



Software Development Kit

Dynamic Thresholding

Copyright 2020

Recogniform Technologies SpA

HOW TO CONTACT US

Recogniform Technologies SpA

Contrada Concistocchi

87036 Rende (CS), Italy

Phone : +39 0984 404174

Fax : +39 0984 830299

Internet : www.recogniform.com

E-Mail : info@recogniform.com

Table of contents

Introduction	5
Copyright	5
License	5
Overview	5
Usage	8
Visual C++	8
C#	8
Visual Basic	8
Visual Basic .NET	8
Delphi	8
Java	8
API Reference	10
LoadDTLibrary (Visual C++ only)	10
FreeDTLibrary (Visual C++ only)	10
DT_Init	10
DT_Done	12
DT_Threshold	13
DT_AutoThreshold	14
DT_DynamicThresholdMinMax	15
DT_DynamicThresholdAverage	16
DT_AdvancedThresholdDeviation	18
DT_AdvancedThresholdVariance	19
DT_EdgeThreshold	21
DT_AdaptiveThresholdMinMax	22
DT_AdaptiveThresholdAverage	24
DT_BackTrackThresholdMinMax	25
DT_BackTrackThresholdAverage	27
DT_AdaptiveThresholdBackTrack	28

Introduction

I

1 Introduction

1.1 Copyright

The software and the documentation are property of:

Recogniform Technologies SpA
Contrada Concistocchi
87036 Rende (CS)
Italy
www.recogniform.com
info@recogniform.com

1.2 License

It is illegal to copy or reproduce this manual, or any part thereof, in any shape or form.

The information contained in this manual is subject to change without notice and does not present a commitment on the part of Recogniform Technologies SpA.

Recogniform Technologies SpA shall not be held liable for technical or editorial errors and/or omissions made here, nor for incidental or consequential damages resulting from the furnishing, performance, or use of the software and documentation.

Recogniform Technologies SpA reserves the right to make changes to the software and documentation without notice.

Product names mentioned here are used for identification purposes only and may be tradenames and/or registered trademarks of their respective companies.

YOU CANNOT DISTRIBUTE SOFTWARE INCLUDING THIS SDK LIBRARY UNLESS YOU HAVE A WRITTEN AGREEMENT (ROYALTIES FREE OR ROYALTIES BASED) WITH RECOGNIFORM TECHNOLOGIES SPA

1.3 Overview

The Recogniform Dynamic Thresholding Library 8.2 allows to binarize grayscale images using state of art dynamic thresholding technology.

The supported thresholding algorithms are:

- Fixed thresholding (global with user supplied threshold level)
- Auto thresholding Otsu (global with auto-calculated threshold level)
- Auto thresholding Lloyd (global with auto-calculated threshold level)
- Auto thresholding Ridel&Cravler (global with auto-calculated threshold level)
- Auto thresholding sigma (global with auto-calculated threshold level)
- Auto thresholding uniform error (global with auto-calculated threshold level)
- Dynamic thresholding minmax (local auto-calculated dynamic threshold level)
- Dynamic thresholding average (local auto-calculated dynamic threshold level)
- Advanced dynamic thresholding (local auto-calculated dynamic threshold level)
- Smart thresholding (local auto-calculated dynamic threshold level)
- Edge thresholding (local auto-calculated dynamic threshold level)
- Adpative thresholding (local auto-calculated dynamic threshold level)
- Background tracking thresholding (local auto-calculated dynamic threshold level)

Usage



2 Usage

2.1 Visual C++

You have to include the RECODTAPI.C in your program (see sample application). Before to execute your application make sure the *RECODT.DLL* is available in your same .exe directory or in windows\system directory.

2.2 C#

You have to include the module RECODTAPI.CS in your program Before to execute your application make sure the *RECODT.DLL* is available in your same .exe directory or in windows\system directory.

