UAVAgroState

Generated by Doxygen 1.8.15

Contents

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 2

Class Documentation

2.1 Calibration Class Reference

Static Public Member Functions

- static void **calibrateImg** (int numCornersHor, int numCornersVer, string cameraName)
- static void **storeCalibrationMat** (Mat intrinsic, Mat distCoeffs, string cameraName)
- static vector< Mat > **readCalibrationMat** (string cameraName)

The documentation for this class was generated from the following file:

- Calibration.h

2.2 Color Class Reference

Static Public Member Functions

- static vector< Vec3f > **generarColores** ()
- static Vec3f **scalarToVec** (Scalar sclColor)

Static Public Attributes

- static cv::Scalar **aliceblue**
- static cv::Scalar **antiquewhite**
- static cv::Scalar **aqua**
- static cv::Scalar **aquamarine**
- static cv::Scalar **azure**
- static cv::Scalar **beige**
- static cv::Scalar **bisque**
- static cv::Scalar **black**
- static cv::Scalar **blanchedalmond**
- static cv::Scalar **blue**
- static cv::Scalar **blueviolet**
- static cv::Scalar **brown**

- static cv::Scalar **burlywood**
- static cv::Scalar **cadetblue**
- static cv::Scalar **chartreuse**
- static cv::Scalar **chocolate**
- static cv::Scalar **coral**
- static cv::Scalar **cornflowerblue**
- static cv::Scalar **cornsilk**
- static cv::Scalar **crimson**
- static cv::Scalar **cyan**
- static cv::Scalar **darkblue**
- static cv::Scalar **darkcyan**
- static cv::Scalar **darkgoldenrod**
- static cv::Scalar **darkgray**
- static cv::Scalar **darkgreen**
- static cv::Scalar **darkgrey**
- static cv::Scalar **darkkhaki**
- static cv::Scalar **darkmagenta**
- static cv::Scalar **darkolivegreen**
- static cv::Scalar **darkorange**
- static cv::Scalar **darkorchid**
- static cv::Scalar **darkred**
- static cv::Scalar **darksalmon**
- static cv::Scalar **darkseagreen**
- static cv::Scalar **darkslateblue**
- static cv::Scalar **darkslategray**
- static cv::Scalar **darkslategrey**
- static cv::Scalar **darkturquoise**
- static cv::Scalar **darkviolet**
- static cv::Scalar **deeppink**
- static cv::Scalar **deepskyblue**
- static cv::Scalar **dimgray**
- static cv::Scalar **dimgrey**
- static cv::Scalar **dodgerblue**
- static cv::Scalar **firebrick**
- static cv::Scalar **floralwhite**
- static cv::Scalar **forestgreen**
- static cv::Scalar **fuchsia**
- static cv::Scalar **gainsboro**
- static cv::Scalar **ghostwhite**
- static cv::Scalar **gold**
- static cv::Scalar **goldenrod**
- static cv::Scalar **gray**
- static cv::Scalar **green**
- static cv::Scalar **greenyellow**
- static cv::Scalar **grey**
- static cv::Scalar **honeydew**
- static cv::Scalar **hotpink**
- static cv::Scalar **indianred**
- static cv::Scalar **indigo**
- static cv::Scalar **ivory**
- static cv::Scalar **khaki**
- static cv::Scalar **lavender**
- static cv::Scalar **lavenderblush**
- static cv::Scalar **lawngreen**
- static cv::Scalar **lemonchiffon**

