ProyectoVision

Generado por Doxygen 1.9.2

1 Proyecto Vision 1
1.1 Desarrollo . . . 1
1.2 Clases . . . 1
1.2.1 Clase Feeder . . . 2
1.2.2 Clase FrameLogger . . . 2
1.2.3 Clase ExtractorLandmarks . . . 2
1.2.3.1 Clase ExtractorLandmarksOpenCV . . . 2
1.2.3.2 Clase ExtractorLandmarksDlib . . . 2
1.2.4 Clase AnalizadorLandmarks . . . 2
1.2.5 Clase LandmarksLogger . . . 3
1.3 Estructura Landmarks . . . 3
1.4 Dependencias . . . 3
1.5 Compilación . . . 3

2 Indice jerárquico 5
2.1 Jerarquía de la clase . . . 5

3 Índice de clases 7
3.1 Lista de clases . . . 7

4 Documentación de las clases 9
4.1 Referencia de la Clase AnalizadorLandmarks . . . 9
4.1.1 Descripción detallada . . . 10
4.1.2 Documentación de las funciones miembro . . . 10
4.1.2.1 calcularAngulo() . . . 10
4.1.2.2 calcularAsimetria() . . . 10
4.1.2.3 calcularMax() . . . 11
4.1.2.4 calcularPendiente() . . . 11
4.1.2.5 getLandmarks() . . . 12
4.1.2.6 getNormLandmarks() . . . 12
4.1.2.7 normalizarLandmarks() . . . 12
4.1.2.8 setLandmarks() . . . 12
4.2 Referencia de la Clase AnalizadorSimetria . . . 13
4.2.1 Descripción detallada . . . 14
4.2.2 Documentación de las funciones miembro . . . 14
4.2.2.1 cargarConfiguracion() . . . 14
4.2.2.2 getAsimetria() . . . 14
4.2.2.3 getExtractor() . . . 15
4.2.2.4 getFeeder() . . . 15
4.2.2.5 getFrame() . . . 15
4.2.2.6 getLandmarks() . . . 16
4.2.2.7 getLandmarksNorm() . . . 16
4.2.2.8 setExtractor() . . . 16

4.2.2.9 setFeeder() . . . 16
4.2.2.10 setNombreLog() . . . 16
4.2.2.11 step() . . . 17
4.3 Referencia de la Clase ExtractorLandmarks . . . 17
4.3.1 Descripción detallada . . . 18
4.3.2 Documentación de las funciones miembro . . . 18
4.3.2.1 getLandmarks() . . . 18
4.3.2.2 parseLandmarks() . . . 19
4.4 Referencia de la Clase ExtractorLandmarksDlib . . . 19
4.4.1 Descripción detallada . . . 20
4.4.2 Documentación del constructor y destructor . . . 20
4.4.2.1 ExtractorLandmarksDlib() . . . 20
4.4.3 Documentación de las funciones miembro . . . 20
4.4.3.1 getExtractor() . . . 20
4.4.3.2 getLandmarks() . . . 20
4.5 Referencia de la Clase ExtractorLandmarksOpenCV . . . 21
4.5.1 Descripción detallada . . . 22
4.5.2 Documentación del constructor y destructor . . . 22
4.5.2.1 ExtractorLandmarksOpenCV() . . . 22
4.5.3 Documentación de las funciones miembro . . . 22
4.5.3.1 getExtractor() . . . 22
4.5.3.2 getLandmarks() . . . 22
4.6 Referencia de la Clase Feeder . . . 23
4.6.1 Descripción detallada . . . 24
4.6.2 Documentación de las funciones miembro . . . 24
4.6.2.1 getFeeder() . . . 24
4.6.2.2 getFrame() . . . 24
4.7 Referencia de la Clase FrameLogger . . . 25
4.7.1 Descripción detallada . . . 25
4.7.2 Documentación del constructor y destructor . . . 25
4.7.2.1 FrameLogger() [1/2] . . . 26
4.7.2.2 FrameLogger() [2/2] . . . 26
4.7.3 Documentación de las funciones miembro . . . 26
4.7.3.1 getCont() . . . 26
4.8 Referencia de la Clase KinectFeeder . . . 26
4.8.1 Descripción detallada . . . 27
4.8.2 Documentación del constructor y destructor . . . 27
4.8.2.1 KinectFeeder() . . . 28
4.8.3 Documentación de las funciones miembro . . . 28
4.8.3.1 getFeeder() . . . 28
4.8.3.2 getFrame() . . . 28
4.9 Referencia de la Estructura Landmarks . . . 29

4.9.1 Descripción detallada . . . 30
4.9.2 Documentación de las funciones miembro . . . 30
4.9.2.1 empty() . . . 30
4.9.3 Documentación de los datos miembro . . . 30
4.9.3.1 boca . . . 30
4.9.3.2 cejaDer . . . 31
4.9.3.3 cejaIzq . . . 31
4.9.3.4 menton . . . 31
4.9.3.5 nariz . . . 31
4.9.3.6 ojoDer . . . 32
4.9.3.7 ojoIzq . . . 32
4.10 Referencia de la Clase LandmarksLogger . . . 32
4.10.1 Descripción detallada . . . 33
4.10.2 Documentación del constructor y destructor . . . 33
4.10.2.1 LandmarksLogger() [1/2] . . . 33
4.10.2.2 LandmarksLogger() [2/2] . . . 33
4.10.3 Documentación de las funciones miembro . . . 33
4.10.3.1 getCont() . . . 34
4.11 Referencia de la Clase MiExcepcion . . . 34
4.11.1 Descripción detallada . . . 34
4.11.2 Documentación de las funciones miembro . . . 35
4.11.2.1 what() . . . 35
4.12 Referencia de la Clase VideoFeeder . . . 35
4.12.1 Descripción detallada . . . 36
4.12.2 Documentación del constructor y destructor . . . 36
4.12.2.1 VideoFeeder() [1/2] . . . 36
4.12.2.2 VideoFeeder() [2/2] . . . 36
4.12.3 Documentación de las funciones miembro . . . 36
4.12.3.1 getFeeder() . . . 37
4.12.3.2 getFrame() . . . 37
4.13 Referencia de la Clase WebcamFeeder . . . 37
4.13.1 Descripción detallada . . . 38
4.13.2 Documentación del constructor y destructor . . . 38
4.13.2.1 WebcamFeeder() . . . 38
4.13.3 Documentación de las funciones miembro . . . 38
4.13.3.1 getFeeder() . . . 38
4.13.3.2 getFrame() . . . 39

Índice alfabético **41**

Capítulo 1

Proyecto Vision

Proyecto final para las materias "Vision Artificial" y "Complementos de Informatica" El proyecto consiste en una clase capaz de obtener imagenes de distintos medios, detectar rostros, obtener puntos de interés y analizar su simetria. Ademas, puede registrar tanto el video adquirido como los puntos de interés detectados. Estos datos provistos pueden ser utilizados para detectar momentos donde la asimetria sea maxima o donde se detecte claramente cierta expresion facial (sonrisa, ceño fruncido, levantar cejas, etc). La idea de la implementación está basada en este paper.

1.1. Desarrollo

El software consiste en una clase base que se compone de otras subclases. Luego, los objetos concretos de estas clases interactúan entre sí.

.

Esquema de composición de la clase base.

.

Esquema de las relaciones internas entre clases.

El sistema instancia los objetos necesarios a partir de un archivo de configuración provisto, con las clases concretas necesarias segun el caso. Si no se pasan argumentos, intenta cargar una configuración por defecto, abriendo una webcam y utilindo un archivo de entrenamiento por defecto.

