PUMATRONIX

CLASSIFIER

CLASSIFIER

SOFTWARE DE RECONOCIMIENTO AUTOMÁTICO DE CARACTERÍSTICAS DEL VEHÍCULO

Integración

Pumatronix Equipamentos Eletrônicos Ltda.

Rua Bartolomeu Lourenço de Gusmão, 1970. Curitiba, Brasil

Visite nuestro sitio web https://www.pumatronix.com

Enviar comentarios sobre este documento al correo electrónico suporte@pumatronix.com

La información contenida en este documento está sujeta a cambios sin previo aviso.

Pumatronix se reserva el derecho de modificar o mejorar este material sin obligación de notificarle sobre cambios o mejoras.

Pumatronix otorga permiso para descargar e imprimir este documento, siempre que la copia electrónica o física de este documento contenga el texto completo. Cualquier cambio en este contenido está estrictamente prohibido..

Cambia la Historia

Data	Revisión	Contenido actualizado
23/07/2024	1.0	Revisión de la maquetación y formato general del documento; Contenido referente a la versión 1.16.0 del producto

Visión General

Este documento tiene como objetivo guiar al desarrollador en la aplicación de la biblioteca de software Classifier responsable de clasificar tipos de vehículos en base al análisis de imágenes y aplicable a software compatible con la biblioteca. Este documento detalla las opciones de configuración para el kit de desarrollo de software (SDK) y las API disponibles.

Sumario

1. Descomprimiendo el SDK

El Classifier SDK se distribuye a través de un archivo comprimido en formato 7zip. Para descomprimir este archivo se requiere una contraseña, que se proporciona junto con el correo electrónico que contiene las instrucciones de descarga de este SDK. Si tiene problemas para descomprimir el SDK, comuníquese con soporte por correo electrónico contato@pumatronix.com.br o WhatsApp +55 (41) 9203-8327.

Para descomprimir el SDK en un entorno Linux, use el siguiente comando desde una terminal (reemplace los campos <Classifier_PC_LINUX_64_vX.Y.Z.7z> y con la contraseña proporcionada por correo electrónico y el nombre correcto del paquete).

```
7z x -p< CONTRASEÑA_PROPORCIONADA> <Classifier_PC_LINUX_64_vX.Y.Z.7z>
```

Permisos del hardkey

Esta configuración solo es necesaria para la versión Linux del SDK.

Para que la llave USB funcione correctamente en Linux, se deben cambiar los permisos de acceso de udev. Añade la siguiente línea:

```
ATTRS{idVendor}=="0403", ATTRS{idProduct}=="c580", MODE="0666"
```

al final del archivo correspondiente a su distribución de Linux:

```
Centos 5.2/5.4:          /etc/udev/rules.d/50-udev.rules
Centos 6.0 en adelante:  /lib/udev/rules.d/50-udev-default.rules
Ubuntu 7.10:             /etc/udev/rules.d/40-permissions.rules
Ubuntu 8.04/8.10:        /etc/udev/rules.d/40-basic-permissions.rules
Ubuntu 9.04 en adelante:       /lib/udev/rules.d/50-udev-default.rules
openSUSE 11.2 en adelante: /lib/udev/rules.d/50-udev-default.rules
```

Si la distribución es Debian, añade las líneas:

```
SUBSYSTEM=="usb_device", MODE="0666"
SUBSYSTEM=="usb", ENV{DEVTYPE}=="usb_device", MODE="0666"
```

al final del archivo:

```
Debian 6.0 en adelante: /lib/udev/rules.d/91-permissions.rules
```

Para obtener instrucciones sobre cómo habilitar la tecla física en otras distribuciones de Linux, comuníquese con su contato@pumatronix.com.br ou WhatsApp +55 (41) 9203-8327.

Variables de entorno

Configuración de LD_LIBRARY_PATH

Esta configuración solo es necesaria para la versión Linux del SDK. Para la versión de Windows, haga una copia de la DLL a la carpeta donde se encuentra el ejecutable o a la carpeta system32

Para que la aplicación pueda encontrar las bibliotecas del clasificador, se debe agregar la ruta del directorio lib del SDK a la RUTA. En un entorno Linux, esto se puede hacer usando la variable de entorno `LD_LIBRARY_PATH`, como se muestra a continuación (reemplace con la ruta absoluta de instalación del SDK).

```
export LD_LIBRARY_PATH=/lib
```

2. Estructura del SDK

Classifier no fue diseñado como una aplicación independiente sino más bien como una biblioteca para integrarse en otras aplicaciones. Por lo tanto, Classifier SDK se distribuye como un conjunto de bibliotecas compartidas (.so y .dll) y sus encabezados en lenguaje C.

El SDK también viene con una aplicación de ejemplo (código fuente y binario precompilado) que puede usarse como base para implementar una aplicación que use el Clasificador. La sección de ejemplo básico de este manual también proporciona una guía paso a paso sobre cómo implementar una aplicación.

Árbol de archivos de la versión de Linux

```
include
      ptx
            classifier
                  ptx_classifier_api_local.h
            common
                  ptx_codes.h
                  ptx_defines.h
                  ptx_dict.h
                  ptx_image.h
            ptx_classifier.h
            ptx_common.h
lib
      libptx_classifierJava.so
      libptx_classifier.so
      libptx_commonJava.so
      libptx_common.so
sample
      bin
            libptx_classifierJava.so
            libptx_classifier.so
            libptx_commonJava.so
            libptx_common.so
            PtxClassifierSample
      src
            PtxClassifierSample.c
      wrapper
            java
                  br
                        com
                              pumatronix
                                    classifier
                                          PtxClassifier.java
                                          PtxClassifierSample.java
                                    common
                                          PtxCommon.java
                                          PtxDict.java
                                          PtxImage.java
```

```
      python
            PtxClassifier.py
            PtxClassifierSample.py
            PtxCommonLoader.py
            PtxCommon.py
            PtxDict.py
            PtxImage.py
```