2.3 Visual Basic

You have to include the module RECODTAPI.BAS in your program Before to execute your application make sure the *RECODT.DLL* is available in your same .exe directory or in windows\system directory.

2.4 Visual Basic .NET

You have to include the module RECODTAPI.VB in your program Before to execute your application make sure the *RECODT.DLL* is available in your same .exe directory or in windows\system directory.

2.5 Delphi

You have to include the unit RECODTAPI.PAS in your program. Before to execute your application make sure the *RECODT.DLL* is available in your same .exe directory or in windows\system directory.

2.6 Java

You have to use JVM and you have to include the RECODTAPI.JAVA in your program. Before to execute your application make sure the *RECODT.DLL* is available in your same .jar directory.

API Reference



3 API Reference

3.1 LoadDTLibrary (Visual C++ only)

C/C++ Declaration

```
long LoadDTLibrary();
```

Description

Load the Dynamic Thresholding DLL library: you have to use this function one time before to use other API functions from Visual C++. This function is not required using the API from the other languages.

3.2 FreeDTLibrary (Visual C++ only)

C/C++ Declaration

```
void FreeDTLibrary();
```

Description

Unload the Dynamic Thresholding DLL library: you have to use this function one time before to exit from your Visual C++ application. This function is not required using the API from the other languages.

3.3 DT_Init

C/C++ Declaration

```
__stdcall long DT_Init(char* Name, char*  
LicenseKey);
```

C# Declaration

```
int DT_Init(String Name, String LicenseKey);
```

Visual Basic Declaration

```
Function DT_Init(ByVal Name As String, ByVal  
LicenseKey As String) As Integer
```

Visual Basic .NET Declaration

```
Function DT_Init(ByVal Name As String, ByVal  
LicenseKey As String) As Integer
```

Delphi Declaration

```
function DT_Init(Name:PAnsiChar;  
LicenseKey:PAnsiChar) :Integer; stdcall;
```

Java Declaration

```
int DT_Init(String Name, String LicenseKey);
```

Description

This is the first function to call: initialize the library and returns a session handle to use in next calls. When you buy the library you receive an "user" and a "password" string necessary to initialize the library in normal mode: without this value or with wrong values the library is initialized in evaluation mode. The evaluation mode works exactly as normal mode but some time, when you call the thresholding function, a warning dialog box remembering the evaluation state is displayed: you can close it and continue to work with no problems.

Parameters

User (in) - the user name string

Passwors (in) - the password string

Return values

SessionHandle (out) - the session handle to use in next library calls.
This value is 0 if the library is not initialized.

Note

The "status" string, shown in the windows dialog, is built by 1 and 0 meaning OK or KO for this checking in the order from left to right:

- SDK initialized with valid user/key
- License file integrity
- License file matching sdk version
- Computer verification done or not required
- Dongle verification done or not required
- Date expiration done or not required

- SDK option used unlocked

3.4 DT_Done

C/C++ Declaration

```
__stdcall void DT_Done(long SessionHandle);
```

C# Declaration

```
void DT_Done(int SessionHandle);
```

Visual Basic Declaration

```
Sub DT_Done(ByVal SessionHandle As Integer)
```

Visual Basic .NET Declaration

```
Sub DT_Done(ByVal SessionHandle As Integer)
```

Delphi Declaration

```
procedure DT_Done(SessionHandle:Integer); stdcall;
```

Java Declaration

```
void DT_Done(int SessionHandle);
```

Description

This is the last function to call when you do not need more services from the library: deinitialize the library and free all used resources.

