PUMATRONIX

CLASSIFIER

CLASSIFIER

SOFTWARE PARA RECONHECIMENTO AUTOMÁTICO DE CARACTERÍSTICAS VEICULARES

Integração

Pumatronix Equipamentos Eletrônicos Ltda.

Rua Bartolomeu Lourenço de Gusmão, 1970. Curitiba, Brasil

Visite nosso website https://www.pumatronix.com

Envie comentários sobre este documento no e-mail suporte@pumatronix.com

Informações contidas neste documento estão sujeitas a mudança sem aviso prévio.

A Pumatronix se reserva o direito de modificar ou melhorar este material sem obrigação de notificação das alterações ou melhorias.

A Pumatronix assegura permissão para download e impressão deste documento, desde que a cópia eletrônica ou física deste documento contenha o texto na íntegra. Qualquer alteração neste conteúdo é estritamente proibida.

Histórico de Alterações

Data	Revisão	Conteúdo atualizado
23/07/2024	1.0	Revisão do layout e formatação geral do documento; Conteúdo referente à versão 1.16.0 do produto

Visão Geral

Este documento tem o objetivo de orientar o desenvolvedor na aplicação da biblioteca de software *Classifier* responsável pela classificação dos tipos de veículos a partir da análise de imagens e aplicável em softwares compatíveis com a biblioteca. Neste documento estão detalhadas as opções de configuração do kit de desenvolvimento de software (SDK) e das APIs disponibilizadas.

Sumário

1. Descompactando o SDK

O SDK do *Classifier* é distribuído através de um arquivo compactado no formato 7zip. A descompactação deste arquivo requer uma senha, fornecida junto com o e-mail contendo as instruções de download deste SDK. Caso você tenha problemas com a descompactação do SDK contate o suporte pelo e-mail contato@pumatronix.com.br ou WhatsApp +55 (41) 9203-8327.

Para descompactar o SDK em ambiente Linux, utilize o seguinte comando a partir de um terminal (substitua os campos <Classifier_PC_LINUX_64_vX.Y.Z.7z> e pela senha fornecida por e-mail e pelo nome correto do pacote).

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

Permissões do hardkey

Esta configuração só é necessária para a versão Linux do SDK.

Para o correto funcionamento do hardkey USB no Linux, as permissões de acesso do udev devem ser alteradas. Adicione a seguinte linha:

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

ao final do arquivo correspondente a sua distribuição Linux:

```
Centos 5.2/5.4:          /etc/udev/rules.d/50-udev.rules
Centos 6.0 em diante:    /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 em diante:   /lib/udev/rules.d/50-udev-default.rules
openSUSE 11.2 em diante: /lib/udev/rules.d/50-udev-default.rules
```

Caso a distribuição seja Debian, adicione as linhas:

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

ao final do arquivo:

```
Debian 6.0 em diante: /lib/udev/rules.d/91-permissions.rules
```

Para instruções de como habilitar o hardkey em outras distribuições Linux, entre em contato com o contato@pumatronix.com.br ou WhatsApp +55 (41) 9203-8327.

Variáveis de ambiente

Configuração do LD_LIBRARY_PATH

Esta configuração só é necessária para a versão Linux do SDK. Para a versão Windows, faça uma cópia da DLL para a pasta onde está o executável ou para a pasta system32

Para que a aplicação seja capaz de encontrar as bibliotecas do *Classifier* deve-se adicionar ao PATH o caminho do diretório lib do SDK. Em ambiente Linux isto pode ser feito através da variável de ambiente `LD_LIBRARY_PATH`, como exemplificado abaixo (substitua pelo caminho absoluto da instalação do SDK).

```
export LD_LIBRARY_PATH=/lib
```

2. Estrutura do SDK

O *Classifier* não foi projetado como uma aplicação standalone e sim como uma biblioteca a ser integrada em outras aplicações. Dessa forma, o SDK do *Classifier* é distribuído como um conjunto de shared libraries (.so e .dll) e seus headers em linguagem C.

O SDK acompanha ainda uma aplicação de exemplo (código fonte e binário pré-compilado) que pode ser utilizada como base na implementação de uma aplicação que utiliza o *Classifier*. A seção Exemplo Básico deste manual traz também um passo-a-passo de como implementar uma aplicação.