Árbol de archivos de la versión de Windows

```
include
      ptx
            classifier
                  ptx_classifier_api_local.h
            common
                  ptx_codes.h
                  ptx_defines.h
                  ptx_dict.h
                  ptx_image.h
            ptx_classifier.h
            ptx_common.h
lib
      ptx_classifier.dll
      ptx_classifierJava.dll
      ptx_classifierJava.lib
      ptx_classifier.lib
      ptx_common.dll
      ptx_commonJava.dll
      ptx_commonJava.lib
      ptx_common.lib
res
      manual_classifier.pdf
sample
      bin
            PtxClassifierSample.exe
      src
            PtxClassifierSample.c
wrapper
      delphi
            ptx
                  common
                        ptx_common.pas
      java
            br
                  com
                        pumatronix
                              classifier
                                    PtxClassifier.java
                                    PtxClassifierSample.java
                              common
                                    PtxCommon.java
                                    PtxDict.java
                                    PtxImage.java
      python
            PtxClassifier.py
            PtxClassifierSample.py
            PtxCommonLoader.py
            PtxCommon.py
```

```
            PtxDict.py
            PtxImage.py
```

3. Arquitectura de software

El classifier tiene una arquitectura simple. Todas las llamadas API son sincrónicas, lo que significa que se bloquean hasta que se completa el procesamiento. La función principal es ptx_classifier_classify, que recibe una imagen previamente cargada y un identificador que contiene la configuración y, al final del procesamiento, completa una estructura de resultados.

La configuración de la biblioteca consta de: tipo de escena (panorámica o cerrada) y la probabilidad mínima para que una detección se considere un vehículo válido.

Los resultados constan de un conjunto de vehículos detectados, cada uno con los siguientes datos asociados: clase de vehículo, fiabilidad de clasificación en forma de probabilidad y coordenadas en la imagen del rectángulo que contiene el vehículo.

4. Ejemplo de uso deAPI

Para ver un ejemplo completo, consulte el archivo `sample/src/PtxClassifierSample.c` del SDK.

Para compilar la aplicación de ejemplo con gcc y ejecutarla, ejecute la siguiente secuencia de comandos desde una terminal Linux:

```
gcc sample/src/PtxClassifierSample.c -I include/ -L lib/ -l ptx_classifier -l ptx_common -o
PtxClassifierSample LD_LIBRARY_PATH=lib/ ./PtxClassifierSample
```

Una vez que se inicia la aplicación, debería aparecer en la salida del terminal:

```
[PUMA] JidoshaClassifier - Client Sample

[PUMA] JidoshaClassifier library Version: X.Y.Z
[PUMA] JidoshaClassifier library SHA1: 0123456789abcdef0123456789abcdef01234567
[PUMA] ptx_common library SHA1: 123456789abcdef0123456789abcdef012345678
[PUMA] LicenseInfo - Serial: 123456789 - maxThreads: 6 - maxConnections: 0
[PUMA] LicenseInfo - State: 0 - TTL: -1 - Customer: Pumatronix
[PUMA] ./PtxClassifierSample
```

5. APIs de usuario

La biblioteca Classifier exporta una API para el reconocimiento automático de características del vehículo.

De forma predeterminada, los lenguajes admitidos por la API que viene con el SDK son C, C++ y Java. Se pueden proporcionar contenedores en Python, C# y Delphi bajo demanda. Si tiene dudas o soporte para otros idiomas, envíe un correo electrónico a contato@pumatronix.com.br o WhatsApp +55 (41) 9203-8327.

Para estandarizar las estructuras comunes utilizadas por sus productos, Pumatronix creó la biblioteca ptx_common, que implementa estructuras de datos, códigos de error y otras definiciones de uso común en C. La biblioteca ptx_common es necesaria para utilizar Classifier y está incluida en el SDK. La documentación de su API se puede encontrar en la sección API ptx_common C/C++.

API Classifier C/C++

La API (interfaz de programación de aplicaciones) nativa de la biblioteca está escrita en lenguaje C, lo que facilita la creación de enlaces para su uso en otros lenguajes. Toda la API de C está disponible a través de un conjunto de encabezados dentro de la carpeta de inclusión del SDK.

La API contiene los tipos, definiciones y funciones para procesar imágenes. Está definido en el archivo `ptx_classifier_api_local.h`.