Parameters

SessionHandle (in) - the session handle to free

Return values

None

3.5 DT_Threshold

C/C++ Declaration

```
__stdcall void DT_Threshold(long SessionHandle,
                           long DIBHandle, long Level);
```

C# Declaration

```
void DT_Threshold(int SessionHandle, int
                  DIBHandle, int Level);
```

Visual Basic Declaration

```
Sub DT_Threshold (ByVal SessionHandle As Long,
                  ByVal DIBHandle As Long, ByVal Level As Long)
```

Visual Basic .NET Declaration

```
Sub DT_Threshold(ByVal SessionHandle As Integer,
                  ByVal DIBHandle As Integer, ByVal Level As
                  Integer)
```

Delphi Declaration

```
procedure DT_Threshold(SessionHandle:Integer;
                       DIBHandle:Integer; Level:Integer); stdcall;
```

Java Declaration

```
void DT_Threshold(int SessionHandle, int
                  DIBHandle, int Level);
```

Description

Thresholds a grayscale image to create a monochrome image. All grayscale pixel values under the threshold level are converted to black while all the pixels above are converted to white.

Parameters

SessionHandle (in) - the session handle to use
intDIBHandle: the image handle.
intLevel: threshold value in the range 0 - 255.

Return values

None

3.6 DT_AutoThreshold

C/C++ Declaration

```
__stdcall long DT_AutoThreshold(long SessionHandle, long DIBHandle, long Algo);
```

C# Declaration

```
int DT_AutoThreshold(int SessionHandle, int DIBHandle, int Algo);
```

Visual Basic Declaration

```
Function DT_AutoThreshold(ByVal SessionHandle As Long, ByVal DIBHandle As Long, ByVal Algo As Long)
```

Visual Basic .NET Declaration

```
Function DT_AutoThreshold(ByVal SessionHandle As Integer, ByVal DIBHandle As Integer, ByVal Algo As Integer)
```

Delphi Declaration

```
function DT_AutoThreshold(SessionHandle:Integer; DIBHandle:Integer; Algo:Integer):Integer; stdcall;
```

Java Declaration

```
int DT_AutoThreshold(int SessionHandle, int DIBHandle, int Algo);
```

Description

Thresholds a grayscale image by automatically calculating the best threshold level. The threshold level is used globally and can be calculated using several algorithms.

Parameters

SessionHandle (in) - the session handle to use
intDIBHandle: the image handle.

Algo: the thresholding algorithm. Allowed values are:

```

0 = Otsu
1 = Lloyd
2 = Ridler-Calvard
3 = Uniform Error
4 = Standard Deviation
5 = Clustering
6 = Median

```

Return values

The calculated threshold value from 0 - 255 used for binarization.

3.7 DT_DynamicThresholdMinMax

C/C++ Declaration

```
__stdcall void DT_DynamicThresholdMinMax(long SessionHandle,
```

C# Declaration

```
void DT_DynamicThresholdMinMax(int SessionHandle,
int DIBHandle, int Width, int Height, int
Contrast, int Brightness);
```

Visual Basic Declaration

```
Sub DT_DynamicThresholdMinMax(ByVal SessionHandle
As Long, ByVal DIBHandle As Long, ByVal Width As
Long, ByVal Height As Long, ByVal Contrast As
Long, ByVal Brightness As Long)
```

Visual Basic .NET Declaration

```
Sub DT_DynamicThresholdMinMax(ByVal SessionHandle
As Integer, ByVal DIBHandle As Integer, ByVal
Width As Integer, ByVal Height As Integer, ByVal
Contrast As Integer, ByVal Brightness As Integer)
```

Delphi Declaration

```

procedure
DT_DynamicThresholdMinMax(SessionHandle:Integer;
DIBHandle:Integer; Width:Integer; Height:Integer;
Contrast:Integer; Brightness:Integer); stdcall;
```

Java Declaration

```
void DT_DynamicThresholdMinMax(int SessionHandle,  
int DIBHandle, int Width, int Height, int Contrast  
, int Brightness);
```

Description

Converts a grayscale image applying dynamic thresholding based on min/max algo. For each pixel its evaluated its contrast respect other pixels in the local window and if its above the local contrast value, it's binarized using the local threshold computed automatically using minimum and maximum gray pixels in the local window. The process can be customized selecting local windows size, local contrast and global brightness.

