Traitement d'image : Détection des buts

Généré par Doxygen 1.8.13

Table des matières

Chapitre 1

Liste des éléments obsolètes

Classe cPoint

Membre grassProcessing (const Mat grass, Mat &dst)

Membre jarvis (const Mat grass, int size)

Membre jarvisSlave (cPoint points[], int n, const Mat grass)

Membre orientation (cPoint p, cPoint q, cPoint r)

Membre zoneColor (const Mat grass, Mat &current, int color_pixel, int x, int y)

Chapitre 2

Index des classes

2.1 Liste des classes

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

Chapitre 3

Index des fichiers

3.1 Liste des fichiers

Liste de tous les fichiers documentés avec une brève description :

Chapitre 4

Documentation des classes

4.1 Référence de la structure cPoint

Attributs publics

- int **x**
- int y

4.1.1 Description détaillée

Structure permettant de définir un point

Obsolète

4.1.2 Documentation des données membres

4.1.2.1 y

```
int cPoint::y
```

Entier correspondant à l'abscisse du point

La documentation de cette structure a été générée à partir du fichier suivant :

- postDetection.cpp

Chapitre 5

Documentation des fichiers

5.1 Référence du fichier postDetection.cpp

```
#include <iostream>
#include <cstdlib>
#include <cstdio>
#include <dirent.h>
#include <opencv2/opencv.hpp>
#include <fstream>
#include <ctime>
#include <list>
```

Classes

- struct cPoint

Macros

- #define param 2

Fonctions

- RNG rng (12345)
- int hasValue (Mat image)
- Mat getPixelMatrix (Mat image, int cols, int rows, int radius)
- void patchAll3 (const Mat src1, const Mat src2, const Mat src3, Mat &dst, int taille_patch)
- void contoursTerrain (const Mat src, Mat &dst)
- void detectionGrass (const Mat im, Mat &dst)
- void init (String folder)
- bool tooMuchGrass (const Mat grass)
- int getNextMatrix (Mat &M)
- void contourBlob (Mat src, Mat originale)
- void progressBar ()
- void process (String benchmark)
- void usage (const char ∗s)
- int main (int argc, char ∗argv[ ])
- int zoneColor (const Mat grass, Mat &current, int color_pixel, int x, int y)
- int orientation (cPoint p, cPoint q, cPoint r)
- void jarvisSlave (cPoint points[ ], int n, const Mat grass)
- void jarvis (const Mat grass, int size)
- void grassProcessing (const Mat grass, Mat &dst)

Variables

- float progress = 0.0
- vector< Mat > images
- unsigned int image_counter
- int bordure = 5
- bool **bench**

5.1.1 Documentation des macros

5.1.1.1 param

```
#define param 2
```

Nombre de paramètres attendus lors de l'appel du programme

5.1.2 Documentation des fonctions

5.1.2.1 contourBlob()

```
void contourBlob (
            Mat src,
            Mat originale )
```

Cette fonction va prendre une image binaire passé en paramètre et va chercher les zones blanches qui correspondent à la base des poteaux détectés. Elle va ensuite entourer ces zones avec des cercles rouges sur l'image d'origine

Paramètres

src	Image binaire représantant les zones ou sont detectés le ou les poteaux des buts
originale	Image originale ou seront dessiné les cercles

5.1.2.2 contoursTerrain()

```
void contoursTerrain (
            const Mat src,
            Mat & dst )
```

Cette fonction va chercher les contours du terrain

Paramètres

src	Matrice correspondant à l'image dont on veut extraire les contours du terrain
dst	Image permettant d'afficher le contour du terrain

5.1.2.3 detectionGrass()

```
void detectionGrass (
            const Mat im,
            Mat & dst )
```

Cette fonction va chercher l'ensemble de l'herbe dans une image passé en paramètre

Paramètres

im	Image dont on veut extraire l'herbe
dst	Image binaire dont les zones blanches correspondront à l'herbe trouvé dans im

5.1.2.4 getNextMatrix()

```
int getNextMatrix (
            Mat & M )
```

Cette fonction va permettre d'obtenir l'image suivante dans la liste d'image a traiter

Paramètres

M	Matrice passé en référence qui contiendra l'image suivante

Renvoie

Un entier valant 0 si on arrive à la fin de la liste d'image et 1 dans le cas inverse

5.1.2.5 getPixelMatrix()

```
Mat getPixelMatrix (
            Mat image,
            int cols,
            int rows,
            int radius )
```

Récupére tous les pixels contenus dans un carré de coté 2∗radius+1 centré sur un pixel de coordonnées désirées sous force de matrice

Paramètres

image,l'image	source
cols,l'abscisse	du pixel sur lequel sera centrée le carré
rows,l'ordonnée	du pixel sur lequel sera centrée le carré
radius,le	rayon de la sous-matrice

Renvoie

une matrice de taille au moins 2∗radius+1

5.1.2.6 grassProcessing()

```
void grassProcessing (
            const Mat grass,
            Mat & dst )
```

Par référence, renvoie le plus grand élément connexe d'une image binaire entrée en paramètre. Ne foncionne uniquement si moins de 255 éléments connexes.

Paramètres

grass	Matrie source
dst	Matrice destination

Obsolète

5.1.2.7 hasValue()

```
int hasValue (
            Mat image )
```

Cette fonction renvoie le nombre de pixel non nul d'une image binaire.