ptx_classifier_common.h

```
typedef enum PtxClassifierConfigs {
      PTX_CLASSIFIER_CONFIG_SCENE_TYPE                          = 0,
      PTX_CLASSIFIER_CONFIG_VEHICLE_TYPE_MIN_PROB               = 1,
      PTX_CLASSIFIER_CONFIG_MODEL_TYPE                          = 2,
      PTX_CLASSIFIER_CONFIG_ENABLE_VEHICLE_CHARACTERISTICS      = 3,
      PTX_CLASSIFIER_CONFIG_ENUM_MAX
} PtxClassifierConfigs;

typedef enum PtxClassifierSceneType {
      PTX_CLASSIFIER_SCENE_TYPE_CLOSEUP                         = 0, // Default scene
      PTX_CLASSIFIER_SCENE_TYPE_PANORAMIC                       = 1,
      PTX_CLASSIFIER_SCENE_TYPE_PANORAMIC_NIGHT                 = 2,
      PTX_CLASSIFIER_SCENE_TYPE_WIDERANGE                       = 3,
      PTX_CLASSIFIER_SCENE_TYPE_ENUM_MAX
} PtxClassifierSceneType;

typedef enum PtxClassifierModelType {
      PTX_CLASSIFIER_MODEL_TYPE_VEHICLES                        = 0,
      PTX_CLASSIFIER_MODEL_TYPE_BMC                             = 1,
      PTX_CLASSIFIER_MODEL_TYPE_WIND_SHIELD                     = 2, // Model not supported
yet, in development
      PTX_CLASSIFIER_MODEL_TYPE_VEHICLES_EXTENDED               = 3,
      PTX_CLASSIFIER_MODEL_TYPE_ENUM_MAX
} PtxClassifierModelType;

typedef enum PtxClassifierObjExtras {
      // Upper left
      PTX_CLASSIFIER_OBJ_UL_PT_X            = 0,
      PTX_CLASSIFIER_OBJ_UL_PT_Y            = 1,

      // Lower right
      PTX_CLASSIFIER_OBJ_LR_PT_X            = 2,
      PTX_CLASSIFIER_OBJ_LR_PT_Y            = 3,

      // Type
      PTX_CLASSIFIER_OBJ_TYPE_CLASS         = 4,
} PtxClassifierObjExtras;

//============================================================================
// Results
//============================================================================

// Prediction types
typedef enum PtxClassifierResultType {
      PTX_CLASSIFIER_GET_INT_RESULTS_SIZE                       = 0,
```

```
    PTX_CLASSIFIER_GET_PTXDICT_AT_RESULT                    = 1,

    PTX_CLASSIFIER_GET_INT_VEHICLE_TYPE_CLASS               = 2,
    PTX_CLASSIFIER_GET_FLOAT_VEHICLE_TYPE_PROB              = 3,
    PTX_CLASSIFIER_GET_INT_AT_VEHICLE_X_POINT               = 4,
    PTX_CLASSIFIER_GET_INT_VEHICLE_X_POINTS_SIZE            = 5,
    PTX_CLASSIFIER_GET_INT_AT_VEHICLE_Y_POINT               = 6,
    PTX_CLASSIFIER_GET_INT_VEHICLE_Y_POINTS_SIZE            = 7,

    PTX_CLASSIFIER_GET_INT_PROCESSING_TIME                  = 8,

    PTX_CLASSIFIER_GET_PTXSTRING_VEHICLE_COLOR_NAME         = 9,
    PTX_CLASSIFIER_GET_FLOAT_VEHICLE_COLOR_PROB             = 10,
    PTX_CLASSIFIER_GET_PTXSTRING_VEHICLE_BRAND_NAME         = 11,
    PTX_CLASSIFIER_GET_FLOAT_VEHICLE_BRAND_PROB             = 12,
    PTX_CLASSIFIER_GET_PTXSTRING_VEHICLE_MODEL_NAME         = 13,
    PTX_CLASSIFIER_GET_FLOAT_VEHICLE_MODEL_PROB             = 14,

    // Custom output
    PTX_CLASSIFIER_GET_PTXSTRING_OBJECT_TYPE_CLASS          = 15,
    PTX_CLASSIFIER_GET_FLOAT_OBJECT_TYPE_PROB               = 16,
    PTX_CLASSIFIER_GET_INT_AT_OBJECT_X_POINT                = 17,
    PTX_CLASSIFIER_GET_INT_OBJECT_X_POINTS_SIZE             = 18,
    PTX_CLASSIFIER_GET_INT_AT_OBJECT_Y_POINT                = 19,
    PTX_CLASSIFIER_GET_INT_OBJECT_Y_POINTS_SIZE             = 20,
    PTX_CLASSIFIER_GET_INT_OBJECT_ATTRIBUTES_SIZE           = 21,
    PTX_CLASSIFIER_GET_PTXDICT_AT_OBJECT_ATTRIBUTE          = 22,

    PTX_CLASSIFIER_GET_ENUM_MAX
} PtxClassifierResultType;

typedef enum PtxClassifierVehicleType {
    PTX_CLASSIFIER_VEHICLE_TYPE_UNKNOWN                     = 0,
    PTX_CLASSIFIER_VEHICLE_TYPE_CAR                         = 1,
    PTX_CLASSIFIER_VEHICLE_TYPE_MOTORCYCLE                  = 2,
    PTX_CLASSIFIER_VEHICLE_TYPE_TRUCK                       = 3,
    PTX_CLASSIFIER_VEHICLE_TYPE_BUS                         = 4,
    PTX_CLASSIFIER_VEHICLE_TYPE_PICKUP                      = 5,
    PTX_CLASSIFIER_VEHICLE_TYPE_SUV                         = 6,
    PTX_CLASSIFIER_VEHICLE_TYPE_VAN                         = 7,
    PTX_CLASSIFIER_VEHICLE_TYPE_TOW                         = 8,

    PTX_CLASSIFIER_VEHICLE_TYPE_ENUM_MAX
} PtxClassifierVehicleType;
```

ptx_classifier_api_local.h

```
#include "ptx/common/ptx_defines.h"
#include "ptx/common/ptx_codes.h"
#include "ptx/common/ptx_image.h"
#include "ptx/common/ptx_dict.h"
#include "ptx/classifier/ptx_classifier_common.h"

//============================================================================
// PtxClassifierHandle
//============================================================================
typedef struct PtxClassifierHandle PtxClassifierHandle;
```

```
//============================================================================
// FUNCTIONS
//============================================================================

/* Classifier functions */
PTXEXPORT int PTXAPI ptx_classifier_init();
PTXEXPORT int PTXAPI ptx_classifier_destroy();
PTXEXPORT int PTXAPI ptx_classifier_get_version(int* major, int* minor, int* release);
PTXEXPORT const char* PTXAPI ptx_classifier_get_SHA1();

/* Classifier handle */
PTXEXPORT PtxClassifierHandle* PTXAPI ptx_classifier_create_handle();
PTXEXPORT int PTXAPI ptx_classifier_free_handle(PtxClassifierHandle* handle);
PTXEXPORT int PTXAPI ptx_classifier_set_config(PtxClassifierHandle* handle, PtxDict*
config);

/* Classifier processing */
PTXEXPORT int PTXAPI ptx_classifier_classify(PtxClassifierHandle* handle, PtxImage* img,
PtxDict* results);

/* License */
PTXEXPORT int PTXAPI ptx_classifier_get_license_info(PtxProductLicenseInfo* license);
```

Tipos

enum PtxClassifierConfigs	
Descripción	Define la configuración de biblioteca disponible. Los ajustes se cambian mediante la función ptx_classifier_set_config.
Miembros	`PTX_CLASSIFIER_CONFIG_SCENE_TYPE`: tipo de configuración de escena a utilizar para la clasificación. `PTX_CLASSIFIER_CONFIG_VEHICLE_TYPE_MIN_PROB`: configuración de la probabilidad mínima de retorno permitida por el Clasificador. Las clasificaciones con una probabilidad inferior a esta serán descartadas internamente por la biblioteca y no serán devueltas al usuario. `PTX_CLASSIFIER_CONFIG_MODEL_TYPE`: configuración del tipo de modelo a utilizar. `PTX_CLASSIFIER_CONFIG_ENABLE_VEHICLE_CHARACTERISTICS`: configuración para habilitar/deshabilitar la lectura de características.

enum PtxClassifierSceneType	
Descripción	Define las escenas que se pueden configurar en el Clasificador, permitiendo que la biblioteca funcione mejor dependiendo de la instalación.
Miembros	`PTX_CLASSIFIER_SCENE_TYPE_CLOSEUP`: Tipo de escena en la que el vehículo ocupa más del 80% de la imagen. `PTX_CLASSIFIER_SCENE_TYPE_PANORAMIC`: Tipo de escena diurna donde los vehículos ocupan menos del 60% de la imagen. `PTX_CLASSIFIER_SCENE_TYPE_PANORAMIC_NIGHT`: Tipo de escena nocturna donde los vehículos ocupan menos del 60% de la imagen. `PTX_CLASSIFIER_SCENE_TYPE_WIDERANGE`: Tipo de escena donde los vehículos no tienen placas legibles 3 carriles o más.