Árvore de arquivos versão 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
```

Árvore de arquivos versão 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. Arquitetura de software

O *Classifier* possui uma arquitetura simples. Todas as chamadas da API são síncronas, ou seja, bloqueiam até que o processamento esteja concluído. A função principal é a *ptx_classifier_classify*, que recebe uma imagem previamente carregada e um handle contendo a configuração e, ao final do processamento, preenche uma estrutura de resultados.

A configuração da biblioteca consiste em: tipo de cena (panorâmico ou fechado), e a mínima probabilidade para que uma detecção seja considerada um veículo válido.

Os resultados consistem em um conjunto de veículos detectados, cada um com os seguintes dados associados: classe do veículo, confiabilidade da classificação na forma de probabilidade, e coordenadas na imagem do retângulo que contém o veículo.

4. Exemplo de uso da API

Para um exemplo completo, veja o arquivo `sample/src/PtxClassifierSample.c` do SDK.

Para compilar a aplicação exemplo com o gcc e executar, rode a seguinte sequência de comandos a partir de um terminal no Linux:

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

Assim que a aplicação for iniciada, deverá aparecer na saída do 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 usuário

A biblioteca *Classifier* exporta uma API para o reconhecimento automático de características de veículos.

Por padrão, as linguagens suportadas pela API que acompanham o SDK são C, C++ e Java. Wrappers em Python, C# e Delphi podem ser fornecidos sob demanda. Em caso de dúvidas ou suporte a outras linguagens, envie um e-mail para contato@pumatronix.com.br ou WhatsApp +55 (41) 9203-8327.

Para padronizar estruturas comuns utilizadas por seus produtos, a Pumatronix criou a biblioteca ptx_common, que implementa em C estruturas de dados, códigos de erro e outras definições comumente utilizadas. A biblioteca ptx_common é necessária para uso do Classifier e está inclusa no SDK. A documentação de sua API pode ser encontrada na seção API ptx_common C/C++.

API Classifier C/C++

A API (Application Programming Interface) nativa da biblioteca está escrita em linguagem C, o que facilita a criação de bindings para uso em outras linguagens. Toda a API C está disponível através de um conjunto de headers dentro da pasta *include* do SDK.

A API contém os tipos, definições e funções para o processamento das imagens. Ela está definida no arquivo `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	
Descrição	Define as configurações disponíveis da biblioteca. As configurações são alteradas através da função ptx_classifier_set_config.
Membros	`PTX_CLASSIFIER_CONFIG_SCENE_TYPE`: tipo de configuração de cena a ser usado para classificação. `PTX_CLASSIFIER_CONFIG_VEHICLE_TYPE_MIN_PROB`: configuração da probabilidade mínima admitida de retorno por parte do Classifier. Classificações com probabilidade menor que essa serão descartadas internamente pela biblioteca e não serão retornadas ao usuário. `PTX_CLASSIFIER_CONFIG_MODEL_TYPE`: configuração do tipo de modelo a ser utilizado. `PTX_CLASSIFIER_CONFIG_ENABLE_VEHICLE_CHARACTERISTICS`: configuração para habilitar/desabilitar a leitura de características.

enum PtxClassifierSceneType	
Descrição	Define as cenas que podem ser configuradas no *Classifier*, permitindo um melhor funcionamento da biblioteca dependendo da instalação.