Parameters

SessionHandle (in) - the session handle to use

DIBHandle: the image handle.

Width: value specifying window width.

Height: value specifying window height.

Contrast: reference value for evaluating local contrast, between 0 and 255. Use -1 to allow auto-calculation of the best value for the image.

Brightness: global brightness of resulting image, between 0 and 255. A low value produces a darker images, an high value produces a lighter image. Use -1 to allow auto-calculation of the best value for the image.

Return values

None

3.8 DT_DynamicThresholdAverage

C/C++ Declaration

```
__stdcall void DT_DynamicThresholdAverage(long  
SessionHandle, long DIBHandle, long Width, long  
Height, long Contrast, long Brightness);
```

C# Declaration

```
void DT_DynamicThresholdAverage(int SessionHandle,
```

```
int ImageHandle, int Width, int Height, int
Contrast, int Brightness);
```

Visual Basic Declaration

```
Sub DT_DynamicThresholdAverage(ByVal SessionHandle
As Long, ByVal DIBHandle As Long, ByVal Width As
Long, ByVal Height As Long, ByVal Contrast As
Long, ByVal Brightness As Long)
```

Visual Basic .NET Declaration

```
Sub DT_DynamicThresholdAverage(ByVal SessionHandle
As Integer, ByVal DIBHandle As Integer, ByVal
Width As Integer, ByVal Height As Integer, ByVal
Contrast As Integer, ByVal Brightness As Integer)
```

Delphi Declaration

```
procedure
DT_DynamicThresholdAverage(SessionHandle:Integer;
DIBHandle:Integer; Width:Integer; Height:Integer;
Contrast:Integer; Brightness:Integer); stdcall;
```

Java Declaration

```
void DT_DynamicThresholdAverage(int SessionHandle,
int DIBHandle, int Width, int Height, int
Contrast, int Brightness);
```

Description

Converts a grayscale image applying dynamic thresholding based on average algo. For each pixel its evaluated its contrast respect other pixels in the local window and if its above the local contrast value, it's binarized using the local threshold computed automatically averaging the pixels in the local window. The process can be customized selecting local windows size, local contrast and global brightness.

Parameters

SessionHandle (in) - the session handle to use
DIBHandle: the image handle.

Width: value specifying window width.

Height: value specifying window height.

Contrast: reference value for evaluating local contrast, between 0 and 255. Use -1 to allow auto-calculation of the best value for the image.

Brightness: global brightness of resulting image, between 0 and 255. Low values produce a darker images, high values produce a lighter image. Use -1 to allow auto-calculation of the best value for the image.

Return values

None

3.9 DT_AdvancedThresholdDeviation

C/C++ Declaration

```
__stdcall void DT_AdvancedThresholdDeviation(long SessionHandle, long DIBHandle, long Width, long Height, long Contrast, long Brightness);
```

C# Declaration

```
void DT_AdvancedThresholdDeviation(int SessionHandle, int DIBHandle, int Width, int Height, int Contrast, int Brightness);
```

Visual Basic Declaration

```
Sub DT_AdvancedThresholdDeviation (ByVal SessionHandle As Long, ByVal DIBHandle As Long, ByVal Width As Long, ByVal Height As Long, ByVal Contrast As Long, ByVal Brightness As Long)
```

Visual Basic .NET Declaration

```
Sub DT_AdvancedThresholdDeviation(ByVal SessionHandle As Integer, ByVal DIBHandle As Integer, ByVal Width As Integer, ByVal Height As Integer, ByVal Contrast As Integer, ByVal Brightness As Integer)
```

Delphi Declaration

```
procedure DT_AdvancedThresholdDeviation(SessionHandle: Integer; DIBHandle: Integer; Width: Integer;
```

```
Height:Integer; Contrast:Integer;
Brightness:Integer); stdcall;
```

Java Declaration

```
void DT_AdvancedThresholdDeviation(int
SessionHandle, int DIBHandle, int Width, int
Height, int Contrast, int Brightness);
```

Description

Applies advanced thresholding to the image using standard deviation algo. For each pixel of the image its calculated its standard deviation analyzing surrounding pixels in the local window, and thresholded according.