Para vehículos que ocupan entre el 60% y el 80% de la imagen, es interesante probar cada tipo de escena y ver cuál funciona mejor.

enum PtxClassifierModelType	
Descripción	Define el tipo de modelo que se puede utilizar en Classifier.
Miembros	`PTX_CLASSIFIER_MODEL_TYPE_VEHICLES`: Tipo de modelo que funciona en la detección de vehículos. `PTX_CLASSIFIER_MODEL_TYPE_VEHICLES_EXTENDED`: Tipo de modelo que trabaja en la detección de vehículos, permitiendo la ampliación de clases para escenas panorámicas y diurnas (pickup, SUV, furgoneta y remolque). `PTX_CLASSIFIER_MODEL_TYPE_BMC`: Tipo de modelo que recibe detección de tipo de vehículo y clasifica marca, modelo y color. `PTX_CLASSIFIER_MODEL_TYPE_WIND_SHIELD`: Tipo de modelo que realiza la detección de parabrisas (En desarrollo, puede no estar disponible en su versión).

enum PtxClassifierObjExtras	
Descripción	Define las estructuras adicionales que se pueden colocar en la imagen para ser utilizadas por el *Classifier*.
Miembros	`PTX_CLASSIFIER_OBJ_UL_PT_X`: Establece el punto x superior izquierdo de un objeto que se definirá en la imagen (`PtxImage`). `PTX_CLASSIFIER_OBJ_UL_PT_Y`: Establece el punto y superior izquierdo de un objeto que se definirá en la imagen (`PtxImage`). `PTX_CLASSIFIER_OBJ_LR_PT_X`: Establece el punto x inferior derecho de un objeto que se definirá en la imagen (`PtxImage`). `PTX_CLASSIFIER_OBJ_LR_PT_Y`: Establece el punto y inferior derecho de un objeto que se definirá en la imagen (`PtxImage`). `PTX_CLASSIFIER_OBJ_TYPE_CLASS`: Define el tipo de clase de un objeto a definir en la imagen (`PtxImage`).

enum PtxClassifierResultType	
Descripción	Claves para solicitar la devolución de *Classifier* (tipo `PtxDict`) valores de propiedad del vehículo.
Miembros	`PTX_CLASSIFIER_GET_INT_RESULTS_SIZE`: Clave para solicitar el valor int del número de resultados devueltos. `PTX_CLASSIFIER_GET_PTXDICT_AT_RESULT`: Clave para solicitar el valor PtxDict de una posición determinada que contiene un único resultado. `PTX_CLASSIFIER_GET_INT_VEHICLE_TYPE_CLASS`: Clave para solicitar el valor int que representa el tipo de vehículo encontrado. `PTX_CLASSIFIER_GET_FLOAT_VEHICLE_TYPE_PROB`: Clave para solicitar el valor flotante que representa la confiabilidad en forma de probabilidad para el tipo de vehículo. `PTX_CLASSIFIER_GET_INT_VEHICLE_X_POINTS_SIZE`: Clave para solicitar el valor int que representa cuántos valores de x se utilizan para describir el contorno del objeto. `PTX_CLASSIFIER_GET_INT_AT_VEHICLE_X_POINT`: Clave para solicitar el valor int que corresponde a un único valor x perteneciente al contorno del objeto. `PTX_CLASSIFIER_GET_INT_VEHICLE_Y_POINTS_SIZE`: Clave para solicitar el valor int que representa cuántos valores y se utilizan para describir el contorno del objeto. `PTX_CLASSIFIER_GET_INT_AT_VEHICLE_Y_POINT`: Clave para solicitar el valor int que corresponde a un único valor y perteneciente al contorno del objeto. `PTX_CLASSIFIER_GET_INT_PROCESSING_TIME`: Clave para solicitar el valor int que corresponde al tiempo de procesamiento del Clasificador. `PTX_CLASSIFIER_GET_PTXSTRING_VEHICLE_COLOR_NAME`: Clave para solicitar el valor de la cadena que corresponde al nombre del color del vehículo. `PTX_CLASSIFIER_GET_FLOAT_VEHICLE_COLOR_PROB`: Clave para solicitar el valor flotante que corresponde a la probabilidad de color del vehículo. `PTX_CLASSIFIER_GET_PTXSTRING_VEHICLE_BRAND_NAME`: Clave para solicitar el valor de la cadena que corresponde a la marca del vehículo.

	`PTX_CLASSIFIER_GET_FLOAT_VEHICLE_BRAND_PROB`: Clave para solicitar el valor float que corresponde a la probabilidad de la marca del vehículo. `PTX_CLASSIFIER_GET_PTXSTRING_VEHICLE_MODEL_NAME`: Clave para solicitar el valor de la cadena que corresponde al modelo de vehículo. `PTX_CLASSIFIER_GET_FLOAT_VEHICLE_MODEL_PROB`: Clave para solicitar el valor float que corresponde a la probabilidad del modelo de vehículo. `PTX_CLASSIFIER_GET_PTXSTRING_OBJECT_TYPE_CLASS`: Clave para solicitar el valor de cadena que corresponde al nombre de clase del objeto. `PTX_CLASSIFIER_GET_FLOAT_OBJECT_TYPE_PROB`: Clave para solicitar el valor flotante que corresponde a la probabilidad de la clase de objeto. `PTX_CLASSIFIER_GET_INT_AT_OBJECT_X_POINT`: Clave para solicitar el valor int que corresponde a un único valor x perteneciente al contorno del objeto. `PTX_CLASSIFIER_GET_INT_OBJECT_X_POINTS_SIZE`: Clave para solicitar el valor int que representa cuántos valores de x se utilizan para describir el contorno del objeto. `PTX_CLASSIFIER_GET_INT_AT_OBJECT_Y_POINT`: Clave para solicitar el valor int que corresponde a un único valor y perteneciente al contorno del objeto. `PTX_CLASSIFIER_GET_INT_OBJECT_Y_POINTS_SIZE`: Clave para solicitar el valor int que representa cuántos valores y se utilizan para describir el contorno del objeto. `PTX_CLASSIFIER_GET_INT_OBJECT_ATTRIBUTES_SIZE`: Clave para solicitar el valor int que corresponde al número de atributos del objeto. `PTX_CLASSIFIER_GET_PTXDICT_AT_OBJECT_ATTRIBUTE`: Clave para solicitar el valor PtxDict de una posición determinada que contiene un solo atributo.