Membros	`PTX_CLASSIFIER_SCENE_TYPE_CLOSEUP`: Tipo de cena em que o veículo ocupa mais de 80% da imagem. `PTX_CLASSIFIER_SCENE_TYPE_PANORAMIC`: Tipo de cena diurna onde os veículos ocupam menos de 60% da imagem. `PTX_CLASSIFIER_SCENE_TYPE_PANORAMIC_NIGHT`: Tipo de cena noturna onde os veículos ocupam menos de 60% da imagem. `PTX_CLASSIFIER_SCENE_TYPE_WIDERANGE`: Tipo de cena onde os veículos não tem placa legível 3 pistas ou mais.

Para veículos ocupando entre 60% e 80% da imagem é interessante experimentar cada um dos tipos de cena e verificar qual delas apresenta melhor desempenho.

enum PtxClassifierModelType	
Descrição	Define o tipo de modelo que pode ser usado no *Classifier*.
Membros	`PTX_CLASSIFIER_MODEL_TYPE_VEHICLES`: Tipo de modelo que trabalha na detecção de veículos. `PTX_CLASSIFIER_MODEL_TYPE_VEHICLES_EXTENDED`: Tipo de modelo que trabalha na detecção de veículos habilitando a extensão de classes para cenas panorâmicas e de dia (pickup, suv, van e reboque). `PTX_CLASSIFIER_MODEL_TYPE_BMC`: Tipo de modelo que recebe a detecção de tipo de veículo e classifica marca, modelo e cor. `PTX_CLASSIFIER_MODEL_TYPE_WIND_SHIELD`: Tipo de modelo que faz a detecção de para-brisa (Em desenvolvimento, pode não estar disponível em sua versão).

enum PtxClassifierObjExtras	
Descrição	Define as estruturas extras que podem ser colocadas na imagem para serem utilizadas pelo *Classifier*.
Membros	`PTX_CLASSIFIER_OBJ_UL_PT_X`: Define o ponto x superior esquerdo de um objeto a ser definido na imagem (`PtxImage`). `PTX_CLASSIFIER_OBJ_UL_PT_Y`: Define o ponto y superior esquerdo de um objeto a ser definido na imagem (`PtxImage`). `PTX_CLASSIFIER_OBJ_LR_PT_X`: Define o ponto x inferior direito de um objeto a ser definido na imagem (`PtxImage`). `PTX_CLASSIFIER_OBJ_LR_PT_Y`: Define o ponto y inferior direito de um objeto a ser definido na imagem (`PtxImage`). `PTX_CLASSIFIER_OBJ_TYPE_CLASS`: Define o tipo de classe de um objeto a ser definido na imagem (`PtxImage`).

enum PtxClassifierResultType	
Descrição	Chaves para requisitar do retorno do *Classifier* (tipo `PtxDict`) os valores das propriedades dos veículos.
Membros	`PTX_CLASSIFIER_GET_INT_RESULTS_SIZE`: Chave para requisitar o valor `int` da quantidade de resultados retornados. `PTX_CLASSIFIER_GET_PTXDICT_AT_RESULT`: Chave para requisitar o valor `PtxDict` de uma determinada posição contendo um único resultado. `PTX_CLASSIFIER_GET_INT_VEHICLE_TYPE_CLASS`: Chave para requisitar o valor `int` que representa o tipo de veículo encontrado. `PTX_CLASSIFIER_GET_FLOAT_VEHICLE_TYPE_PROB`: Chave para requisitar o valor `float` que representa a confiabilidade em forma de probabilidade para o tipo de veículo. `PTX_CLASSIFIER_GET_INT_VEHICLE_X_POINTS_SIZE`: Chave para requisitar o valor `int` que representa quantos valores x são utilizados para descrever o contorno do objeto. `PTX_CLASSIFIER_GET_INT_AT_VEHICLE_X_POINT`: Chave para requisitar o valor `int` que corresponde a um único valor x pertencente ao contorno do objeto. `PTX_CLASSIFIER_GET_INT_VEHICLE_Y_POINTS_SIZE`: Chave para requisitar o valor `int` que representa quantos valores y são utilizados para descrever o contorno do objeto. `PTX_CLASSIFIER_GET_INT_AT_VEHICLE_Y_POINT`: Chave para requisitar o valor `int` que corresponde a um único valor y pertencente ao contorno do objeto. `PTX_CLASSIFIER_GET_INT_PROCESSING_TIME`: Chave para requisitar o valor `int` que corresponde ao tempo de processamento do Classifier. `PTX_CLASSIFIER_GET_PTXSTRING_VEHICLE_COLOR_NAME`: Chave para requisitar o valor `string` que corresponde ao nome da cor do veículo. `PTX_CLASSIFIER_GET_FLOAT_VEHICLE_COLOR_PROB`: Chave para requisitar o valor `float` que corresponde a probabilidade da cor do veículo. `PTX_CLASSIFIER_GET_PTXSTRING_VEHICLE_BRAND_NAME`: Chave para requisitar o valor `string` que corresponde ao nome da marca do veículo.