Parameters

SessionHandle (*in*) - the session handle to use

DIBHandle: the image handle.

Width: value specifying window width.

Height: value specifying window height.

Contrast: reference value for evaluating standard deviation, between 0 and 255. Use -1 to allow auto-calculation of the best value for the image.

Brightness: global brightness of resulting image, between 0 and 255. A low value produces a darker images, an high value produces a lighter image. Use -1 to allow auto-calculation of the best value for the image.

Return values

None

3.10 DT_AdvancedThresholdVariance

C/C++ Declaration

```
__stdcall void DT_AdvancedThresholdVariance(long
SessionHandle, long DIBHandle, long Width, long
Height, long Contrast, long Brightness);
```

C# Declaration

```
void DT_AdvancedThresholdVariance(int
SessionHandle, int DIBHandle, int Width, int
```

```
Height, int Contrast, int Brightness);
```

Visual Basic Declaration

```
Sub DT_AdvancedThresholdVariance (ByVal  
SessionHandle As Long, ByVal DIBHandle As Long,  
ByVal Width As Long, ByVal Height As Long, ByVal  
Contrast As Long, ByVal Brightness As Long)
```

Visual Basic .NET Declaration

```
Sub DT_AdvancedThresholdVariance (ByVal  
SessionHandle As Integer, ByVal DIBHandle As  
Integer, ByVal Width As Integer, ByVal Height As  
Integer, ByVal Contrast As Integer, ByVal  
Brightness As Integer)
```

Delphi Declaration

```
procedure  
DT_AdvancedThresholdVariance(SessionHandle: Integer  
; DIBHandle: Integer; Width: Integer;  
Height: Integer; Contrast: Integer;  
Brightness: Integer); stdcall;
```

Java Declaration

```
void DT_AdvancedThresholdVariance(int  
SessionHandle, int DIBHandle, int Width, int  
Height, int Contrast, int Brightness);
```

Description

Applies advanced thresholding to the image using variance algo.
For each pixel of the image its calculated its variance analyzing
surrounding pixels in the local window, and thresholded according.

Parameters

SessionHandle (in) - the session handle to use
DIBHandle: the image handle.

Width: value specifying window width.

Height: value specifying window height.

Contrast: reference value for evaluating variance,between 0 and
255. Use -1 to allow auto-calculation of the best value for the image.

Brightness: global brightness of resulting image, between 0 and

255. A low value produces a darker images, an high value produces a lighter image. Use -1 to allow auto-calculation of the best value for the image.

Return values

None

3.11 DT_EdgeThreshold

C/C++ Declaration

```
__stdcall void DT_EdgeThreshold(long SessionHandle, long DIBHandle, long Width, long Height, long Activity, long MaxSpotSize);
```

C# Declaration

```
void DT_EdgeThreshold(int SessionHandle, int ImageHandle, int Width, int Height, int Activity, int MaxSpotSize);
```

Visual Basic Declaration

```
Sub DT_EdgeThreshold(ByVal SessionHandle As Long, ByVal DIBHandle As Long, ByVal Width As Long, ByVal Height As Long, ByVal Activity As Long, ByVal MaxSpotSize As Long)
```

Visual Basic .NET Declaration

```
Sub DT_EdgeThreshold(ByVal SessionHandle As Integer, ByVal DIBHandle As Integer, ByVal Width As Integer, ByVal Height As Integer, ByVal Activity As Integer, ByVal MaxSpotSize As Integer)
```

Delphi Declaration

```
procedure DT_EdgeThreshold(SessionHandle:Integer; DIBHandle:Integer; Width:Integer; Height:Integer; Activity:Integer; MaxSpotSize:Integer); stdcall;
```

Java Declaration

```
void DT_EdgeThreshold(int SessionHandle, int DIBHandle, int Width, int Height, int Activity,
```

```
int MaxSpotSize);
```

Description

Applies edge thresholding to the image.