Además, se incluye el código para probar diversas funcionalidades de la clase.

1.2. Clases

A continuacion, se detallan las distintas clases base que compondran al sistema.

1.2.1. Clase Feeder

Clase encargada de obtener los fotogramas a analizar. De ésta se derivan tres clases para proveer frames de distintos medios. Las tres clases que heredan de Feeder, serán:

+ VideoFeeder, pensada para trabajar con videos. + WebcamFeeder, pensada para adquirir fotogramas de una webcam. + KinectFeeder, pensada para adquirir fotogramas de una kinect utilizando libfreenect2.

Estas clases tendrán un método general para devolver un fotograma del tipo `cv::Mat` que serán utilizados por los objetos de las clases (`FrameLogger` y `ExtractorLandmarks`).

Ejemplo en UML de la clase "feeder"

1.2.2. Clase FrameLogger

Esta clase está encargada de registrar los frames provistos por el `Feeder`. Por el momento, va registrando los fotogramas en un video por defecto. Debe tener un metodo de actualizacion que consista en guardar el archivo de imagen en algun lugar en particular, con un nombre que lo identifique unicamente, y que de alguna manera quede linkeado a una "base de datos".

1.2.3. Clase ExtractorLandmarks

Esta clase está encargada de obtener los puntos de interés de un rostro a partir de las imágenes provistas por el `Feeder`. Esta clase abstracta en principio tiene dos implementaciones:

+ clase ExtractorLandmarksOpenCV, utilizando openCV + clase ExtractorLandmarksDlib, utilizando dlib

Estas dos clases utilizarían distintos algoritmos para la deteccion de puntos de interes, basados en distintos papers y con distintos entrenamientos.

Diagrama UML de la clase ExtractorLandmarks

1.2.3.1. Clase ExtractorLandmarksOpenCV

Esta clase está basada en las librerias de openCV.

1.2.3.2. Clase ExtractorLandmarksDlib

Esta clase está basada en las librerias de dlib

1.2.4. Clase AnalizadorLandmarks

Esta clase normaliza los puntos de interés provistos por el `extractor de landmarks`, es decir, corrige la inclinacion de la cabeza, tomando como referencias los puntos a mitad de cada oreja. Con esto, calcula la simetria de la cara basándose en algunos puntos particulares(Elegidos medio aleatoriamente) Respecto a la simetria, la clase será la encargada de analizar los puntos de interés normalizados, haciendo algunos calculos geométricos y devolviendo distintas medidas sobre la simetria facial. En el `paper` de referencia, estas medidas se utilizan para luego alimentar un clasificador. Al no tener acceso a los datasets para poder "clasificar" distintos rostros, este ultimo paso se dificulta. Aun asi, debe ser posible obtener un puntaje analizando distintas medidas y comparando ambos lados del rostro. Además, el normalizador podria filtrar solo los landmarks necesarios para el calculo de simetria, reduciendo asi el tamaño de los datos guardados. Finalmente, la clase devuelve un vector de `una estructura predefinida`, habiendo una estructura por cada rostro detectado.

1.2.5. Clase LandmarksLogger

Esta clase está encargada de registrar cada vector de `landmarks` obtenido. Registra los datos en un archivo de tipo YAML, tomando un nombre de base y agregandole el tipo de feeder utilizado y un timestamp.

1.3. Estructura Landmarks

Es la estructura que devuelve el extractor de landmarks para facilitar su uso. Está compuesta por las siguientes propiedades:

+ vacio : Bandera para detectar si la estructura posee información + rotacion : Indica la rotación de la cabeza, en caso de haberse normalizado. + escala : Indica la escala de los puntos, en caso de haberse normalizado. + menton : `std::vector<cv::Point2f>` vector de puntos de openCV que demarcan el mentón. + ojoIzq : `std::vector<cv::Point2f>` vector de puntos de openCV que demarcan el ojo izquierdo. + ojoDer : `std::vector<cv::Point2f>` vector de puntos de openCV que demarcan el ojo derecho. + cejaIzq : `std::vector<cv::Point2f>` vector de puntos de openCV que demarcan la ceja izquierda. + cejaDer : `std::vector<cv::Point2f>` vector de puntos de openCV que demarcan la ceja derecha. + boca : `std::vector<cv::Point2f>` vector de puntos de openCV que demarcan la boca. + nariz : `std::vector<cv::Point2f>` vector de puntos de openCV que demarcan la nariz.

1.4. Dependencias

+ `dlib` (utilizada version 19.22) + `libfreenect2` (utilizada version 0.2.0) + `openCV` (utilizada version 4.5.2)

1.5. Compilación

Para compilarlo, se utilizó el siguiente comando:
```
g++ -Wall -DUSE_AVX_INSTRUCTIONS=ON /home/agustin/Facultad/5to/ProyectoVision/*.cc /home/agustin/Facultad/5to/ProyectoVision/include/src/* -o /home/agustin/Facultad/5to/ProyectoVision/Release/mainProyectoVision -I/home/agustin/Facultad/5to/ProyectoVision/include -I/usr/include/ -I/usr/local/lib/ -I/usr/local/include -I/usr/local/include/opencv4 -L/usr/local/freenect2/lib -L/usr/lib -lopencv_core -lopencv_highgui -lopencv_imgcodecs -lopencv_video -lopencv_videoio -lopencv_plot -lopencv_objdetect -lopencv_imgproc -lopencv_face -ldlib -lturbojpeg -ljpeg -lfreenect2 -llapack -lopenblas -O3
```
Generado con el archivo `tasks.json` en visual studio code.

Capítulo 2

Indice jerárquico

2.1. Jerarquía de la clase

Esta lista de herencias esta ordenada aproximadamente por orden alfabético:

AnalizadorLandmarks 9
AnalizadorSimetria 13
ExtractorLandmarks 17
    ExtractorLandmarksDlib 19
    ExtractorLandmarksOpenCV 21
Feeder 23
    KinectFeeder 26
    VideoFeeder 35
    WebcamFeeder 37
FrameLogger 25
Landmarks 29
LandmarksLogger 32
std::exception
    MiExcepcion 34

Capítulo 3

Índice de clases

3.1. Lista de clases

Lista de las clases, estructuras, uniones e interfaces con una breve descripción:

AnalizadorLandmarks
: Clase que maneja los metodos para analizar los landmarks 9

AnalizadorSimetria
: Clase principal del programa 13

ExtractorLandmarks
: Clase abstracta para extraer landmarks de un Mat 17

ExtractorLandmarksDlib
: Implementación concreta de la clase abstracta ExtractorLandmarks usando dlib 19

ExtractorLandmarksOpenCV
: Clase concreta derivada de ExtractorLandmarks para extraer landmarks utilizando openCV . . 21

Feeder
: Clase abstracta para proveer obtener nuevos mats 23

FrameLogger
: Clase para guardar los frames. Por defecto genera un video 25

KinectFeeder
: Sobrecarga de la clase Feeder para proveer objetos Mat desde una kinect 26

Landmarks
: Estructura para almacenar los landmarks, discriminados segun rasgo facial 29

LandmarksLogger
: Clase para guardar los landmarks. Por defecto genera un video 32

MiExcepcion
: Clase de excepciones personal heredando de las excepciones estandar 34

VideoFeeder
: Sobrecarga de la clase Feeder para abrir un archivo de video 35

WebcamFeeder
: Sobrecarga de la clase Feeder para abrir una webcam 37

Capítulo 4

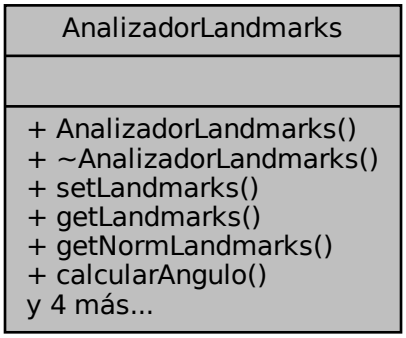
Documentación de las clases

4.1. Referencia de la Clase AnalizadorLandmarks

Clase que maneja los metodos para analizar los landmarks.