	`PTX_CLASSIFIER_GET_FLOAT_VEHICLE_BRAND_PROB`: Chave para requisitar o valor `float` que corresponde a probabilidade da marca do veículo. `PTX_CLASSIFIER_GET_PTXSTRING_VEHICLE_MODEL_NAME`: Chave para requisitar o valor `string` que corresponde ao modelo do veículo. `PTX_CLASSIFIER_GET_FLOAT_VEHICLE_MODEL_PROB`: Chave para requisitar o valor `float` que corresponde a probabilidade do modelo do veículo. `PTX_CLASSIFIER_GET_PTXSTRING_OBJECT_TYPE_CLASS`: Chave para requisitar o valor `string` que corresponde ao nome da classe do objeto. `PTX_CLASSIFIER_GET_FLOAT_OBJECT_TYPE_PROB`: Chave para requisitar o valor `float` que corresponde a probabilidade da classe do objeto. `PTX_CLASSIFIER_GET_INT_AT_OBJECT_X_POINT`: Chave para requisitar o valor `int` que corresponde a um único valor x pertencente ao contorno do objeto. `PTX_CLASSIFIER_GET_INT_OBJECT_X_POINTS_SIZE`: Chave para requisitar o valor `int` que representa quantos valores x são utilizados para descrever o contorno do objeto. `PTX_CLASSIFIER_GET_INT_AT_OBJECT_Y_POINT`: Chave para requisitar o valor `int` que corresponde a um único valor y pertencente ao contorno do objeto. `PTX_CLASSIFIER_GET_INT_OBJECT_Y_POINTS_SIZE`: Chave para requisitar o valor `int` que representa quantos valores y são utilizados para descrever o contorno do objeto. `PTX_CLASSIFIER_GET_INT_OBJECT_ATTRIBUTES_SIZE`: Chave para requisitar o valor `int` que corresponde a quantidade de atributos do objeto. `PTX_CLASSIFIER_GET_PTXDICT_AT_OBJECT_ATTRIBUTE`: Chave para requisitar o valor `PtxDict` de uma determinada posição contendo um único atributo.

enum PtxClassifierVehicleType	
Descrição	Campos representando os possíveis retornos da chave `PTX_CLASSIFIER_GET_INT_VEHICLE_TYPE_CLASS,` ou seja, os possíveis tipos de veículos retornados pela biblioteca.
Membros	***`PTX_CLASSIFIER_VEHICLE_TYPE_UNKNOWN:`*** Tipo de veículo que não conseguiu ser categorizado ***`PTX_CLASSIFIER_VEHICLE_TYPE_CAR:`*** Veículo do tipo carro ***`PTX_CLASSIFIER_VEHICLE_TYPE_MOTORCYCLE:`*** Veículo do tipo moto ***`PTX_CLASSIFIER_VEHICLE_TYPE_TRUCK:`*** Veículo do tipo caminhão ***`PTX_CLASSIFIER_VEHICLE_TYPE_BUS:`*** Veículo do tipo ônibus ***`PTX_CLASSIFIER_VEHICLE_TYPE_PICKUP:`*** Veículo do tipo pickup ***`PTX_CLASSIFIER_VEHICLE_TYPE_SUV:`*** Veículo do tipo suv ***`PTX_CLASSIFIER_VEHICLE_TYPE_VAN:`*** Veículo do tipo van ***`PTX_CLASSIFIER_VEHICLE_TYPE_TOW:`*** Veículo do tipo reboque

struct PtxClassifierHandle	
Descrição	Estrutura que contém as configurações internas e estados da biblioteca.

Métodos

ptx_classifier_init	
Protótipo da Função	`int ptx_classifier_init();`
Descrição	Função utilizada para inicializar a biblioteca após sua carga.
Parâmetros	Nenhum
Retorno	Um inteiro com código de retorno de função.

ptx_classifier_destroy	
Protótipo da Função	`int ptx_classifier_destroy();`

Descrição	Função utilizada para descarregar a biblioteca.
Parâmetros	Nenhum
Retorno	Um inteiro com código de retorno de função.

ptx_classifier_get_version	
Protótipo da Função	`int ptx_classifier_get_version(int* major, int* minor, int* release)`
Descrição	Função utilizada para consultar a versão da biblioteca. O *Classifier* segue um sistema de versionamento semântico major.minor.release. O número major é incrementado quando a versão tem quebra de API em relação à versão anterior. Caso não tenha quebra de API e haja inclusão de nova funcionalidade, o número minor é incrementado. Caso não haja quebra de API nem inclusão de nova funcionalidade, o número release é incrementado (é o caso de correções de bugs e alterações pequenas).