Parameters

SessionHandle (in) - the session handle to use

DIBHandle: the image handle.

Width: value specifying window width.

Height: value specifying window height.

Activity: the sensitivity level to find edges, between 0 and 255. Use -1 to allow auto-calculation of the best value for the image.

MaxSpotSize: the radius of smallest biggest foreground element to consider noise.

Return values

None

3.12 DT_AdaptiveThresholdMinMax

C/C++ Declaration

```
_stdcall void DT_AdaptiveThresholdMinMax(long SessionHandle, long DIBHandle, long Width, long Height, long Contrast, long Brightness);
```

C# Declaration

```
void DT_AdaptiveThresholdMinMax(int SessionHandle, int ImageHandle, int Width, int Height, int Contrast, int Brightness);
```

Visual Basic Declaration

```
Sub DT_AdaptiveThresholdMinMax(ByVal SessionHandle As Long, ByVal DIBHandle As Long, ByVal Width As Long, ByVal Height As Long, ByVal Contrast As Long, ByVal Brightness As Long)
```

Visual Basic .NET Declaration

```
Sub DT_AdaptiveThresholdMinMax(ByVal SessionHandle As Integer, ByVal DIBHandle As Integer, ByVal
```

```
Width As Integer, ByVal Height As Integer, ByVal  
Contrast As Integer, ByVal Brightness As Integer)
```

Delphi Declaration

```
procedure  
DT_AdaptiveThresholdMinMax(SessionHandle: Integer;  
DIBHandle: Integer; Width: Integer; Height: Integer;  
Contrast: Integer; Brightness: Integer); stdcall;
```

Java Declaration

```
void DT_AdaptiveThresholdMinMax(int SessionHandle,  
int DIBHandle, int Width, int Height, int  
Contrast, int Brightness);
```

Description

Converts a grayscale image applying dynamic thresholding based on min/max algo. Each pixel in not homogeneous areas its it's binarized using the local threshold computed automatically using minimum and maximum gray pixels in the local window. The process can be customized selecting local windows size, local contrast and global brightness.

Parameters

SessionHandle (in) - the session handle to use

DIBHandle: the image handle.

Width: value specifying window width.

Height: value specifying window height.

Contrast: reference value for evaluating homogeneous area, between 0 and 255. Use -1 to allow auto-calculation of the best value for the image.

Brightness: global brightness of resulting image, between 0 and 255. A low value produces a darker images, an high value produces a lighter image. Use -1 to allow auto-calculation of the best value for the image.

Return values

None

3.13 DT_AdaptiveThresholdAverage

C/C++ Declaration

```
__stdcall void DT_AdaptiveThresholdAverage(long SessionHandle, long DIBHandle, long Width, long Height, long Contrast, long Brightness);
```

C# Declaration

```
void DT_AdaptiveThresholdAverage(int SessionHandle, int DIBHandle, int Width, int Height, int Contrast, int Brightness);
```

Visual Basic Declaration

```
Sub DT_AdaptiveThresholdAverage(ByVal SessionHandle As Long, ByVal DIBHandle As Long, ByVal Width As Long, ByVal Height As Long, ByVal Contrast As Long, ByVal Brightness As Long)
```

Visual Basic .NET Declaration

```
Sub DT_AdaptiveThresholdAverage(ByVal SessionHandle As Integer, ByVal DIBHandle As Integer, ByVal Width As Integer, ByVal Height As Integer, ByVal Contrast As Integer, ByVal Brightness As Integer)
```

Delphi Declaration

```
procedure DT_AdaptiveThresholdAverage(SessionHandle:Integer; DIBHandle:Integer; Width:Integer; Height:Integer; Contrast:Integer; Brightness:Integer); stdcall;
```

Java Declaration

```
void DT_AdaptiveThresholdAverage(int SessionHandle, int DIBHandle, int Width, int Height,int Contrast, int Brightness);
```

Description

Converts a grayscale image applying adaptive thresholding based on average algo. Each pixel in not homogeneous areas its it's binarized using the local threshold computed automatically

averaging the pixels in the local window. The process can be customized selecting local windows size, local contrast and global brightness

Parameters

SessionHandle (in) - the session handle to use

DIBHandle: the image handle.