`#include <analizadorlandmarks.h>`

Diagrama de colaboración para AnalizadorLandmarks:



Métodos públicos

- AnalizadorLandmarks ()

  Construye un nuevo objeto de la clase Analizador Landmarks.
- ~AnalizadorLandmarks ()

  Destruye el objeto de la clase Analizador Landmarks.
- void setLandmarks (const std::vector< Landmarks > &)

  Actualiza los landmarks del objeto.
- const std::vector< Landmarks > getLandmarks ()

  Devuelve los landmarks del objeto.

- std::vector< Landmarks > getNormLandmarks ()

  Devuelve los landmarks normalizados del objeto.
- const float calcularAngulo (const Point2f &, const Point2f &)

  Método que devuelve el angulo entre dos puntos a y b.
- const float calcularPendiente (const Point2f &, const Point2f &)

  Metodo que devuelve la pendiente entre dos puntos a y b.
- const float calcularMax (const float &, const float &)

  Método que devuelve el máximo valor entre dos puntos.
- const float calcularAsimetria ()

  Método principal de la clase, analiza la asimetria de un rostro.
- void normalizarLandmarks ()

  Funcion que normaliza los landmarks, eliminando la posible rotacion del rostro.

4.1.1. Descripción detallada

Clase que maneja los metodos para analizar los landmarks.

4.1.2. Documentación de las funciones miembro

4.1.2.1. calcularAngulo()

```
const float AnalizadorLandmarks::calcularAngulo (
            const Point2f & a,
            const Point2f & b )  [inline]
```

Método que devuelve el angulo entre dos puntos a y b.

Parámetros

a	- punto a
b	- punto b

Devuelve

const float - angulo en grandos

Referenciado por normalizarLandmarks().

4.1.2.2. calcularAsimetria()

```
const float AnalizadorLandmarks::calcularAsimetria ( )
```

Método principal de la clase, analiza la asimetria de un rostro.

No está completamente desarrollada, pero debería devolver una asimetría de un rostro. Lo ideal en realidad sería que devuelva una estructura con diferentes parámetros, para luego alimentar a una red neuronal y discernir la asimetria.

Devuelve

const float

Hace referencia a calcularMax().

Referenciado por AnalizadorSimetria::step().

4.1.2.3. calcularMax()

```
const float AnalizadorLandmarks::calcularMax (
            const float & a,
            const float & b )  [inline]
```

Método que devuelve el máximo valor entre dos puntos.

Parámetros

a	- punto a
b	- punto b

Devuelve

const float - valor máximo

Referenciado por calcularAsimetria().

4.1.2.4. calcularPendiente()

```
const float AnalizadorLandmarks::calcularPendiente (
            const Point2f & a,
            const Point2f & b )  [inline]
```

Metodo que devuelve la pendiente entre dos puntos a y b.

Parámetros

a	- punto a
b	- punto b

Devuelve

const float - pendiente entre los dos puntos

4.1.2.5. getLandmarks()

```
const std::vector<Landmarks> AnalizadorLandmarks::getLandmarks ( )  [inline]
```

Devuelve los landmarks del objeto.

Devuelve

const std::vector<Landmarks>

4.1.2.6. getNormLandmarks()

```
std::vector<Landmarks> AnalizadorLandmarks::getNormLandmarks ( )  [inline]
```

Devuelve los landmarks normalizados del objeto.

Devuelve

std::vector<Landmarks>

4.1.2.7. normalizarLandmarks()

```
void AnalizadorLandmarks::normalizarLandmarks ( )
```

Funcion que normaliza los landmarks, eliminando la posible rotacion del rostro.

La normalización cobra relevancia si se toma como parámetros la pendiente o el angulo entre distintos landmarks en el rostro. Por el momento no tiene utilidad, pero en el caso de alimentar una RN cobraría relevancia.

Hace referencia a Landmarks::boca, calcularAngulo(), Landmarks::cejaDer, Landmarks::cejaIzq, Landmarks::escala, Landmarks::menton, Landmarks::nariz, Landmarks::ojoDer, Landmarks::ojoIzq y Landmarks::rotacion.

Referenciado por AnalizadorSimetria::step().

4.1.2.8. setLandmarks()

```
void AnalizadorLandmarks::setLandmarks (
            const std::vector< Landmarks > &  )
```

Actualiza los landmarks del objeto.

Parámetros

lm	landmarks nuevos.

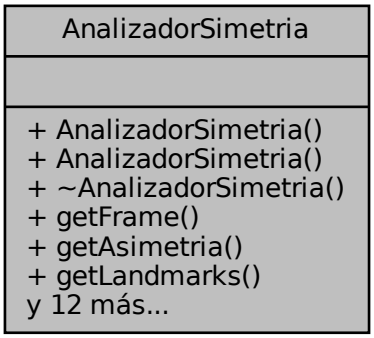
Referenciado por AnalizadorSimetria::step().

4.2. Referencia de la Clase AnalizadorSimetria

Clase principal del programa.

`#include <analizadorsimetria.h>`

Diagrama de colaboración para AnalizadorSimetria:



Métodos públicos

- AnalizadorSimetria ()

  Construye un nuevo objeto de la clase Analizador Simetria. Utiliza valores por defecto.
- AnalizadorSimetria (const string &)

  Construye un nuevo objeto de la clase Analizador Simetria, proveyendole un archivo de configuración.
- ~AnalizadorSimetria ()

  Destruye el objeto de la clase Analizador Simetria.
- Mat getFrame ()

  Devuelve el último frame obtenido.
- const float getAsimetria ()

  Devuelve la última asimetria calculada.
- const std::vector< Landmarks > getLandmarks ()

  Devuelve el ultimo vector de Landmarks calculado.
- const std::vector< Landmarks > getLandmarksNorm ()

  Devuelve el último vector de Landmarks normalizado.
- TipoFeeder getFeeder ()

  Devuelve el tipo concreto de Feeder utilizado.
- TipoExtractor getExtractor ()

  Devuelve el tipo de ExtractorLandmarks utilizado.
- void setFeeder (TipoFeeder)

  Setea tipo concreto de Feeder del analizador.

- void setExtractor (TipoExtractor)

  Setea el tipo concreto de Extractor del analizador.
- void setNombreLog (const string &nombre)

  Setea el nombre base del video del logger.
- void empezarVideoLog (const TipoFeeder &)

  Método para invocar un FrameLogger y comenzar la grabación.
- void empezarLandmarksLog (const TipoFeeder &)

  Método para invocar un LandmarksLogger y comenzar la grabación.
- void stopVideoLog ()

  Método para detener la grabación del FrameLogger.
- void stopLandmarksLog ()

  Método para detener la grabación del LandmarksLogger.
- Mat step ()

  Método principal a llamar para operar la clase.
- void cargarConfiguracion (const string &)

  Método encargado de cargar el archivo de configuración y definir las propiedades del objeto.

4.2.1. Descripción detallada

Clase principal del programa.