Paramètres

image,une	image binaire

Renvoie

le nombre de pixels non nul de l'image binaire

5.1.2.8 init()

```
void init (
            String folder )
```

Cette fonction va charger l'ensemble des images dans le dossier passé en paramètres

Paramètres

folder	String correspondant au dossier ou seront cherché les images

5.1.2.9 jarvis()

```
void jarvis (
            const Mat grass,
            int size )
```

Interface entre la représentation de données d'opencv et de la marche de jarvis implémentée, voir jarvisSlave.

Paramètres

grass	Matrice binaire contenant les points à envelopper
size	Nombre de pixels à 255.

Obsolète

5.1.2.10 jarvisSlave()

```
void jarvisSlave (
            cPoint points[],
            int n,
            const Mat grass )
```

Implémentation de la marche de Jarvis permettant de trouver et d'afficher l'enveloppe connexe d'un ensemble de points. Cette fonction n'est pas utilisée en effet un bug a été découvert et par le manque de temps, nous avons stoppé le développement de cette méthode. A l'initialisation de l'algorithme, on place le pixel le plus à gauche dans la liste des points de l'enveloppe. Or l'algorithme s'arrête quand il retombe sur ce point. Mais comme la plupart des points sont alignés, l'algorithme retombe rarement sur ce point et boucle indéfiniment.

Paramètres

points	tableau des points
n	taille du tableau
grass	image binaire source

Obsolète

5.1.2.11 main()

```
int main (
            int argc,
            char * argv[] )
```

Fonction principale du programme

5.1.2.12 orientation()

```
int orientation (
            cPoint p,
            cPoint q,
            cPoint r )
```

Renvoi de quel côté se trouve le point r par rapport au vecteur pq

Paramètres

p	Point
q	Point
r	Point

Renvoie

0 si les trois points sont alignés, 1 si r est à gauche du vecteur pq, 0 sinon

Obsolète

5.1.2.13 patchAll3()

```
void patchAll3 (
            const Mat src1,
            const Mat src2,
            const Mat src3,
            Mat & dst,
            int taille_patch )
```

Par référence, renvoie une matrice binaire de même taille que les 3 autres matrices en paramètres. Pour chaque pixels des 3 matrices src, le pixel associé à la matrice renvoyée est mis à 255 si en appliquant un patch de taille taille_patch : aucun pixel est à 255 sur src1 ET 10∗taille_patch pixels sont à 255 sur src2 ET 5∗taille_patch pixels sont à 255 sur src3. Si aucun pixels n'a été trouvé et que taille_patch est à 10, la fonction est rappelée avec taille_patch à 5.

Paramètres

src1	Matrice binaire correspondant à l'herbe
src2	Matrice binaire correspondant aux objets blancs
src3	Matrice binaire correspondant au contours du terrain
dst	Référence vers matrice de destination
taille_patch	taille du patch à appliquer

5.1.2.14 process()

```
void process (
            String benchmark )
```

Cette fonction va réaliser tout le processus de détections des buts

Paramètres

benchmark	Option permettant de spécifier si l'on veut le type d'affichage voulue

Détection des poteaux des buts (zones blanches de l'image)

Détection de l'herbe (zone verte de l'image

Opération morphologique sur l'herbe afin d'obtenir une zone la plus lisse possible

5.1.2.15 progressBar()

```
void progressBar ( )
```

Cette fonction va gérer l'affichage de la barre de progression

5.1.2.16 rng()

```
RNG rng (
            12345  )
```

Cette variable sera utilisé pour générer aléatoirement des couleurs

5.1.2.17 tooMuchGrass()

```
bool tooMuchGrass (
            const Mat grass )
```

Cette fonction va vérifier la présence d'herbe sur les bords de l'image. Cette fonction est utilisée pour vérifier si le robot ne regarde pas a ces pieds et donc ne voit pas les bords du terrain

Paramètres

grass	Une image binaire représentant l'endroit ou se trouve l'herbe

Renvoie

Un booléen qui vaudra vrai si il y'a trop d'herbe en bordure de l'image (cela signie que le robot regarde à ces pieds) où false dans le cas contraire

5.1.2.18 usage()

```
void usage (
            const char * s )
```

Fonction qui sera appelé lorsque le nombre de paramètres utilisé pour appelé le programme est incorrecte

Paramètres

s	Le nom du programme

5.1.2.19 zoneColor()

```
int zoneColor (
            const Mat grass,
            Mat & current,
            int color_pixel,
            int x,
            int y )
```

Colorie l'espace connexe associé au point de coordonnées (x,y) à la couleur color_pixel, en nuance de gris uniquement

Paramètres

grass	Matrice binaire source
current	Matrice de nuance de gris destination
color_pixel	niveau de gris (0-255)
x	abscisse du point désiré
y	ordonnée du point désiré

Renvoie

Le nombre de pixels de cette espace connexe

Obsolète

5.1.3 Documentation des variables

5.1.3.1 bordure

```
int bordure = 5
```

Nombre depixel rajouté en bordure de l'image

5.1.3.2 image_counter

```
unsigned int image_counter
```

Entier qui va compter le nombre d'image traité

5.1.3.3 images

```
vector<Mat> images
```

Vecteur contenant l'ensemble des images du dossier passé en paramètre

5.1.3.4 progress

```
float progress = 0.0
```

Variable utilisé pour la barre de progression

Index