enum PtxClassifierVehicleType	
Descripción	Campos que representan las posibles devoluciones de la clave `PTX_CLASSIFIER_GET_INT_VEHICLE_TYPE_CLASS,` es decir, los posibles tipos de vehículos devueltos por la biblioteca.
Miembros	***`PTX_CLASSIFIER_VEHICLE_TYPE_UNKNOWN:`*** Tipo de vehículo que no se pudo categorizar ***`PTX_CLASSIFIER_VEHICLE_TYPE_CAR:`*** Vehículo tipo coche ***`PTX_CLASSIFIER_VEHICLE_TYPE_MOTORCYCLE:`*** Vehículo tipo motocicleta ***`PTX_CLASSIFIER_VEHICLE_TYPE_TRUCK:`*** Vehículo tipo camión ***`PTX_CLASSIFIER_VEHICLE_TYPE_BUS:`*** Vehículo tipo autobús ***`PTX_CLASSIFIER_VEHICLE_TYPE_PICKUP:`*** Vehículo tipo pickup ***`PTX_CLASSIFIER_VEHICLE_TYPE_SUV:`*** Vehículo tipo SUV ***`PTX_CLASSIFIER_VEHICLE_TYPE_VAN:`*** Vehículo tipo furgoneta ***`PTX_CLASSIFIER_VEHICLE_TYPE_TOW:`*** Vehículo tipo remolque

struct PtxClassifierHandle	
Descripción	Estructura que contiene las configuraciones internas y los estados de la biblioteca.

Métodos

ptx_classifier_init	
Prototipo de función	`int ptx_classifier_init();`
Descripción	Función utilizada para inicializar la biblioteca después de cargarla.
Parámetros	Ninguno
Devolver	Un número entero con código de retorno de función.

ptx_classifier_destroy	
Prototipo de función	`int ptx_classifier_destroy();`
Descripción	Función utilizada para descargar la biblioteca.

Parámetros	Ninguno
Devolver	Un número entero con código de retorno de función.

ptx_classifier_get_version	
Prototipo de función	`int ptx_classifier_get_version(int* major, int* minor, int* release)`
Descripción	Función utilizada para consultar la versión de la biblioteca. Classifier sigue un sistema de versiones semánticas de lanzamiento mayor.menor. El número mayor aumenta cuando la versión tiene una interrupción de API en relación con la versión anterior. Si no hay ninguna interrupción de API y se incluye nueva funcionalidad, se aumenta el número menor. Si no hay una interrupción de la API o se incluye una nueva funcionalidad, el número de versión aumenta (este es el caso de las correcciones de errores y los pequeños cambios).
Parámetros	`int *major, int *minor, int *release`: punteros a variables enteras donde se escribirán los números que componen la versión.
Devolver	Un número entero con código de retorno de función.

ptx_classifier_get_SHA1	
Prototipo de función	`const char* ptx_classifier_get_SHA1();`
Descripción	Función utilizada para comprobar el hash SHA1 de la compilación de la biblioteca. Pumatronix utiliza esta cadena para el seguimiento de compilaciones.
Parámetros	Ninguno
Devolver	Devuelve un puntero al comienzo de una cadena que termina en \0 y que contiene el hash SHA1 de la compilación

ptx_classifier_create_handle	
Prototipo de función	`PtxClassifierHandle* ptx_classifier_create_handle();`
Descripción	Función utilizada para asignar memoria para la configuración de la biblioteca. En el caso de uso de subprocesos múltiples, cada subproceso debe llamar a ptx_classifier_create_handle y usar su propio `PtxClassifierHandle`. El mismo `PtxClassifierHandle` se puede usar tantas veces como sea necesario, siempre que solo se use un subproceso a la vez. La configuración de la biblioteca, realizada a través de la función ptx_classifier_set_config, se almacena en `PtxClassifierHandle`. Por tanto, cada `PtxClassifierHandle` puede tener una configuración diferente. Esto puede resultar útil, por ejemplo, cuando desea utilizar la biblioteca Clasificador para procesar imágenes de diferentes cámaras y desea aplicar diferentes configuraciones a imágenes de diferentes cámaras.
Parámetros	Ninguno
Devolver	Devuelve un puntero a una estructura de tipo PtxClassifierHandle que se utilizará en llamadas de función posteriores.

ptx_classifier_free_handle	
Prototipo de función	`int ptx_classifier_free_handle(PtxClassifierHandle* handle);`
Descripción	Función utilizada para liberar la memoria asignada al objeto PtxClassifierHandle.
Parámetros	`PtxClassifierHandle* handle`: Puntero al identificador de tipo PtxClassifierHandle.
Devolver	Un número entero con código de retorno de función.

ptx_classifier_set_config	
Prototipo de función	`int ptx_classifier_set_config(PtxClassifierHandle* handle, PtxDict* config);`
Descripción	Función utilizada para aplicar la configuración realizada a través de un campo PtxDict.
Parámetros	`PtxClassifierHandle* handle`: Puntero al identificador de tipo PtxClassifierHandle `PtxDict* config`: Puntero a objeto de tipo PtxDict
Devolver	Un número entero con código de retorno de función.

ptx_classifier_classify	
Prototipo de función	`int ptx_classifier_classify(PtxClassifierHandle* handle, PtxImage* img, PtxDict* results);`
Descripción	Función utilizada para procesar y extraer información de la imagen y devolverla a través del parámetro `results`. **Esta función puede llevar bastante tiempo, ya que puede tardar cientos de milisegundos en completarse o incluso más, dependiendo de la CPU utilizada.**
Parámetros	`PtxClassifierHandle* handle`: Puntero al identificador de tipo PtxClassifierHandle `PtxImage* img`: Puntero a imagen de tipo PtxImage `PtxDict* results`: Puntero a resultados de tipo PtxDict
Devolver	Un número entero con código de retorno de función.

ptx_classifier_get_license_info	
Prototipo de función	`int ptx_classifier_get_license_info(PtxProductLicenseInfo* license);`
Descripción	Función utilizada para solicitar información de licencia sobre el producto Classifier.
Parámetros	`PtxProductLicenseInfo* license:` Puntero al objeto de tipo PtxProductLicenseInfo
Devolver	Un número entero con código de retorno de función.