Parâmetros	`int *major, int *minor, int *release`: ponteiros para variáveis inteiras onde serão escritos os números que compõem a versão.
Retorno	Um inteiro com código de retorno de função.

ptx_classifier_get_SHA1	
Protótipo da Função	`const char* ptx_classifier_get_SHA1();`
Descrição	Função utilizada para verificar o hash SHA1 do build da biblioteca. Essa string é utilizada para Pumatronix para rastreamento do build.
Parâmetros	Nenhum
Retorno	Retorna um ponteiro para o início de uma string terminada em `\0` contendo o hash SHA1 do build

ptx_classifier_create_handle	
Protótipo da Função	`PtxClassifierHandle* ptx_classifier_create_handle();`
Descrição	Função utilizada para alocar a memória para a configuração da biblioteca. No caso de uso multithread, cada thread deverá chamar `ptx_classifier_create_handle` e usar seu próprio `PtxClassifierHandle`. O mesmo `PtxClassifierHandle` pode ser utilizado quantas vezes for necessário, desde que de apenas uma thread por vez. As configurações da biblioteca, feitas através da função *ptx_classifier_set_config*, são armazenadas no `PtxClassifierHandle`. Portanto, cada `PtxClassifierHandle` pode ter uma configuração diferente. Isso pode ser útil, por exemplo, quando deseja-se utilizar a bilbioteca *Classifier* para processar imagens oriundas de diversas câmeras, e deseja-se aplicar configurações diferentes para imagens de câmeras diferentes.
Parâmetros	Nenhum
Retorno	Retorna um ponteiro para um struct tipo *PtxClassifierHandle* que será utilizado nas chamadas das funções subsequentes.

ptx_classifier_free_handle	
Protótipo da Função	`int ptx_classifier_free_handle(PtxClassifierHandle* handle);`
Descrição	Função utilizada para liberar a memória alocada para o objeto do tipo *PtxClassifierHandle*.
Parâmetros	`PtxClassifierHandle* handle`: Ponteiro para o handle do tipo *PtxClassifierHandle*.
Retorno	Um inteiro com código de retorno de função.

ptx_classifier_set_config	
Protótipo da Função	`int ptx_classifier_set_config(PtxClassifierHandle* handle, PtxDict* config);`
Descrição	Função utilizada para aplicar a configuração feita através de um campo `PtxDict`.
Parâmetros	`PtxClassifierHandle* handle`: Ponteiro para o handle do tipo *PtxClassifierHandle* `PtxDict* config`: Ponteiro para objeto do tipo *PtxDict*
Retorno	Um inteiro com código de retorno de função.

ptx_classifier_classify	
Protótipo da Função	`int ptx_classifier_classify(PtxClassifierHandle* handle, PtxImage* img, PtxDict* results);`
Descrição	Função utilizada para processar e extrair as informações da imagem e retorná-las através do parâmetro `results`. **Esta função pode ser bastante demorada, levando centenas de milissegundos para terminar ou até mais, dependendo da CPU utilizada.**
Parâmetros	`PtxClassifierHandle* handle`: Ponteiro para o handle do tipo *PtxClassifierHandle* `PtxImage* img`: Ponteiro para imagem do tipo *PtxImage* `PtxDict* results`: Ponteiro para resultados do tipo *PtxDict*
Retorno	Um inteiro com código de retorno de função.

ptx_classifier_get_license_info	
Protótipo da Função	`int ptx_classifier_get_license_info(PtxProductLicenseInfo* license);`
Descrição	Função utilizada para requisitar as informações de licença em relação ao produto *Classifier*.
Parâmetros	`PtxProductLicenseInfo* license:` Ponteiro para o objeto do tipo *PtxProductLicenseInfo*
Retorno	Um inteiro com código de retorno de função.

API ptx_common C/C++