Se construye pasándole un archivo de configuración (por defecto "config.yaml"), el cual es encargado de elegir el tipo de feeder y extractor a utilizar. La clase además provee distintos métodos para seleccionar el tipo de feeder o extractor a utilizar. Además, permite iniciar o detener el registro en video o de landmarks. Una vez inicializada, su método principal es step(), el cual devuelve un nuevo frame, y si está habilitado el log y el extractor de landmarks, agrega el fotograma al video y analiza los nuevos landmarks.

4.2.2. Documentación de las funciones miembro

4.2.2.1. cargarConfiguracion()

```
void AnalizadorSimetria::cargarConfiguracion (
            const string & nombreArchivo )
```

Método encargado de cargar el archivo de configuración y definir las propiedades del objeto.

Parámetros

nombreConf	- Nombre del archivo de configuración.

Referenciado por AnalizadorSimetria().

4.2.2.2. getAsimetria()

```
const float AnalizadorSimetria::getAsimetria ( )  [inline]
```

Devuelve la última asimetria calculada.

Devuelve

const float - Asimetria calculada

4.2.2.3. getExtractor()

```
TipoExtractor AnalizadorSimetria::getExtractor ( )  [inline]
```

Devuelve el tipo de ExtractorLandmarks utilizado.

Devuelve

TipoExtractor

4.2.2.4. getFeeder()

```
TipoFeeder AnalizadorSimetria::getFeeder ( )  [inline]
```

Devuelve el tipo concreto de Feeder utilizado.

Devuelve

TipoFeeder

Hace referencia a Feeder::getFeeder().

4.2.2.5. getFrame()

```
Mat AnalizadorSimetria::getFrame ( )  [inline]
```

Devuelve el último frame obtenido.

Devuelve

Mat - objeto del tipo cv::Mat conteniendo el fotograma actual

4.2.2.6. getLandmarks()

```
const std::vector<Landmarks> AnalizadorSimetria::getLandmarks ( )  [inline]
```

Devuelve el ultimo vector de Landmarks calculado.

Devuelve

const std::vector<Landmarks> - Vector de Landmarks

4.2.2.7. getLandmarksNorm()

```
const std::vector<Landmarks> AnalizadorSimetria::getLandmarksNorm ( )  [inline]
```

Devuelve el último vector de Landmarks normalizado.

Devuelve

const std::vector<Landmarks> - Vector de Landmarks normalizado

4.2.2.8. setExtractor()

```
void AnalizadorSimetria::setExtractor (
            TipoExtractor extractor_ )
```

Setea el tipo concreto de Extractor del analizador.

Genera un nuevo objeto de un ExtractoLandmarks concreto. Si se elige el mismo tipo que el ya establecido, no hace nada. Si no, elimina el que se estaba utilizando y genera uno nuevo.

Hace referencia a empezarLandmarksLog().

Referenciado por AnalizadorSimetria().

4.2.2.9. setFeeder()

```
void AnalizadorSimetria::setFeeder (
            TipoFeeder feeder_ )
```

Setea tipo concreto de Feeder del analizador.

Genera un nuevo objeto de algun Feeder concreto. Si se elige el mismo tipo de feeder que el actual, no hace nada. Si no, elimina el que se estaba utilizando y genera uno nuevo.

Hace referencia a empezarVideoLog() y Feeder::getFeeder().

Referenciado por AnalizadorSimetria().

4.2.2.10. setNombreLog()

```
void AnalizadorSimetria::setNombreLog (
            const string & nombre )  [inline]
```

Setea el nombre base del video del logger.

Parámetros

nombre	- Nombre base del video de salida del logger

4.2.2.11. step()

```
Mat AnalizadorSimetria::step ( )
```

Método principal a llamar para operar la clase.

Este método es el encargado de actualizar toda la clase. Si hay un feeder definido, intenta obtener un nuevo frame. Si el log está habilitado, le envia el frame al logger. Si hay un extractor de landmarks, obtiene los landmarks. En caso de obtener un rostro, actualiza las propiedades de la clase y finalmente calcula la asimetría.

Devuelve

Mat

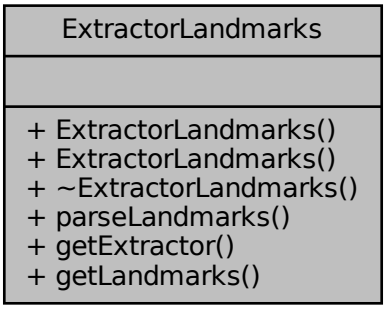
Hace referencia a AnalizadorLandmarks::calcularAsimetria(), Feeder::getFrame(), ExtractorLandmarks::get↩ Landmarks(), FrameLogger::log(), LandmarksLogger::log(), AnalizadorLandmarks::normalizarLandmarks() y AnalizadorLandmarks::setLandmarks().

4.3. Referencia de la Clase ExtractorLandmarks

Clase abstracta para extraer landmarks de un Mat.

```
#include <extractorlandmarks.h>
```

Diagrama de colaboración para ExtractorLandmarks:



Métodos públicos

- ExtractorLandmarks (const std::vector< string > &)

  Construye un nuevo objeto de la clase abstracta Extractor Landmarks.
- virtual ~ExtractorLandmarks ()

  Destruye el objeto de la clase Extractor Landmarks.
- const std::vector< Landmarks > parseLandmarks (const std::vector< std::vector< cv::Point2f >> &)

  Método para convertir los landmarks "crudos" a una estructura con distintos rasgos.
- virtual const std::vector< Landmarks > getLandmarks (const cv::Mat &)=0

  Devuelve la propiedad Landmarks.

4.3.1. Descripción detallada

Clase abstracta para extraer landmarks de un Mat.

4.3.2. Documentación de las funciones miembro

4.3.2.1. getLandmarks()

```
const std::vector< Landmarks > ExtractorLandmarks::getLandmarks (
            const cv::Mat & frame )  [pure virtual]
```

Devuelve la propiedad Landmarks.

Metodo para obtener landmarks de un Mat.

Devuelve

const std::vector<Landmarks>

Parámetros

cv::Mat	Frame a analizar

Devuelve

std::vector<cv::Point2f>

Implementado en ExtractorLandmarksDlib y ExtractorLandmarksOpenCV.

Referenciado por AnalizadorSimetria::step().

4.3.2.2. parseLandmarks()

```
const std::vector< Landmarks > ExtractorLandmarks::parseLandmarks (
            const std::vector< std::vector< cv::Point2f >> & landmarksSerie )
```

Método para convertir los landmarks "crudos" a una estructura con distintos rasgos.

Devuelve

const std::vector<Landmarks>

Hace referencia a Landmarks::boca, Landmarks::cejaDer, Landmarks::cejaIzq, Landmarks::menton, Landmarks::nariz, Landmarks::ojoDer, Landmarks::ojoIzq y Landmarks::vacio.

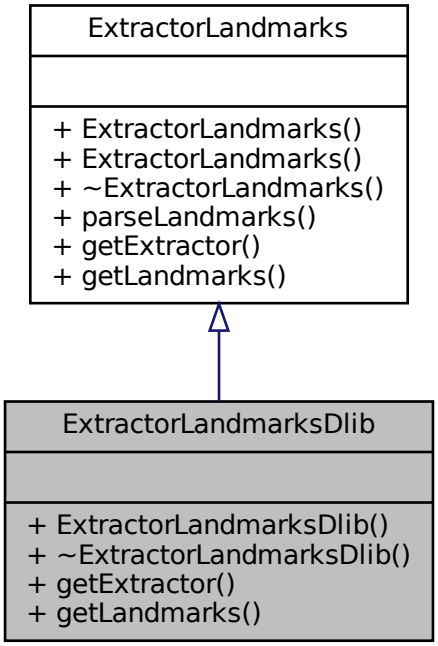
Referenciado por ExtractorLandmarksDlib::getLandmarks() y ExtractorLandmarksOpenCV::getLandmarks().