API ptx_common C/C++

La biblioteca ptx_common implementa estructuras de datos, códigos de error y otras definiciones comúnmente utilizadas por los productos Pumatronix en C. Su API está definida en el archivo `ptx_common.h`, que a su vez incluye otros encabezados.

ptx_codes.h

```
//============================================================================
// RETURN CODES
//============================================================================
enum PtxCommonReturnCode {

     /* API CALL RETURN CODES */
     /* SUCCESS */
     PTX_SUCCESS                                = 0,

     /* BASIC ERRORS */
     PTX_FILE_NOT_FOUND                         = 1,
     PTX_INVALID_IMAGE                          = 2,
     PTX_INVALID_IMAGE_TYPE                     = 3,
```

```
	PTX_INVALID_PARAMETER                       = 4,
	PTX_COUNTRY_NOT_SUPPORTED                   = 5,
	PTX_API_CALL_NOT_SUPPORTED                  = 6,
	PTX_INVALID_ROI                             = 7,
	PTX_INVALID_HANDLE                          = 8,
	PTX_API_CALL_HAS_NO_EFFECT                  = 9,
	PTX_INVALID_IMAGE_SIZE                      = 10,
	PTX_MODEL_UNAVALIABLE                       = 11,

	/* LICENSE ERRORS */
	PTX_LICENSE_INVALID                         = 16,
	PTX_LICENSE_EXPIRED                         = 17,
	PTX_LICENSE_MAX_THREADS_EXCEEDED            = 18,
	PTX_LICENSE_UNTRUSTED_RTC                   = 19,
	PTX_LICENSE_MAX_CONNS_EXCEEDED              = 20,
	PTX_LICENSE_UNAUTHORIZED_PRODUCT            = 21,

	/* NETWORK ERRORS */
	PTX_CONNECT_FAILED                          = 100,
	PTX_SOCKET_DISCONNECT                       = 101,
	PTX_SOCKET_QUEUE_TIMEOUT                    = 202,
	PTX_SOCKET_QUEUE_FULL                       = 103,
	PTX_SOCKET_IO_ERROR                         = 104,
	PTX_SOCKET_WRITE_FAILED                     = 105,
	PTX_SOCKET_READ_TIMEOUT                     = 106,
	PTX_INVALID_RESPONSE                        = 107,
	PTX_HANDLE_QUEUE_FULL                       = 108,
	PTX_INVALID_REQUEST                         = 109,
	PTX_INVALID_MESSAGE                         = 110,
	PTX_INVALID_STREAM_FRAME                    = 209,
	PTX_FRAME_QUEUE_FULL                        = 211,
	PTX_LAST_FRAME_UNAVAILABLE                  = 212,
	PTX_SERVER_CONN_LIMIT_REACHED               = 213,
	PTX_SERVER_VERSION_NOT_SUPPORTED            = 214,

	/* MJPEG ERRORS */
	PTX_MJPEG_ERROR_BASE                        = 1000,
	PTX_MJPEG_HTTP_HEADER_OVERFLOW              = 1001,
	PTX_MJPEG_HTTP_RESPONSE_NOT_OK              = 1002,
	PTX_MJPEG_HTTP_CONTENT_TYPE_ERROR           = 1003,
	PTX_MJPEG_HTTP_CONTENT_LENGTH_ERROR         = 1004,
	PTX_MJPEG_HTTP_FRAME_BOUNDARY_NOT_FOUND     = 1005,
	PTX_MJPEG_CONNECTION_CLOSED = 1006,
	PTX_MJPEG_CONNECT_FAILED                    = 1007,

	/* HARDWARE ERRORS - returned by the process */
	PTX_HW_FPGA_INIT_FAILED                     = 2000,
	PTX_HW_FPGA_LOCK_FAILED                     = 2001,

	/* OTHERS */
	PTX_UNKNOWN_ERROR                           = 99999
};
typedef struct PtxProductLicenseInfo
{
	uint64_t serial;
	char customer[64];
	int maxThreads;
	int maxConnections;
```

```
    int state;
    int ttl;
} PtxProductLicenseInfo;
PTXEXPORT const char* PTXAPI ptx_common_get_SHA1();
);
```

Tipos

enum PtxCommonReturnCode	
Descripción	Establece códigos de error de la biblioteca ptx_common.
Miembros	Los distintos miembros de esta enumeración son devueltos por funciones en diferentes tipos de situaciones: éxito, imagen no válida, parámetro no válido, errores de licencia, errores de red, etc.

enum PtxProductLicenseInfo	
Descripción	Struct utilizado para almacenar información sobre la licencia utilizada por la biblioteca
Miembros	• `uint64_t serial`: número de serie de licencia • `char customer[64]`: nombre del cliente que compró la licencia • `int maxThreads`: número máximo de subprocesos de procesamiento habilitados • `int maxConnections`: número máximo de conexiones paralelas habilitadas • `int state`: estado de la licencia - ver *PtxCommonReturnCode* • `int ttl`: Tiempo de vida en horas para licencias tipo RTC. Este campo tiene el valor -1 si la licencia no tiene vencimiento

Métodos

ptx_common_get_SHA1	
Prototipo de Función	`const char* PTXAPI ptx_common_get_SHA1();`
Descripción	Función utilizada para comprobar el hash SHA1 de la compilación de la biblioteca ptx_common. Pumatronix utiliza esta cadena para el seguimiento de compilaciones.
Parámetros	Ninguno
Devolver	Devuelve un puntero al comienzo de una cadena que termina en \0 que contiene el hash SHA1 de la compilación.

ptx_dict.h

Este archivo define una estructura de datos de diccionario, que se utiliza para almacenar relaciones clave-valor. La clave siempre es de tipo int. Cada producto Pumatronix que utiliza la biblioteca ptx_common enumera en su encabezado qué claves utiliza.

```
//============================================================================
// PtxDict
//============================================================================
typedef struct PtxDict PtxDict;

/* lifecycle */
PTXEXPORT PtxDict* PTXAPI ptxdict_create();
PTXEXPORT int PTXAPI ptxdict_free(PtxDict* ptx_dict);

/* Setters */
PTXEXPORT int PTXAPI ptxdict_set_int(PtxDict* ptx_dict, int key, int value);
PTXEXPORT int PTXAPI ptxdict_set_float(PtxDict* ptx_dict, int key, float value);

/* Getters */
```

```
PTXEXPORT int PTXAPI ptxdict_get_int(PtxDict* ptx_dict, int key, int* value);
PTXEXPORT int PTXAPI ptxdict_get_float(PtxDict* ptx_dict, int key, float* value);
PTXEXPORT int PTXAPI ptxdict_get_ptxstring(PtxDict* ptx_dict, int key, PtxString* value);
PTXEXPORT int PTXAPI ptxdict_get_int_at(PtxDict* ptx_dict, int key, int index, int* value);
PTXEXPORT int PTXAPI ptxdict_get_ptxdict_at(PtxDict* ptx_dict, int key, int index, PtxDict*
value);
```

Tipos

struct PtxDict	
Descripción	Struct archivo opaco utilizado para almacenar el diccionario.
Miembros	Ninguno, ya que es una estructura opaca.