- static cv::Scalar **lightblue**
- static cv::Scalar **lightcoral**
- static cv::Scalar **lightcyan**
- static cv::Scalar **lightgoldenrodyellow**
- static cv::Scalar **lightgray**
- static cv::Scalar **lightgreen**
- static cv::Scalar **lightgrey**
- static cv::Scalar **lightpink**
- static cv::Scalar **lightsalmon**
- static cv::Scalar **lightseagreen**
- static cv::Scalar **lightskyblue**
- static cv::Scalar **lightslategray**
- static cv::Scalar **lightslategrey**
- static cv::Scalar **lightsteelblue**
- static cv::Scalar **lightyellow**
- static cv::Scalar **lime**
- static cv::Scalar **limegreen**
- static cv::Scalar **linen**
- static cv::Scalar **magenta**
- static cv::Scalar **maroon**
- static cv::Scalar **mediumaquamarine**
- static cv::Scalar **mediumblue**
- static cv::Scalar **mediumorchid**
- static cv::Scalar **mediumpurple**
- static cv::Scalar **mediumseagreen**
- static cv::Scalar **mediumslateblue**
- static cv::Scalar **mediumspringgreen**
- static cv::Scalar **mediumturquoise**
- static cv::Scalar **mediumvioletred**
- static cv::Scalar **midnightblue**
- static cv::Scalar **mintcream**
- static cv::Scalar **mistyrose**
- static cv::Scalar **moccasin**
- static cv::Scalar **navajowhite**
- static cv::Scalar **navy**
- static cv::Scalar **oldlace**
- static cv::Scalar **olive**
- static cv::Scalar **olivedrab**
- static cv::Scalar **orange**
- static cv::Scalar **orangered**
- static cv::Scalar **orchid**
- static cv::Scalar **palegoldenrod**
- static cv::Scalar **palegreen**
- static cv::Scalar **paleturquoise**
- static cv::Scalar **palevioletred**
- static cv::Scalar **papayawhip**
- static cv::Scalar **peachpuff**
- static cv::Scalar **peru**
- static cv::Scalar **pink**
- static cv::Scalar **plum**
- static cv::Scalar **powderblue**
- static cv::Scalar **purple**
- static cv::Scalar **red**
- static cv::Scalar **rosybrown**
- static cv::Scalar **royalblue**

- static cv::Scalar **saddlebrown**
- static cv::Scalar **salmon**
- static cv::Scalar **sandybrown**
- static cv::Scalar **seagreen**
- static cv::Scalar **seashell**
- static cv::Scalar **sienna**
- static cv::Scalar **silver**
- static cv::Scalar **skyblue**
- static cv::Scalar **slateblue**
- static cv::Scalar **slategray**
- static cv::Scalar **slategrey**
- static cv::Scalar **snow**
- static cv::Scalar **springgreen**
- static cv::Scalar **steelblue**
- static cv::Scalar **tan**
- static cv::Scalar **teal**
- static cv::Scalar **thistle**
- static cv::Scalar **tomato**
- static cv::Scalar **turquoise**
- static cv::Scalar **violet**
- static cv::Scalar **wheat**
- static cv::Scalar **white**
- static cv::Scalar **whitesmoke**
- static cv::Scalar **yellow**
- static cv::Scalar **yellowgreen**

The documentation for this class was generated from the following file:

- Color.h

2.3 CommonFunctions Class Reference

Static Public Member Functions

- static Mat **addIndent** (Mat boundBox, int position, int cols, int rows, int colGrid, int rowGrid, float colOverlap, float rowOverlap, int extraIndent=200)
- static void **info** (const cv::Mat &image, std::ostream &out=std::cout)
- static Mat **cargarImagen** (string strImg, int tamano=4, int Tipo=int(IMREAD_UNCHANGED))
- static vector< int > **setBorder** (Mat img)
- static Mat **Aindane** (Mat inputImage, int sigma)
- static Mat **getBorder** (Mat img, vector< int > minMax)
- static vector< Mat > **getBorders** (vector< Mat > imgs, vector< int > minMax)
- static vector< Mat > **cargarImagenes** (vector< string > strImgs, int tamano=4, int Tipo=int(IMREAD_U↩ NCHANGED))
- static vector< string > **obtenerImagenes** (const char *carpeta, bool reverse=false)
- static bool **crearCarpeta** (String str)
- static string **obtenerUltimoDirectorio** (string &carpeta)
- static string **obtenerUltimoDirectorio2** (string carpeta)
- static void **manejarCarpeta** (string carpeta)
- static void **escribirImagen** (string carpeta, Mat img)
- static void **showWindowNormal** (Mat img, String namewindow="img")