4.4. Referencia de la Clase ExtractorLandmarksDlib

Implementación concreta de la clase abstracta ExtractorLandmarks usando dlib.

```
#include <extractorlandmarks_dlib.h>
```

Diagrama de colaboración para ExtractorLandmarksDlib:



Métodos públicos

- ExtractorLandmarksDlib (const std::vector< string > &)

  Construye un nuevo objeto de la clase ExtractorLandmarksDlib.
- virtual ~ExtractorLandmarksDlib ()

  Destruye el objeto de la clase ExtractorLandmarksDlib.
- virtual const TipoExtractor getExtractor ()

  Devuelve el TipoExtractor, en este caso DLIB.
- virtual const std::vector< Landmarks > getLandmarks (const cv::Mat &)

  Obtiene y devuelve los landmarks.

4.4.1. Descripción detallada

Implementación concreta de la clase abstracta ExtractorLandmarks usando dlib.

Implementación de un extractor de landmarks utilizando dlib. Se implementó utilizando como referencia el código de ejemplo provisto por la libreria.

4.4.2. Documentación del constructor y destructor

4.4.2.1. ExtractorLandmarksDlib()

```
ExtractorLandmarksDlib::ExtractorLandmarksDlib (
            const std::vector< string > & nombres )
```

Construye un nuevo objeto de la clase ExtractorLandmarksDlib.

Se le debe proveer el nombre del archivo del detector.

Hace referencia a MiExcepcion::what().

4.4.3. Documentación de las funciones miembro

4.4.3.1. getExtractor()

```
virtual const TipoExtractor ExtractorLandmarksDlib::getExtractor ( )  [inline], [virtual]
```

Devuelve el TipoExtractor, en este caso DLIB.

Devuelve

const TipoExtractor

Implementa ExtractorLandmarks.

4.4.3.2. getLandmarks()

```
const std::vector< Landmarks > ExtractorLandmarksDlib::getLandmarks (
            const cv::Mat & frame )  [virtual]
```

Obtiene y devuelve los landmarks.

Parámetros

frame	- Frame a analizar.

Devuelve

const std::vector<Landmarks>

Implementa ExtractorLandmarks.

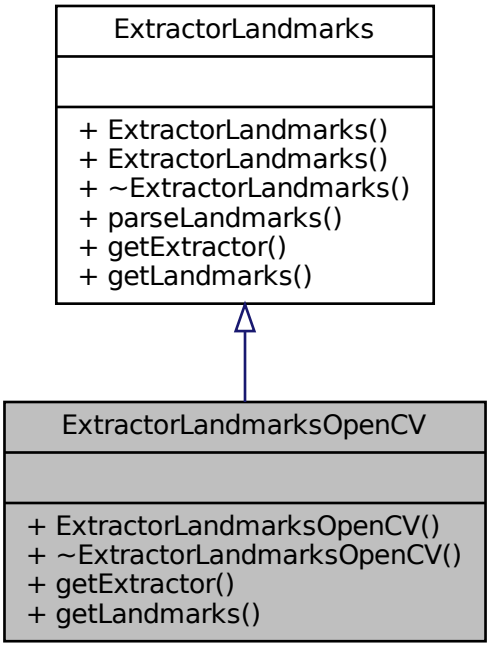
Hace referencia a ExtractorLandmarks::parseLandmarks() y Landmarks::vacio.

4.5. Referencia de la Clase ExtractorLandmarksOpenCV

Clase concreta derivada de ExtractorLandmarks para extraer landmarks utilizando openCV.

`#include <extractorlandmarks_opencv.h>`

Diagrama de colaboración para ExtractorLandmarksOpenCV:



Métodos públicos

- ExtractorLandmarksOpenCV (const std::vector< string > &)

  Construye un nuevo objeto de la clase ExtractorLandmarksOpenCV.
- virtual ~ExtractorLandmarksOpenCV ()

  Destruye el objeto de la clase ExtractorLandmarksOpenCV.
- virtual const TipoExtractor getExtractor ()

  Devuelve el parametro TipoExtractor (en este caso OPENCV)
- virtual const std::vector< Landmarks > getLandmarks (const cv::Mat &)

  Método que analiza un frame y devuelve los landmarks de un solo rostro.

4.5.1. Descripción detallada

Clase concreta derivada de ExtractorLandmarks para extraer landmarks utilizando openCV.

4.5.2. Documentación del constructor y destructor

4.5.2.1. ExtractorLandmarksOpenCV()

```
ExtractorLandmarksOpenCV::ExtractorLandmarksOpenCV (
            const std::vector< string > & nombres )
```

Construye un nuevo objeto de la clase ExtractorLandmarksOpenCV.

Se le debe proveer un vector con los nombres de los archivos a utilizar.

@params nombres - Vector de strings con los nombres de los archivos necesarios

Hace referencia a MiExcepcion::what().

4.5.3. Documentación de las funciones miembro

4.5.3.1. getExtractor()

```
virtual const TipoExtractor ExtractorLandmarksOpenCV::getExtractor ( )  [inline], [virtual]
```

Devuelve el parametro TipoExtractor (en este caso OPENCV)

Devuelve

const TipoExtractor

Implementa ExtractorLandmarks.

4.5.3.2. getLandmarks()

```
const std::vector< Landmarks > ExtractorLandmarksOpenCV::getLandmarks (
            const cv::Mat & frame )  [virtual]
```

Método que analiza un frame y devuelve los landmarks de un solo rostro.

Parámetros

frame	objeto Mat a analizar

Devuelve

std::vector<cv::Point2f>

Implementa ExtractorLandmarks.

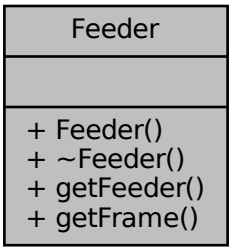
Hace referencia a ExtractorLandmarks::parseLandmarks() y Landmarks::vacio.

4.6. Referencia de la Clase Feeder

Clase abstracta para proveer obtener nuevos mats.

`#include <feeder.h>`

Diagrama de colaboración para Feeder:



Métodos públicos

- Feeder ()

  Construye un nuevo objeto de la clase abstracta Feeder.
- virtual ~Feeder ()

  Destruye el objeto de la clase abstracta Feeder.
- virtual const TipoFeeder getFeeder ()=0

  Devuelve el TipoFeeder del feeder implementado.
- virtual const Mat getFrame ()=0

  Devuelve el frame actual.

4.6.1. Descripción detallada

Clase abstracta para proveer obtener nuevos mats.

Clase abstracta para utilizar como interfaz a las implementaciones concretas. Provee dos métodos básicos que todas las clases utilizarán.

4.6.2. Documentación de las funciones miembro

4.6.2.1. getFeeder()

```
virtual const TipoFeeder Feeder::getFeeder ( )  [pure virtual]
```

Devuelve el TipoFeeder del feeder implementado.

Devuelve

const TipoFeeder

Implementado en KinectFeeder, VideoFeeder y WebcamFeeder.

Referenciado por AnalizadorSimetria::getFeeder() y AnalizadorSimetria::setFeeder().

4.6.2.2. getFrame()

```
virtual const Mat Feeder::getFrame ( )  [pure virtual]
```

Devuelve el frame actual.

Devuelve

const Mat

Implementado en KinectFeeder, VideoFeeder y WebcamFeeder.