Métodos

ptxdict_create	
Prototipo de función	`PtxDict* PTXAPI ptxdict_create();`
Descripción	Función utilizada para asignar memoria para un diccionario.
Parámetros	Ninguno
Devolver	Devuelve un puntero a un tipo de estructura [PtxDict](#struct-ptxdict] que se utilizará en llamadas de función posteriores.

ptxdict_free	
Prototipo de función	`int ptxdict_free(PtxDict* ptx_dict);`
Descripción	Función utilizada para liberar la memoria asignada al objeto de tipo *PtxDict*.
Parámetros	`PtxDict* dict`: Puntero al objeto de tipo PtxDict cuya memoria se liberará.
Devolver	Un número entero con código de retorno de función.

ptxdict_set_int	
Prototipo de función	`int ptxdict_set_int(PtxDict* ptx_dict, int key, int value);`
Descripción	Función utilizada para almacenar un par clave-valor.
Parámetros	`PtxDict* dict`: Puntero al objeto de tipo PtxDict donde se almacenará el par clave-valor. `int key`: clave de tipo int. `int value`: valor de tipo int que se almacenará como asociado con el `key`.
Devolver	Un número entero con código de retorno de función.

ptxdict_set_float	
Prototipo de función	`int ptxdict_set_float(PtxDict* ptx_dict, int key, float value);`
Descripción	Función utilizada para almacenar un par clave-valor.
Parámetros	`PtxDict* dict`: Puntero al objeto de tipo PtxDict donde se almacenará el par clave-valor. `int key`: clave de tipo int. `float value`: valor de tipo int que se almacenará como asociado con el `key`

Devolver	Un número entero con código de retorno de función.

ptxdict_get_int	
Prototipo de función	`int ptxdict_get_int(PtxDict* ptx_dict, int key, int* value);`
Descripción	Función utilizada para solicitar un valor asociado a una clave.
Parámetros	`PtxDict* dict`: Puntero a objeto de tipo PtxDict. `int key`: clave de tipo int cuyo valor desea solicitar. `int* value`: valor de tipo int que se almacenará como asociado con la clave, si existe.
Devolver	Un número entero con código de retorno de función.

ptxdict_get_float	
Prototipo de función	`int ptxdict_get_float(PtxDict* ptx_dict, int key, float* value);`
Descripción	Función utilizada para solicitar un valor asociado a una clave.
Parámetros	`PtxDict* dict`: Puntero al objeto de tipo *PtxDict*. `int key`: clave de tipo int cuyo valor desea solicitar. `float value`: puntero a int donde se escribirá el valor asociado a la clave, si existe.
Devolver	Un número entero con código de retorno de función.

ptxdict_get_ptxstring	
Prototipo de función	`int ptxdict_get_ptxstring(PtxDict* ptx_dict, int key, PtxString* value);`
Descripción	Función utilizada para solicitar una cadena asociada a una clave.
Parámetros	`PtxDict* dict`: Puntero al objeto de tipo *PtxDict*. `int key`: clave de tipo int cuyo valor desea solicitar. `PtxString* value`: puntero al objeto de tipo PtxString donde se escribirá el valor asociado a la clave, si existe.
Devolver	Un número entero con código de retorno de función.

ptxdict_get_int_at	
Prototipo de función	`int ptxdict_get_int_at(PtxDict* ptx_dict, int key, int index, int* value);`
Descripción	Función utilizada para solicitar el valor en un índice dado de un vector asociado con una clave.
Parámetros	`PtxDict* dict`: Puntero al objeto de tipo *PtxDict*. `int key`: clave de tipo int cuyo valor desea solicitar. `int index`: índice deseado del vector asociado con `key.` `int* value`: puntero a flotante donde se escribirá el valor, si lo hay.
Devolver	Un número entero con código de retorno de función.

ptxdict_get_ptxdict_at	
Prototipo de función	`int ptxdict_get_ptxdict_at(PtxDict* ptx_dict, int key, int index, PtxDict* value);`

Descripción	Función utilizada para solicitar el valor en un índice dado de un vector asociado con una clave.
Parámetros	`PtxDict* dict`: Puntero al objeto de tipo *PtxDict*. `int key`: clave de tipo int cuyo valor desea solicitar. `int index`: índice deseado del vector asociado con `key.` `PtxDict* value`: puntero a PtxDict donde se escribirá el valor, si existe.
Devolver	Un número entero con código de retorno de función.

ptx_image.h

Este archivo define una estructura para cargar imágenes. Admite imágenes estructuradas (jpg y bmp) e imágenes RAW.

```
//============================================================================
// Types
//============================================================================

typedef struct PtxImage PtxImage;

/* Raw image pixel format */
typedef enum PtxImagePixelFmt {
      PTX_IMG_FMT_XRGB_8888 = 0,
      PTX_IMG_FMT_RGB_888 = 1,
      PTX_IMG_FMT_LUMA = 2,
      PTX_IMG_FMT_YUV420 = 3,
      PTX_IMG_FMT_YUV_NV12 = 4
} PtxImagePixelFmt;

//============================================================================
// Functions
//============================================================================

/* lifecycle */
PTXEXPORT PtxImage* PTXAPI ptximage_create();
PTXEXPORT int PTXAPI ptximage_free(PtxImage* img);

/* load */
PTXEXPORT int PTXAPI ptximage_load_from_file(PtxImage* img, const char* filename);
PTXEXPORT int PTXAPI ptximage_load_from_memory(PtxImage* img, const uint8_t* buffer, int
bufferSize);
PTXEXPORT int PTXAPI ptximage_load_from_raw_format(PtxImage* img, const uint8_t* buffer,
int width, int height, int stride, PtxImagePixelFmt fmt)

/* setters */
PTXEXPORT int PTXAPI ptximage_append_extra(PtxImage* img, PtxDict* extra);
PTXEXPORT int PTXAPI ptximage_clear_extras(PtxImage* img);
```

Tipos

struct PtxImage	
Descripción	Struct opaco usado para cargar imágenes.
Miembros	Ninguno, ya que es una estructura opaca.

enum PtxImagePixelFmt	
Descripción	Enumeración de tipos de imágenes RAW que se pueden cargar.
Miembros	Ninguno, ya que es una estructura opaca.