- static timeval **tiempo** (timeval begin, string msg)
- static Rect **rectROI** (Mat img)
- static Mat **cropRectorROI** (Mat img, Rect rect)
- static Mat **boundingBox** (Mat img, int colindent, int rowindent)
- static Mat **copyToTransparent** (Mat obj, Mat scene)
- static Mat **boundingBox** (Mat img, int arIndent, int abIndent, int izIndent, int derIndent)
- static bool **escribirPDF** (HPDF_Doc pdf, string str)
- static Mat **makeBackGroundTransparent** (Mat img)
- static void histDraw (Mat img, string imgName)
- static string **type2str** (Mat img)
- static Mat **addAlpha** (Mat img, Mat trans)
- static Mat **addTransparence** (Mat img)
- static Mat **removeAlpha** (Mat img)

2.3.1 Member Function Documentation

2.3.1.1 histDraw()

```
static void CommonFunctions::histDraw (
            Mat img,
            string imgName )  [inline], [static]
```

Establish the number of bins

Set the ranges (for B,G,R))

Compute the histograms:

Normalize the result to [0, histImage.rows]

The documentation for this class was generated from the following file:

- CommonFunctions.h

2.4 IndexCalculation Class Reference

Public Member Functions

- void **processManager** ()
- void **indexCalcu** (string strImg, bool multispectral)
- void **indexCalcuRGB** (string strImg)
- void **rgCalculation** (vector< Mat > BGRA, string strImg)
- void **indexCalcuMS** (string strImg)
- void **ndviCalculation** (vector< Mat > BGRA, string strImg)
- void **rviCalculation** (vector< Mat > BGRA, string strImg)
- void **escribirSegmentaciones** (Mat indice, Mat trans, string Nombre)
- vector< string > **obtenerInput** (bool multiespectral, bool outputStitching)
- vector< string > **obtenerRGBInput** ()
- vector< string > **obtenerMSInput** ()
- vector< string > **obtenerRGBOutput** ()
- vector< string > **obtenerMSOutput** ()

The documentation for this class was generated from the following file:

- IndexCalculation.h

2.5 Performance Class Reference

The documentation for this class was generated from the following file:

- Performance.h

2.6 Redaction Class Reference

Public Member Functions

- int **generatePDF** ()
- int **writeFullPageImg** (HPDF_Doc pdf, HPDF_Page page, const char *file)

The documentation for this class was generated from the following file:

- Redaction.h

2.7 Segmentation Class Reference

Static Public Member Functions

- static Mat segmentationVariation (Mat img, Mat trans, int cantColores)

  Realiza una normalizacion de la imagen (gris) y luego cuantiza sus valores para reducir las escalas de grises.
- static Mat **createLut** (Mat temp, Mat trans)
- static Mat **drawIndexOfIndex** (Mat img, Mat Lut, Mat trans, double min, double max)
- static Mat **normalizateMat** (Mat img, Mat mask, double &min, double &max)
- static vector< Mat > **generarGrafico** (Mat img, int cantidad, Mat trans)
- static vector< int > **threshMat** (Mat img, string str)
- static Mat **separarSuelo** (vector< Mat > BGRA)

Public Attributes

- double **min** =0
- double **max** =0

2.7.1 Member Function Documentation

2.7.1.1 segmentationVariation()

```
static Mat Segmentation::segmentationVariation (
            Mat img,
            Mat trans,
            int cantColores )  [inline], [static]
```

Realiza una normalizacion de la imagen (gris) y luego cuantiza sus valores para reducir las escalas de grises.

Parameters

img	
trans	
cantColores	

Returns

Mat

The documentation for this class was generated from the following file:

- Segmentation.h

2.8 uav::Stitcher Class Reference

Public Member Functions

- **Stitcher** (int tamano=4, int minKeypoints=5000, bool originalSize=false)
- vector< Mat > stitchWarp (Mat scene, Mat obj, Mat homoMatrix)

  funcion para pegar una imagen transformada por una homografia en otra imagen. En el caso de q tenga 4 canales (o sea el cuarto sea alpha [transparente]) hace un pegado especial para que no se pierda la transparencia, y en el caso contrario la pega de una manera q no se note el paso de una imagen a otra
- Mat copyToTransparent (Mat obj, Mat scene, Mat mask)

  Aplica un blending especial donde, en base a una mascara, decide que valor van a aportar los pixeles del objeto y la escena, en la escena final.
- double compareMats (int numHomo, Mat homoMatrix)