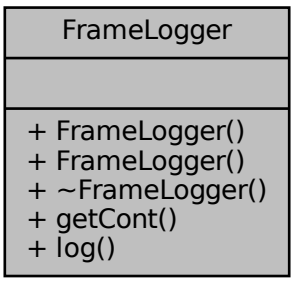
Referenciado por AnalizadorSimetria::step().

4.7. Referencia de la Clase FrameLogger

Clase para guardar los frames. Por defecto genera un video.

`#include <framelogger.h>`

Diagrama de colaboración para FrameLogger:



Métodos públicos

- FrameLogger (const string &, const TipoFeeder &)

  Construye un nuevo objeto de la clase Frame Logger.
- FrameLogger ()

  Constructor por defecto, genera un nombre por defecto.
- ~FrameLogger ()

  Destruye el objeto de la clase Frame Logger.
- double getCont ()

  Devuelve el la cantidad de fotogramas registrados. Puede ser util.
- void log (const Mat &)

  Agrega un frame al video e incrementa el contador.

4.7.1. Descripción detallada

Clase para guardar los frames. Por defecto genera un video.

Clase encargada de guardar el los fotogramas obtenidos. Toma un nombre por defecto para el archivo de video, al cual le inserta el feeder del que se proveen los fotogramas, y la fecha y hora del comienzo del registro. Por lo pronto solo trabaja con videos, pero eventualmetne podria guardar secuencias de imagenes.

4.7.2. Documentación del constructor y destructor

4.7.2.1. FrameLogger() [1/2]

```
FrameLogger::FrameLogger (
            const string & nombre,
            const TipoFeeder & feeder )
```

Construye un nuevo objeto de la clase Frame Logger.

Constructor a utilizar generalmente, se le pasa como argumentos el nombre base del video y el tipo de feeder utilizado.

4.7.2.2. FrameLogger() [2/2]

```
FrameLogger::FrameLogger ( )
```

Constructor por defecto, genera un nombre por defecto.

4.7.3. Documentación de las funciones miembro

4.7.3.1. getCont()

```
double FrameLogger::getCont ( )  [inline]
```

Devuelve el la cantidad de fotogramas registrados. Puede ser util.

Devuelve

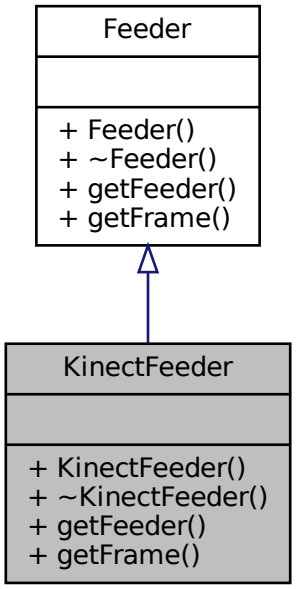
double

4.8. Referencia de la Clase KinectFeeder

Sobrecarga de la clase Feeder para proveer objetos Mat desde una kinect.

```
#include <kinectfeeder.h>
```

Diagrama de colaboración para KinectFeeder:



Métodos públicos

- KinectFeeder ()

  Construye un nuevo objeto de la clase KinectFeeder.
- virtual ~KinectFeeder ()

  Destruye el objeto de la clase KinectFeeder.
- virtual const TipoFeeder getFeeder ()

  Devuelve el TipoFeeder (en este caso, KINECTFEEDER)
- virtual const Mat getFrame ()

  Devuelve el frame actual de la kinect.

4.8.1. Descripción detallada

Sobrecarga de la clase Feeder para proveer objetos Mat desde una kinect.

Por ahora, solo obtiene imágenes de la cámara RGB. Para definir el tipo de log a consola, se debe declarar una variable de entorno "LIBFREENECT2_LOGGER_LEVEL".

4.8.2. Documentación del constructor y destructor

4.8.2.1. KinectFeeder()

```
KinectFeeder::KinectFeeder ( )
```

Construye un nuevo objeto de la clase KinectFeeder.

Obtiene todos los parámetros necesarios y se encarga de inicializar la Kinect.

4.8.3. Documentación de las funciones miembro

4.8.3.1. getFeeder()

```
virtual const TipoFeeder KinectFeeder::getFeeder ( )  [inline], [virtual]
```

Devuelve el TipoFeeder (en este caso, KINECTFEEDER)

Devuelve

const TipoFeeder

Implementa Feeder.

4.8.3.2. getFrame()

```
const Mat KinectFeeder::getFrame ( )  [virtual]
```

Devuelve el frame actual de la kinect.

Devuelve

const Mat

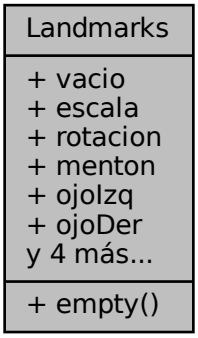
Implementa Feeder.

4.9. Referencia de la Estructura Landmarks

Estructura para almacenar los landmarks, discriminados segun rasgo facial.

`#include <estructuras.h>`

Diagrama de colaboración para Landmarks:



Métodos públicos

- const bool empty ()

  Método para definir si la estructura está vacía.

Atributos públicos

- bool vacio = 0

  Define si la estructura está vacia.
- float escala = 1

  Define la escala del rostro, en caso de aplicarse. Util en caso de normalizar landmarks.
- float rotacion = 0

  Rotación en grados del rostro. Util cuando se normalizan landmarks.
- std::vector< Point2f > menton

  Vector de puntos que definen el menton/contorno del rostro.
- std::vector< Point2f > ojoIzq

  Vector de puntos que definen el ojo izquierdo.
- std::vector< Point2f > ojoDer

  Vector de puntos que definen el ojo Derecho.
- std::vector< Point2f > cejaIzq

  Vector de puntos que definen la ceja izquierda.
- std::vector< Point2f > cejaDer

  Vector de puntos que definen la ceja Derecha.
- std::vector< Point2f > boca

  Vector de puntos que definen la boca.
- std::vector< Point2f > nariz

  Vector de puntos que delimitan la nariz.

4.9.1. Descripción detallada

Estructura para almacenar los landmarks, discriminados segun rasgo facial.

4.9.2. Documentación de las funciones miembro

4.9.2.1. empty()

```
const bool Landmarks::empty ( )  [inline]
```

Método para definir si la estructura está vacía.

No se si tiene sentido, ya que se puede consultar directamente a la propiedad vacio.

Devuelve

true
false

Hace referencia a vacio.

4.9.3. Documentación de los datos miembro

4.9.3.1. boca

```
std::vector<Point2f> Landmarks::boca
```

Vector de puntos que definen la boca.

Consiste en 20 puntos. Comienza desde la comisura externa de los labios, recorriendo el borde externo del labio comenzando por la parte superior. El punto 0 corresponde a la comisura externa izquierda, el punto 3 corresponde al punto central del borde externo superior, el punto 6 corresponde a la comisura externa derecha, el punto 9 corresponde al punto central del borde externo inferior.

El punto 12 corresponde a la comisura interna izquierda, el punto 14 corresponde al punto central del borde interno superior, el punto 16 corresponde a la comisura interna derecha, el punto 18 corresponde al punto central del borde interno inferior.

Referenciado por LandmarksLogger::log(), AnalizadorLandmarks::normalizarLandmarks() y ExtractorLandmarks↩ ::parseLandmarks().

4.9.3.2. cejaDer

`std::vector<Point2f> Landmarks::cejaDer`

Vector de puntos que definen la ceja Derecha.

Consiste en 5 puntos. Comienza desde el punto más central de la ceja, y su último punto es el mas alejado del centro.