A biblioteca *ptx_common* implementa em C estruturas de dados, códigos de erro e outras de'nições comumente utilizadas por produtos da Pumatronix. Sua API está definida no arquivo `ptx_common.h`, que por si inclui outros headers.

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	
Descrição	Define os códigos de erro da biblioteca ptx_common.
Membros	Os vários membros desta enumeração são retornados por funções em diversos tipos de situações: sucesso, imagem inválida, parâmetro inválido, erros de licença, erros de rede etc.

enum PtxProductLicenseInfo	
Descrição	Struct utilizada para armazenar as informações sobre a licença utilizada pela biblioteca
Membros	• `uint64_t serial`: serial number da licença • `char customer[64]`: nome do cliente que adquiriu a licença • `int maxThreads`: número máximo de threads de processamento habilitadas • `int maxConnections`: número máximo de conexões paralelas habilitadas • `int state`: estado da licença - ver *PtxCommonReturnCode* • `int ttl`: time-to-live em horas para licenças do tipo RTC. Este campo possui o valor -1 caso a licença não seja expirável

Métodos

ptx_common_get_SHA1	
Protótipo da Função	`const char* PTXAPI ptx_common_get_SHA1();`
Descrição	Função utilizada para verificar o hash SHA1 do build da biblioteca ptx_common. Essa string é utilizada para Pumatronix para rastreamento do build.
Parâmetros	Nenhum
Retorno	Retorna um ponteiro para o início de uma string terminada em `\0` contendo o hash SHA1 do build.

ptx_dict.h

Este arquivo define uma estrutura de dados de dicionário, usada para armazenar relações do tipo chave-valor. A chave é sempre do tipo int. Cada produto da Pumatronix que utiliza a biblioteca ptx_common enumera em seu header quais chaves 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	
Descrição	Struct opaca utilizada para armazenar o dicionário.
Membros	Nenhum, por ser uma estrutura opaca.

Métodos

ptxdict_create	
Protótipo da Função	`PtxDict* PTXAPI ptxdict_create();`
Descrição	Função utilizada para alocar memória para um dicionário.
Parâmetros	Nenhum
Retorno	Retorna um ponteiro para um struct tipo [PtxDict](#struct-ptxdict] que será utilizado nas chamadas das funções subsequentes.

ptxdict_free	
Protótipo da Função	`int ptxdict_free(PtxDict* ptx_dict);`
Descrição	Função utilizada para liberar a memória alocada para o objeto do tipo *PtxDict*.
Parâmetros	`PtxDict* dict`: Ponteiro para objeto do tipo *PtxDict* cuja memória será liberada.
Retorno	Um inteiro com código de retorno de função.

ptxdict_set_int	
Protótipo da Função	`int ptxdict_set_int(PtxDict* ptx_dict, int key, int value);`
Descrição	Função utilizada para armazenar um par chave-valor.
Parâmetros	`PtxDict* dict`: Ponteiro para objeto do tipo *PtxDict* onde o par chave-valor será armazenado. `int key`: chave do tipo int. `int value`: valor do tipo int que será armazenado como associado à `key`.
Retorno	Um inteiro com código de retorno de função.

ptxdict_set_float	
Protótipo da Função	`int ptxdict_set_float(PtxDict* ptx_dict, int key, float value);`
Descrição	Função utilizada para armazenar um par chave-valor.
Parâmetros	`PtxDict* dict`: Ponteiro para objeto do tipo *PtxDict* onde o par chave-valor será armazenado. `int key`: chave do tipo int.

	`float value`: valor do tipo int que será armazenado como associado à `key`
Retorno	Um inteiro com código de retorno de função.

ptxdict_get_int	
Protótipo da Função	`int ptxdict_get_int(PtxDict* ptx_dict, int key, int* value);`
Descrição	Função utilizada para requisitar um valor associado a uma chave.
Parâmetros	`PtxDict* dict`: Ponteiro para objeto do tipo *PtxDict*. `int key`: chave do tipo int cujo valor deseja-se requisitar. `int* value`: valor do tipo int que será armazenado como associado à `key`, caso exista.