  Compara una imagen(escena) con la consecuente(objeto) transformada.
- void detectAndDescript ()

  Obtiene los keypoints y descriptores de cada imágen y los escribe en vecKp y vecDesc.
- void matchKp ()

  Realiza emparejamientos entre los keypoints de 2 imagenes, en base a sus descriptores(vecDesc), y escribe los resultados en vecMatch.
- Mat rigidToHomography (Mat R)

  convierte una matriz de transformada rigida a una homografía
- vector< DMatch > removeOutliers (vector< DMatch > gm, int numHomo, int numHomo2)

  Elimina emparejamientos erroneos por medio de la media y varianza de las posiciones en X e Y de los keypoints con y sin desplazamientos. Los emparejamientos con desplazamiento muy lejos de la media, se coinsideran erroneos.
- void getHomography (int numHomo)

  Obtengo varias homografias modificando ciertos parametros, elijo la que sea mas adecuada y la escribo en homo↩ NoMultiplicated.
- void getHomographies ()

  Obtengo las homografias entre cada par de imágenes consecutivas, las múltiplico para adaptar las transformaciones al espacio que se use como marco de referencia y las guardo en el vector H.
- void findBoundBoxLimits ()

  En base a las homografias, se obtienen los valores que van a delimitar al bound box.
- bool evaluateHomography ()

  Evalua que los limites del boundbox no sean de una homogragia mal calculada.
- void generateBoundBox ()

  Usando los boundboxlimits obtenidos en la funcion findBoundBoxLimits, se genera el boundbox.

- void rescaleHomographies ()

  En el caso de que las homografias se hayan calculado en base a imagenes a las cuales se les cambio el tamaño para que sea mas rapido el procesamiento, se les modifica la homografia para adaptarlas a su tamaño.
- void eraseFromVectors ()

  Elimina las imágenes que ya hayan sido pegadas.
- Mat stitchImgs ()

  En base a las homografias realiza el pegado de las imagenes.
- void removeCorners ()

  quito las esquinas para remover el vignetting
- void compensateBright ()

  Hago que todas las imágenes tengan la misma media en cada canal.
- Mat runAll ()

  Utilizo todas las funciones anteriores para realizar el stitching, siguiento el siguiente proceso:
- void processManager ()

  Realiza todo el proceso para pegar las imágenes.
- vector< string > obtenerInput ()

  Obtiene las ubicaciones las imágenes a pegar.
- vector< string > obtenerInputOrto ()

  Obtiene las ubicaciones de los resultados intermedos, para pegarlos y generar el resultado final.
- string obtenerOutputOrto (int num)

  Obtiene la ubicación donde se escribiran los resultados intermedios.
- string obtenerOutputRF ()

  Obtiene la ubicación donde se escribirá el resultado final.
- void escribirOutput (int numRes)

  Escribe los resultados intermedios o finaledependiendo de finalResult.
- void escribirOutputRF ()

  Escribe resultado final.
- void escribirOutputOrto (int numRes)

  Escribe resultados intermedios.

Public Attributes

- int tamano

  Tamaño con el que se redimensionan las imágenes.
- bool originalSize =false

  Booleano que decide si recuperar tamaño original en el resultado final.
- double yMin =0

  Limite superior de la caja de limites.
- double yMax =0

  Limite inferior de la caja de limites.
- double xMin =0

  Limite izquierdo de la caja de limites.
- double xMax =0

  Limite derecho de la caja de limites.
- int minKeypoints

  Cantidad minima de keypoints a calcular.
- bool finalResult

  Booleano que decide si se esta procesando los resultados intermedios o el resultado final.
- vector< Mat > imgs

  Imágenes a pegar.

- vector< Mat > vecDesc

  Descriptores de las imágenes.
- vector< Mat > H

  Transformaciones entre cada par de imágenes adaptadas al marco de referencia inicial.
- vector< Mat > homoNoMultiplicated

  Transformaciones entre cada par de imágenes no adaptadas.
- vector< string > strImgs

  Ubicación de cada imágen.
- vector< vector< KeyPoint > > vecKp

  Puntos claves de las imágenes.
- vector< vector< DMatch > > vecMatch

  Emparejamiento entre cada punto clave de imágenes consecutivas.
- vector< vector< DMatch > > best_inliers

  Emparejamientos correctos.
- Mat boundBox

  Caja de limites.