Referenciado por LandmarksLogger::log(), AnalizadorLandmarks::normalizarLandmarks() y ExtractorLandmarks↩ ::parseLandmarks().

4.9.3.3. cejalzq

`std::vector<Point2f> Landmarks::cejaIzq`

Vector de puntos que definen la ceja izquierda.

Consiste en 5 puntos. Comienza desde el punto más central de la ceja, y su último punto es el mas alejado del centro.

Referenciado por LandmarksLogger::log(), AnalizadorLandmarks::normalizarLandmarks() y ExtractorLandmarks↩ ::parseLandmarks().

4.9.3.4. menton

`std::vector<Point2f> Landmarks::menton`

Vector de puntos que definen el menton/contorno del rostro.

Consiste en 17 puntos. Comienza desde la oreja izquierda, siendo el punto 8 el del centro del mentón

Referenciado por LandmarksLogger::log(), AnalizadorLandmarks::normalizarLandmarks() y ExtractorLandmarks↩ ::parseLandmarks().

4.9.3.5. nariz

`std::vector<Point2f> Landmarks::nariz`

Vector de puntos que delimitan la nariz.

Consiste en 9 puntos. El punto 0 corresponde al punto superior del tabique, el punto 3 corresponde al último punto del tabique. El punto 4 corresponde al extremo izquierdo de la base de la nariz, el punto 6 corresponde a la punta de la nariz, y el punto 8 corresponde al extremo derecho de la base de la nariz

Referenciado por LandmarksLogger::log(), AnalizadorLandmarks::normalizarLandmarks() y ExtractorLandmarks↩ ::parseLandmarks().

4.9.3.6. ojoDer

```
std::vector<Point2f> Landmarks::ojoDer
```

Vector de puntos que definen el ojo Derecho.

Consiste en 6 puntos. Comienza desde la comisura interna del ojo, recorriendo el párpado superior, y siendo la comisura externa el punto 3. Finalmente recorre el parpado inferior.

Referenciado por LandmarksLogger::log(), AnalizadorLandmarks::normalizarLandmarks() y ExtractorLandmarks↩ ::parseLandmarks().

4.9.3.7. ojoIzq

```
std::vector<Point2f> Landmarks::ojoIzq
```

Vector de puntos que definen el ojo izquierdo.

Consiste en 6 puntos. Comienza desde la comisura interna del ojo, recorriendo el párpado superior, y siendo la comisura externa el punto 3. Finalmente recorre el parpado inferior.

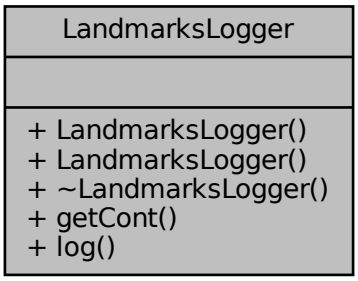
Referenciado por LandmarksLogger::log(), AnalizadorLandmarks::normalizarLandmarks() y ExtractorLandmarks↩ ::parseLandmarks().

4.10. Referencia de la Clase LandmarksLogger

Clase para guardar los landmarks. Por defecto genera un video.

```
#include <landmarkslogger.h>
```

Diagrama de colaboración para LandmarksLogger:



Métodos públicos

- LandmarksLogger (const string &, const TipoFeeder &)

  Construye un nuevo objeto de la clase Landmark Logger.
- LandmarksLogger ()

  Constructor por defecto, genera un nombre por defecto.
- ∼LandmarksLogger ()

  Destruye el objeto de la clase LandmarkLogger.
- double getCont ()

  Devuelve el la cantidad de fotogramas registrados. Puede ser util.
- void log (const std::vector< Landmarks > &)

  Agrega un landmark al archivo e incrementa el contador.

4.10.1. Descripción detallada

Clase para guardar los landmarks. Por defecto genera un video.

Clase encargada de guardar el los fotogramas obtenidos. Toma un nombre por defecto para el archivo de video, al cual le inserta el feeder del que se proveen los fotogramas, y la fecha y hora del comienzo del registro. Por lo pronto solo trabaja con videos, pero eventualmetne podria guardar secuencias de imagenes.

4.10.2. Documentación del constructor y destructor

4.10.2.1. LandmarksLogger() [1/2]

```
LandmarksLogger::LandmarksLogger (
            const string & nombre,
            const TipoFeeder & feeder )
```

Construye un nuevo objeto de la clase Landmark Logger.

Constructor a utilizar generalmente, se le pasa como argumentos el nombre base del video y el tipo de feeder utilizado.

4.10.2.2. LandmarksLogger() [2/2]

```
LandmarksLogger::LandmarksLogger ( )
```

Constructor por defecto, genera un nombre por defecto.

4.10.3. Documentación de las funciones miembro

4.10.3.1. getCont()

```
double LandmarksLogger::getCont ( )  [inline]
```

Devuelve el la cantidad de fotogramas registrados. Puede ser util.

Devuelve

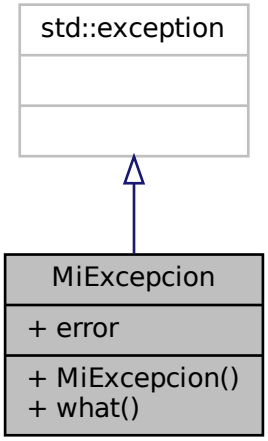
double

4.11. Referencia de la Clase MiExcepcion

Clase de excepciones personal heredando de las excepciones estandar.

```
#include <estructuras.h>
```

Diagrama de colaboración para MiExcepcion:



Métodos públicos

- virtual const char ∗ what () const throw ()

  Devuelve un texto según el tipo de error que se le pase.

4.11.1. Descripción detallada

Clase de excepciones personal heredando de las excepciones estandar.

Devuelve un texto de acuerdo al tipo de error provisto, según TipoError.

4.11.2. Documentación de las funciones miembro

4.11.2.1. what()

```
virtual const char* MiExcepcion::what ( ) const throw ( )   [inline], [virtual]
```

Devuelve un texto según el tipo de error que se le pase.

Devuelve

const char*

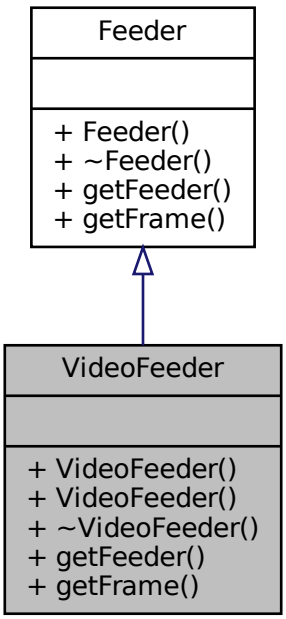
Referenciado por AnalizadorSimetria::AnalizadorSimetria(), AnalizadorSimetria::empezarLandmarksLog(), AnalizadorSimetria::empezarVideoLog(), ExtractorLandmarksDlib::ExtractorLandmarksDlib() y ExtractorLandmarksOpenCV::ExtractorLandmarksOpenCV().

4.12. Referencia de la Clase VideoFeeder

Sobrecarga de la clase Feeder para abrir un archivo de video.

```
#include <videofeeder.h>
```

Diagrama de colaboración para VideoFeeder:



Métodos públicos

- VideoFeeder (string &)

  Construye un nuevo objeto de la clase VideoFeeder.
- VideoFeeder ()

  Construye un nuevo objeto de la clase VideoFeeder, tomando un nombre por defecto.
- virtual ~VideoFeeder ()

  Destruye el objeto de la clase VideoFeeder.
- virtual const TipoFeeder getFeeder ()

  Devuelve un valor de TipoFeeder (En este caso VIDEOFEEDER)
- virtual const Mat getFrame ()

  Devuelve el ultimo frame procesado.