Retorno	Um inteiro com código de retorno de função.

ptxdict_get_float	
Protótipo da Função	`int ptxdict_get_float(PtxDict* ptx_dict, int key, float* value);`
Descrição	Função utilizada para requisitar um valor associado a uma chave.
Parâmetros	`PtxDict* dict`: Ponteiro para objeto do tipo *PtxDict*. `int key`: chave do tipo int cujo valor deseja-se requisitar. `float value`: ponteiro para int onde será escrito o valor associado à `key`, caso exista.
Retorno	Um inteiro com código de retorno de função.

ptxdict_get_ptxstring	
Protótipo da Função	`int ptxdict_get_ptxstring(PtxDict* ptx_dict, int key, PtxString* value);`
Descrição	Função utilizada para requisitar uma string associada a uma chave.
Parâmetros	`PtxDict* dict`: Ponteiro para objeto do tipo *PtxDict*. `int key`: chave do tipo int cujo valor deseja-se requisitar. `PtxString* value`: ponteiro para objeto do tipo *PtxString* onde será escrito o valor associado à `key`, caso exista.
Retorno	Um inteiro com código de retorno de função.

ptxdict_get_int_at	
Protótipo da Função	`int ptxdict_get_int_at(PtxDict* ptx_dict, int key, int index, int* value);`
Descrição	Função utilizada para requisitar o valor em um determinado índice de um vetor associado a uma chave.
Parâmetros	`PtxDict* dict`: Ponteiro para objeto do tipo *PtxDict*. `int key`: chave do tipo int cujo valor deseja-se requisitar. `int index`: índice desejado do vetor associado à `key.` `int* value`: ponteiro para float onde será escrito o valor, caso exista.
Retorno	Um inteiro com código de retorno de função.

ptxdict_get_ptxdict_at	
Protótipo da Função	`int ptxdict_get_ptxdict_at(PtxDict* ptx_dict, int key, int index, PtxDict* value);`

Descrição	Função utilizada para requisitar o valor em um determinado índice de um vetor associado a uma chave.
Parâmetros	`PtxDict* dict`: Ponteiro para objeto do tipo *PtxDict*. `int key`: chave do tipo int cujo valor deseja-se requisitar. `int index`: índice desejado do vetor associado à `key`. `PtxDict* value`: ponteiro para *PtxDict* onde será escrito o valor, caso exista.
Retorno	Um inteiro com código de retorno de função.

ptx_image.h

Este arquivo define uma estrutura para carregar imagens. Suporta tanto imagens estruturadas (jpg e bmp) quanto imagens 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	
Descrição	Struct opaca utilizada para carregar imagens.
Membros	Nenhum, por ser uma estrutura opaca.

enum PtxImagePixelFmt	
Descrição	Enumeração dos tipos de imagem RAW que podem ser carregados.
Membros	Nenhum, por ser uma estrutura opaca.

Métodos

ptximage_create	
Protótipo da Função	`PtxImage* ptximage_create();`
Descrição	Função utilizada para alocar memória para uma estrutura de imagem.
Parâmetros	Nenhum
Retorno	Retorna um ponteiro para um struct tipo [PtxImage](#struct-ptximage] que será utilizado nas chamadas das funções subsequentes.

ptximage_free	
Protótipo da Função	`int ptximage_free(PtxImage* img);`
Descrição	Função utilizada para liberar a memória alocada para um objeto do tipo *PtxImage*.
Parâmetros	`PtxImage* img`: Ponteiro para objeto do tipo *PtxImage* cuja memória será liberada.
Retorno	Um inteiro com código de retorno de função.

ptximage_load_from_file	
Protótipo da Função	`int ptximage_load_from_file(PtxImage* img, const char* filename);`
Descrição	Função utilizada para carregar uma imagem estruturada (jpg ou bmp) a partir de um arquivo.
Parâmetros	`PtxImage* img`: Ponteiro para objeto do tipo *PtxImage* onde será armazenada a imagem em memória. `const char* filename`: Arquivo que contém uma imagem estruturada a ser carregada.