Width: value specifying window width.

Height: value specifying window height.

Contrast: reference value for evaluating homogeneous area, between 0 and 255. Use -1 to allow auto-calculation of the best value for the image.

Brightness: global brightness of resulting image, between 0 and 255. Low values produces a darker images, high values produce a lighter image. Use -1 to allow auto-calculation of the best value for the image.

Return values

None

3.14 DT_BackTrackThresholdMinMax

C/C++ Declaration

```
__stdcall void DT_BackTrackThresholdMinMax(long SessionHandle, long DIBHandle, long Width, long Height, long MinThreshold, long MaxThreshold, long Contribute);
```

C# Declaration

```
void DT_BackTrackThresholdMinMax(int SessionHandle, int DIBHandle, int Width, int Height, int MinThreshold, int MaxThreshold, int Contribute);
```

Visual Basic Declaration

```
Sub DT_BackTrackThresholdMinMax( ByVal SessionHandle As Long, ByVal DIBHandle As Long, ByVal Width As Long, ByVal Height As Long, ByVal MinThreshold As Long, ByVal MaxThreshold As Long, ByVal Contribute As Long)
```

Visual Basic .NET Declaration

```
Sub DT_BackTrackThresholdMinMax(ByVal  
SessionHandle As Integer, ByVal DIBHandle As  
Integer, ByVal Width As Integer, ByVal Height As  
Integer, ByVal MinThreshold As Integer, ByVal  
MaxThreshold As Integer, ByVal Contribute As  
Integer)
```

Delphi Declaration

```
procedure DT_BackTrackThresholdMinMax  
(SessionHandle:Integer; DIBHandle:Integer;  
Width:Integer; Height:Integer;  
MinThreshold:Integer; MaxThreshold:Integer;  
Contribute:Integer); stdcall;
```

Java Declaration

```
void DT_BackTrackThresholdMinMax(int  
SessionHandle, int DIBHandle, int Width, int  
Height, int MinThreshold, int MaxThreshold, int  
Contribute);
```

Description

Converts a grayscale image by applying background tracking thresholding. Each pixel of image is binarized using a local threshold value computed automatically by analyzing surrounding pixels in the specified window size so that a automatic global threshold value is corrected using a local value obtained using minimum and maximum grayscale values of local window. The process can be customized selecting several parameters.

Parameters

SessionHandle (in) - the session handle to use
DIBHandle: the image handle.

Width: value specifying window width.

Height: value specifying window height.

MinThreshold: value specifying minimal threshold to use, between 0 and 255. Use -1 to allow auto-calculation of the best value for the image.

MaxThreshold: value specifying maximum threshold to use, between 0 and 255. Use -1 to allow auto-calculation of the best value for the image.

Contribute: the contribute percentage, between 0 and 100% of local threshold over global threshold.

Return values

None

3.15 DT_BackTrackThresholdAverage

C/C++ Declaration

```
__stdcall void DT_BackTrackThresholdAverage(long SessionHandle, long DIBHandle, long Width, long Height, long MinThreshold, long MaxThreshold, long Contribute);
```

C# Declaration

```
void DT_BackTrackThresholdAverage(int SessionHandle, int DIBHandle, int Width, int Height, int MinThreshold, int MaxThreshold, int Contribute);
```

Visual Basic Declaration

```
Sub DT_BackTrackThresholdAverage(ByVal SessionHandle As Long, ByVal DIBHandle As Long, ByVal Width As Long, ByVal Height As Long, ByVal MinThreshold As Long, ByVal MaxThreshold As Long, ByVal Contribute As Long)
```