4.12.1. Descripción detallada

Sobrecarga de la clase Feeder para abrir un archivo de video.

Abre un archivo de video, cuyo nombre se pasa a la hora de construir el feeder. Por ahora, cuando llega al final del video, se corta la ejecucion.

4.12.2. Documentación del constructor y destructor

4.12.2.1. VideoFeeder() [1/2]

```
VideoFeeder::VideoFeeder (
            string & nombre )
```

Construye un nuevo objeto de la clase VideoFeeder.

Parámetros

nombre	- nombre del video a abrir

4.12.2.2. VideoFeeder() [2/2]

```
VideoFeeder::VideoFeeder ( )
```

Construye un nuevo objeto de la clase VideoFeeder, tomando un nombre por defecto.

Intenta abrir el archivo "video.avi"

4.12.3. Documentación de las funciones miembro

4.12.3.1. getFeeder()

```
virtual const TipoFeeder VideoFeeder::getFeeder ( )  [inline], [virtual]
```

Devuelve un valor de TipoFeeder (En este caso VIDEOFEEDER)

Devuelve

const TipoFeeder

Implementa Feeder.

4.12.3.2. getFrame()

```
const Mat VideoFeeder::getFrame ( )  [virtual]
```

Devuelve el ultimo frame procesado.

Devuelve

const Mat

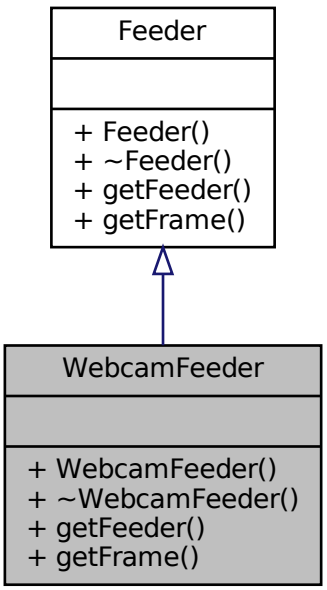
Implementa Feeder.

4.13. Referencia de la Clase WebcamFeeder

Sobrecarga de la clase Feeder para abrir una webcam.

```
#include <webcamfeeder.h>
```

Diagrama de colaboración para WebcamFeeder:



Métodos públicos

- WebcamFeeder (int idx=0)

  Construye un nuevo objeto de la clase WebcamFeeder.
- virtual ~WebcamFeeder ()

  Destruye el objeto de la clase WebcamFeeder.
- virtual const TipoFeeder getFeeder ()

  Devuelve el TipoFeeder (en este caso WEBCAMFEEDER)
- virtual const Mat getFrame ()

  Devuelve el fotograma actual.

4.13.1. Descripción detallada

Sobrecarga de la clase Feeder para abrir una webcam.

4.13.2. Documentación del constructor y destructor

4.13.2.1. WebcamFeeder()

```
WebcamFeeder::WebcamFeeder (
            int idx = 0 )
```

Construye un nuevo objeto de la clase WebcamFeeder.

Se encarga de inicializar la webcam a través del objeto de la clase VideoCapture

Parámetros

idx	- índice de la webcam a abrir

4.13.3. Documentación de las funciones miembro

4.13.3.1. getFeeder()

```
virtual const TipoFeeder WebcamFeeder::getFeeder ( )  [inline], [virtual]
```

Devuelve el TipoFeeder (en este caso WEBCAMFEEDER)

Devuelve

const TipoFeeder

Implementa Feeder.

4.13.3.2. getFrame()

```
const Mat WebcamFeeder::getFrame ( )  [virtual]
```

Devuelve el fotograma actual.

Devuelve

const Mat

Implementa Feeder.

Índice alfabético

AnalizadorLandmarks, 9
  calcularAngulo, 10
  calcularAsimetria, 10
  calcularMax, 11
  calcularPendiente, 11
  getLandmarks, 11
  getNormLandmarks, 12
  normalizarLandmarks, 12
  setLandmarks, 12
AnalizadorSimetria, 13
  cargarConfiguracion, 14
  getAsimetria, 14
  getExtractor, 15
  getFeeder, 15
  getFrame, 15
  getLandmarks, 15
  getLandmarksNorm, 16
  setExtractor, 16
  setFeeder, 16
  setNombreLog, 16
  step, 17

boca
  Landmarks, 30

calcularAngulo
  AnalizadorLandmarks, 10
calcularAsimetria
  AnalizadorLandmarks, 10
calcularMax
  AnalizadorLandmarks, 11
calcularPendiente
  AnalizadorLandmarks, 11
cargarConfiguracion
  AnalizadorSimetria, 14
cejaDer
  Landmarks, 30
cejaIzq
  Landmarks, 31

empty
  Landmarks, 30
ExtractorLandmarks, 17
  getLandmarks, 18
  parseLandmarks, 18
ExtractorLandmarksDlib, 19
  ExtractorLandmarksDlib, 20
  getExtractor, 20
  getLandmarks, 20
ExtractorLandmarksOpenCV, 21
  ExtractorLandmarksOpenCV, 22
  getExtractor, 22
  getLandmarks, 22

Feeder, 23
  getFeeder, 24
  getFrame, 24
FrameLogger, 25
  FrameLogger, 25, 26
  getCont, 26

getAsimetria
  AnalizadorSimetria, 14
getCont
  FrameLogger, 26
  LandmarksLogger, 33
getExtractor
  AnalizadorSimetria, 15
  ExtractorLandmarksDlib, 20
  ExtractorLandmarksOpenCV, 22
getFeeder
  AnalizadorSimetria, 15
  Feeder, 24
  KinectFeeder, 28
  VideoFeeder, 36
  WebcamFeeder, 38
getFrame
  AnalizadorSimetria, 15
  Feeder, 24
  KinectFeeder, 28
  VideoFeeder, 37
  WebcamFeeder, 38
getLandmarks
  AnalizadorLandmarks, 11
  AnalizadorSimetria, 15
  ExtractorLandmarks, 18
  ExtractorLandmarksDlib, 20
  ExtractorLandmarksOpenCV, 22
getLandmarksNorm
  AnalizadorSimetria, 16
getNormLandmarks
  AnalizadorLandmarks, 12

KinectFeeder, 26
  getFeeder, 28
  getFrame, 28
  KinectFeeder, 27

Landmarks, 29
  boca, 30

cejaDer, 30
cejaIzq, 31
empty, 30
menton, 31
nariz, 31
ojoDer, 31
ojoIzq, 32
LandmarksLogger, 32
getCont, 33
LandmarksLogger, 33

menton
Landmarks, 31
MiExcepcion, 34
what, 35

nariz
Landmarks, 31
normalizarLandmarks
AnalizadorLandmarks, 12

ojoDer
Landmarks, 31
ojoIzq
Landmarks, 32

parseLandmarks
ExtractorLandmarks, 18

setExtractor
AnalizadorSimetria, 16
setFeeder
AnalizadorSimetria, 16
setLandmarks
AnalizadorLandmarks, 12
setNombreLog
AnalizadorSimetria, 16
step
AnalizadorSimetria, 17

VideoFeeder, 35
getFeeder, 36
getFrame, 37
VideoFeeder, 36

WebcamFeeder, 37
getFeeder, 38
getFrame, 38
WebcamFeeder, 38
what
MiExcepcion, 35