier **Pixel.hpp**.

5.26 Référence du fichier PixelMap.cpp

Implémentation de la classe **PixelMap**(p. 73).

```
#include "PixelMap.hpp"
```

5.26.1 Description détaillée

Implémentation de la classe **PixelMap**(p. 73).

Définition dans le fichier **PixelMap.cpp**.

5.27 Référence du fichier PixelMap.hpp

Définition de la classe **PixelMap**(p. 73).

```
#include <vector>
#include <string>
#include "PNG.hpp"
#include "Node.hpp"
```

Classes

– class **PixelMap**
  Labelise un tableau de pixel.

5.27.1 Description détaillée

Définition de la classe **PixelMap**(p. 73).

Définition dans le fichier **PixelMap.hpp**.

5.28 Référence du fichier PNG.cpp

Implémentation des classes permettant de manipuler des fichiers au format **PNG**(p. 75).

```
#include "PNG.hpp"
```

5.28.1 Description détaillée

Implémentation des classes permettant de manipuler des fichiers au format **PNG**(p. 75).

Définition dans le fichier **PNG.cpp**.

5.29 Référence du fichier PNG.hpp

Définition des classes permettant de manipuler des fichiers au format **PNG**(p. 75).

```
#include <string>
#include <setjmp.h>
#include <png.h>
#include <stdio.h>
#include <iostream>
#include <stdexcept>
#include <sstream>
#include "Debug.hpp"
#include "Histogram.hpp"
```

Classes

- class **PNG**
  Interface sur une classe PNG concrète.
- class **PNG_Base**
  Classe de base abstraite pour les 2 type de **PNG**(p. 75) *: lecture et écriture.*
- class **PNG_Reader**
  Lecture d'une image au format **PNG**(p. 75).
- class **PNG_Writer**
  Création d'un fichier au format **PNG**(p. 75).

Énumérations

- enum **EPNG_TYPE** {
  PNG_EMPTY, **PNG_GRAY**, **PNG_COLOR**, **PNG_PALET_COLOR**, **PNG_GRAY_ALPHA**, **PNG_COLOR_ALPHA** }
  Mode de fichier du format **PNG**(p. 75).

5.29.1 Description détaillée

Définition des classes permettant de manipuler des fichiers au format **PNG**(p. 75).

Définition dans le fichier **PNG.hpp**.

5.29.2 Documentation du type de l'énumération

5.29.2.1 enum EPNG_TYPE

Mode de fichier du format **PNG**(p. 75).

Valeurs énumérées :

PNG_EMPTY Pour raison d'implémentation.

PNG_GRAY Niveau de gris.

PNG_COLOR Couleur RGB.

PNG_PALET_COLOR Palette de couleurs : NON SUPPORTE.

PNG_GRAY_ALPHA Niveau de gris et transparence : NON SUPPORTE.

PNG_COLOR_ALPHA Couleur RGB et transparence : NON SUPPORTE.

Définition à la ligne 23 du fichier PNG.hpp.

Référencé par PNG_Base : :getType(), et PNG : :getType().

5.30 Référence du fichier segment.cpp

Programme de segmentations.

```
#include "segment.hpp"
```

Fonctions

- int **main** (int argc, char ∗argv[ ])
- void **segmentHelp** ()
- void **segmentOption** (int argc, char ∗argv[ ], map< string, vector< string > > &args)

5.30.1 Description détaillée

Programme de segmentations.

Définition dans le fichier **segment.cpp**.

5.31 Référence du fichier segment.hpp

Définition de fonctions utilitaires pour le programme segment.

```
#include <string>
#include <vector>
#include <iostream>
#include <stdexcept>
#include <map>
#include "DisjointSet.hpp"
#include "Node.hpp"
#include "Filter.hpp"
#include "KMean.hpp"
#include "MeanShift.hpp"
```

Fonctions

- void **segmentHelp** ()
- void **segmentOption** (int, char *[], map< string, vector< string > > &)

5.31.1 Description détaillée

Définition de fonctions utilitaires pour le programme segment.

Définition dans le fichier **segment.hpp**.

Index