Visual Basic .NET Declaration

```
Sub DT_BackTrackThresholdAverage(ByVal SessionHandle As Integer, ByVal DIBHandle As Integer, ByVal Width As Integer, ByVal Height As Integer, ByVal MinThreshold As Integer, ByVal MaxThreshold As Integer, ByVal Contribute As Integer)
```

Delphi Declaration

```
procedure DT_BackTrackThresholdAverage(SessionHandle:Integer;
; DIBHandle:Integer; Width:Integer;
Height:Integer; MinThreshold:Integer;
```

```
MaxThreshold:Integer; Contribute:Integer);  
stdcall;
```

Java Declaration

```
void DT_BackTrackThresholdAverage(int  
SessionHandle, int DIBHandle, int Width, int  
Height, int MinThreshold, int MaxThreshold, int  
Contribute);
```

Description

Converts a grayscale image by applying background tracking thresholding. Each pixel of image is binarized using a local threshold value computed automatically by analyzing surrounding pixels in the specified window size so that a automatic global threshold value is corrected using a local value obtained averaging grayscale values of local window. The process can be customized selecting several parameters.

Parameters

SessionHandle (in) - the session handle to use

DIBHandle: the image handle.

Width: value specifying window width.

Height: value specifying window height.

MinThreshold: value specifying minimal threshold to use, between 0 and 255. Use -1 to allow auto-calculation of the best value for the image.

MaxThreshold: value specifying maximum threshold to use, between 0 and 255. Use -1 to allow auto-calculation of the best value for the image.

Contribute: the contribute percentage, between 0 and 100% of local threshold over global threshold.

Return values

None

3.16 DT_AdaptiveThresholdBackTrack

C/C++ Declaration

```
__stdcall void DT_AdaptiveThresholdBackTrack(long  
SessionHandle, long DIBHandle, long Width, long  
Height, long Contrast, long Brightness);
```

C# Declaration

```
void DT_AdaptiveThresholdBackTrack(int
SessionHandle, int DIBHandle, int Width, int
Height, int Contrast, int Brightness);
```

Visual Basic Declaration

```
Sub DT_AdaptiveThresholdBackTrack (ByVal
SessionHandle As Long, ByVal DIBHandle As Long,
ByVal Width As Long, ByVal Height As Long, ByVal
Contrast As Long, ByVal Brightness As Long)
```

Visual Basic .NET Declaration

```
Sub DT_AdaptiveThresholdBackTrack(ByVal
SessionHandle As Integer, ByVal DIBHandle As
Integer, ByVal Width As Integer, ByVal Height As
Integer, ByVal Contrast As Integer, ByVal
Brightness As Integer)
```

Delphi Declaration

```
procedure
DT_AdaptiveThresholdBackTrack(SessionHandle: Integer;
DIBHandle: Integer; Width: Integer;
Height: Integer; Contrast: Integer;
Brightness: Integer); stdcall;
```

Java Declaration

```
void DT_AdaptiveThresholdBackTrack(int
SessionHandle, int DIBHandle, int Width, int
Height, int Contrast, int Brightness);
```

Description

Converts a grayscale image applying dynamic thresholding based on background tracking algo. Each pixel in not homogeneous areas its it's binarized using the local threshold computed automatically using background tracking system. The process can be customized selecting local windows size, local contrast and global brightness.

Parameters

SessionHandle (in) - the session handle to use

DIBHandle: the image handle.

Width: value specifying window width.

Height: value specifying window height.

Contrast: reference value for evaluating homogeneous area, between 0 and 255. Use -1 to allow auto-calculation of the best value for the image.

Brightness: global brightness of resulting image, between 0 and 255. A low value produces a darker images, an high value produces a lighter image. Use -1 to allow auto-calculation of the best value for the image.

Return values

None

Annotation