Métodos

ptximage_create	
Prototipo de función	`PtxImage* ptximage_create();`
Descripción	Función utilizada para asignar memoria para una estructura de imagen.
Parámetros	Ninguno
Devolver	Devuelve un puntero a un tipo de estructura [PtxImage](#struct-ptximage] que se utilizará en llamadas de función posteriores.

ptximage_free	
Prototipo de función	`int ptximage_free(PtxImage* img);`
Descripción	Función utilizada para liberar memoria asignada a un objeto de tipo *PtxImage*.
Parámetros	`PtxImage* img`: Puntero al objeto de tipo PtxImage cuya memoria se liberará.
Devolver	Un número entero con código de retorno de función.

ptximage_load_from_file	
Prototipo de función	`int ptximage_load_from_file(PtxImage* img, const char* filename);`
Descripción	Función utilizada para cargar una imagen estructurada (jpg o bmp) desde un archivo.
Parámetros	`PtxImage* img`: Puntero al objeto de tipo PtxImage donde se almacenará la imagen en la memoria. `const char* filename`: Archivo que contiene una imagen estructurada para cargar.
Devolver	Un número entero con código de retorno de función.

ptximage_load_from_memory	
Prototipo de función	`int ptximage_load_from_memory(PtxImage* img, const uint8_t* buffer, int bufferSize);`
Descripción	Función utilizada para cargar una imagen estructurada (jpg o bmp) desde un buffer de memoria.
Parámetros	`PtxImage* img`: Puntero al objeto de tipo PtxImage donde se almacenará la imagen en la memoria. `const uint8* buffer`: Buffer que contiene la imagen estructurada que se va a cargar. `int bufferSize`: Tamaño del búfer en bytes.
Devolver	Un número entero con código de retorno de función.

ptximage_load_from_raw_format	
Prototipo de función	`int ptximage_load_from_raw_format(PtxImage* img, const uint8_t* buffer, int width, int height, int stride, PtxImagePixelFmt fmt)`
Descripción	Función utilizada para cargar una imagen RAW desde un buffer de memoria. Los tipos de imágenes admitidos están definidos por*PtxImagePixelFmt*.

Parámetros	`PtxImage* img`: Puntero al objeto de tipo PtxImage donde se almacenará la imagen en la memoria. `const uint8* buffer`: Búfer que contiene la imagen RAW que se va a cargar. `int width`: Ancho de la imagen en píxeles. `int height`: Altura de la imagen en píxeles. `int stride`: Número de bytes entre una línea y la siguiente línea de la imagen. `PtxImagePixelFmt fmt`: Tipo de imagen RAW, según *PtxImagePixelFmt*.
Devolver	Un número entero con código de retorno de función.

ptximage_append_extra	
Prototipo de función	`int ptximage_append_extra(PtxImage* img, PtxDict* extra);`
Descripción	Función para agregar un PtxDict adicional a la estructura de la imagen.
Parámetros	`PtxImage* img`: Puntero al objeto de tipo *PtxImage*. `PtxDict* extra`: Puntero al extra PtxDict que se agregará a la imagen.
Devolver	Un número entero con código de retorno de función.

ptximage_clear_extras	
Prototipo de función	`int ptximage_clear_extras(PtxImage* img);`
Descripción	Función utilizada para borrar extras añadidos.
Parámetros	`PtxImage* img`: Puntero al objeto de tipo *PtxImage*.
Devolver	Un número entero con código de retorno de función.

ptx_string.h

```
Este arquivo define uma estrutura para ler textos.
//============================================================================
// PtxString
//============================================================================
typedef struct PtxString PtxString;

/* lifecycle */
PTXEXPORT PtxString* PTXAPI ptxstring_create();
PTXEXPORT int PTXAPI ptxstring_free(PtxString* ptx_string);

/* Setters */
PTXEXPORT int PTXAPI ptxstring_set(PtxString* ptx_string, const char* str);

/* Getters */
PTXEXPORT const char* PTXAPI ptxstring_get(PtxString* ptx_string);
```

Tipos

struct PtxString	
Descripción	Struct opaco utilizado para contener textos.
Miembros	Ninguno, ya que es una estructura opaca.

Métodos

ptxstring_create	
Prototipo de función	`PtxString* ptxstring_create();`
Descripción	Función utilizada para asignar memoria para una estructura de texto.
Parámetros	Ninguno
Devolver	Devuelve un puntero a una estructura PtxString que se utilizará en llamadas de función posteriores.

ptxstring_free	
Prototipo de función	`int ptxstring_free(PtxString* ptx_string);`
Descripción	Función utilizada para liberar memoria asignada a un objeto de tipo *PtxString*.
Parámetros	`PtxString* ptx_string`: Puntero al objeto de tipo PtxString cuya memoria se liberará.
Devolver	Un número entero con código de retorno de función.

ptxstring_set	
Prototipo de función	`int ptxstring_set(PtxString* ptx_string, const char* str);`
Descripción	Función utilizada para copiar los caracteres de un const char* en la estructura *PtxString*.
Parámetros	`PtxString* ptx_string`: Puntero al objeto de tipo *PtxString*. `const char* str`: Puntero al texto en formato const char*.
Devolver	Un número entero con código de retorno de función.

ptxstring_get	
Prototipo de función	`const char* ptxstring_get(PtxString* ptx_string);`
Descripción	Función utilizada para recuperar los caracteres de un const char* desde dentro de la structura *PtxString*.
Parámetros	`PtxString* ptx_string`: Puntero al objeto de tipo *PtxString*.
Devolver	Puntero al texto en formato const char*.


PUMATRONIX
www.pumatronix.com