Retorno	Um inteiro com código de retorno de função.

ptximage_load_from_memory	
Protótipo da Função	`int ptximage_load_from_memory(PtxImage* img, const uint8_t* buffer, int bufferSize);`
Descrição	Função utilizada para carregar uma imagem estruturada (jpg ou bmp) a partir de um buffer em memória.
Parâmetros	`PtxImage* img`: Ponteiro para objeto do tipo *PtxImage* onde será armazenada a imagem em memória. `const uint8* buffer`: Buffer contendo a imagem estruturada a ser carregada. `int bufferSize`: Tamanho de `buffer` em bytes.
Retorno	Um inteiro com código de retorno de função.

ptximage_load_from_raw_format	
Protótipo da Função	`int ptximage_load_from_raw_format(PtxImage* img, const uint8_t* buffer, int width, int height, int stride, PtxImagePixelFmt fmt)`
Descrição	Função utilizada para carregar uma imagem RAW a partir de um buffer em memória. Os tipos de imagens suportados estão definidos por *PtxImagePixelFmt*.

Parâmetros	`PtxImage* img`: Ponteiro para objeto do tipo *PtxImage* onde será armazenada a imagem em memória. `const uint8* buffer`: Buffer contendo a imagem RAW a ser carregada. `int width`: Largura da imagem em pixels. `int height`: Altura da imagem em pixels. `int stride`: Número de bytes entre uma linha e a linha seguinte da imagem. `PtxImagePixelFmt fmt`: Tipo de imagem RAW, conforme *PtxImagePixelFmt*.
Retorno	Um inteiro com código de retorno de função.

ptximage_append_extra	
Protótipo da Função	`int ptximage_append_extra(PtxImage* img, PtxDict* extra);`
Descrição	Função para adicionar um extra *PtxDict* na estrutura de imagem.
Parâmetros	`PtxImage* img`: Ponteiro para objeto do tipo *PtxImage*. `PtxDict* extra`: Ponteiro para o extra *PtxDict* a ser adicionado na imagem.
Retorno	Um inteiro com código de retorno de função.

ptximage_clear_extras	
Protótipo da Função	`int ptximage_clear_extras(PtxImage* img);`
Descrição	Função utilizada para limpar os extras adicionados.
Parâmetros	`PtxImage* img`: Ponteiro para objeto do tipo *PtxImage*.
Retorno	Um inteiro com código de retorno de função.

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	
Descrição	Struct opaca utilizada para manter textos.
Membros	Nenhum, por ser uma estrutura opaca.

Métodos

ptxstring_create	
Protótipo da Função	`PtxString* ptxstring_create();`
Descrição	Função utilizada para alocar memória para uma estrutura de texto.
Parâmetros	Nenhum
Retorno	Retorna um ponteiro para um struct tipo *PtxString* que será utilizado nas chamadas das funções subsequentes.

ptxstring_free	
Protótipo da Função	`int ptxstring_free(PtxString* ptx_string);`
Descrição	Função utilizada para liberar a memória alocada para um objeto do tipo *PtxString*.
Parâmetros	`PtxString* ptx_string`: Ponteiro para objeto do tipo *PtxString* cuja memória será liberada.
Retorno	Um inteiro com código de retorno de função.

ptxstring_set	
Protótipo da Função	`int ptxstring_set(PtxString* ptx_string, const char* str);`
Descrição	Função utilizada para copiar os catacteres de um const char* para dentro da estrutura *PtxString*.
Parâmetros	`PtxString* ptx_string`: Ponteiro para objeto do tipo *PtxString*. `const char* str`: Ponteiro para um texto em formato const char*.
Retorno	Um inteiro com código de retorno de função.

ptxstring_get	
Protótipo da Função	`const char* ptxstring_get(PtxString* ptx_string);`
Descrição	Função utilizada para recuperar os catacteres de um const char* de dentro da estrutura *PtxString*.
Parâmetros	`PtxString* ptx_string`: Ponteiro para objeto do tipo *PtxString*.
Retorno	Ponteiro para um texto em formato const char*.


PUMATRONIX
www.pumatronix.com