2.8.1 Member Function Documentation

2.8.1.1 compareMats()

```
double uav::Stitcher::compareMats (
            int numHomo,
            Mat homoMatrix )  [inline]
```

Compara una imagen(escena) con la consecuente(objeto) transformada.

Parameters

numHomo	
homoMatrix	

Returns

double

2.8.1.2 copyToTransparent()

```
Mat uav::Stitcher::copyToTransparent (
            Mat obj,
            Mat scene,
            Mat mask )  [inline]
```

Aplica un blending especial donde, en base a una mascara, decide que valor van a aportar los pixeles del objeto y la escena, en la escena final.

Parameters

obj	
scene	
mask	

Returns

Mat

2.8.1.3 escribirOutput()

```
void uav::Stitcher::escribirOutput (
            int numRes )  [inline]
```

Escribe los resultados intermedios o finaledependiendo de finalResult.

Parameters

numRes	

2.8.1.4 escribirOutputOrto()

```
void uav::Stitcher::escribirOutputOrto (
            int numRes )  [inline]
```

Escribe resultados intermedios.

Parameters

numRes	

2.8.1.5 evaluateHomography()

```
bool uav::Stitcher::evaluateHomography ( )  [inline]
```

Evalua que los limites del boundbox no sean de una homogragia mal calculada.

Returns

true
false

2.8.1.6 getHomography()

```
void uav::Stitcher::getHomography (
            int numHomo )  [inline]
```

Obtengo varias homografias modificando ciertos parametros, elijo la que sea mas adecuada y la escribo en homo↩ NoMultiplicated.

Parameters

numHomo	

2.8.1.7 obtenerInput()

```
vector<string> uav::Stitcher::obtenerInput ( )  [inline]
```

Obtiene las ubicaciones las imágenes a pegar.

Returns

vector<string>

2.8.1.8 obtenerInputOrto()

```
vector<string> uav::Stitcher::obtenerInputOrto ( )  [inline]
```

Obtiene las ubicaciones de los resultados intermedos, para pegarlos y generar el resultado final.

Returns

vector<string>

2.8.1.9 obtenerOutputOrto()

```
string uav::Stitcher::obtenerOutputOrto (
            int num )  [inline]
```

Obtiene la ubicación donde se escribiran los resultados intermedios.

Parameters

num	

Returns

string

2.8.1.10 obtenerOutputRF()

```
string uav::Stitcher::obtenerOutputRF ( )  [inline]
```

Obtiene la ubicación donde se escribirá el resultado final.

Returns

string

2.8.1.11 runAll()

```
Mat uav::Stitcher::runAll ( )  [inline]
```

Utilizo todas las funciones anteriores para realizar el stitching, siguiento el siguiente proceso:

- Quito esquinas.
- Obtengo keypoints y descriptores.
- Realizo emparejamiento.
- Obtengo homografias.
- Genero caja de límites.
- Adapto homografias al tamaño original.
- Pego las imagenes.

2.8.1.12 stitchImgs()

```
Mat uav::Stitcher::stitchImgs ( )  [inline]
```

En base a las homografias realiza el pegado de las imagenes.

Returns

Mat

2.8.1.13 stitchWarp()

```
vector<Mat> uav::Stitcher::stitchWarp (
            Mat scene,
            Mat obj,
            Mat homoMatrix )  [inline]
```

funcion para pegar una imagen transformada por una homografia en otra imagen. En el caso de q tenga 4 canales (o sea el cuarto sea alpha [transparente]) hace un pegado especial para que no se pierda la transparencia, y en el caso contrario la pega de una manera q no se note el paso de una imagen a otra

Parameters

scene	asd
obj	
homoMatrix	

Returns

vector<Mat>

The documentation for this class was generated from the following file:

- Stitcher.h

2.9 Undistort Class Reference

Static Public Member Functions

- static void **undistortImgs** (string cameraName)

The documentation for this class was generated from the following file:

- Undistort.h

Index