OMarks – version 1.0

CONTENTS

INTRODUCTION

Library OMarks intended for recognition of optical markers. In version 1.0 supported markers based on barcode Small Aztec. There are different types of licenses for free use, use for yourself and distributing with your software. Library functions include: recognition of the marker on the 8, 24 and 32 bit images, the correction position of the markers (for augmented reality) that is distributed as an alpha version. With the library is an example that is compiled for version OpenCV 2.4.3 (on http://intbusoft.com/omarks/ you can download the necessary libraries to run OpenCV). The main language of the library – C/C++.

1. LICENCES

OMarks FREE. This license is for unlimited free use of library (including in your software) with limited recognition: no more than 3 markers on a single frame, up to 10 characters encoded in each marker.

OMarks SM PRO LIMITED. This is a paid license that provides all the features to recognize markers Small Aztec, but in this case, the library can be used only in your company and on their computers. **OMarks** can not be distributed together with your software and/or hardware.

OMarks SM PRO FULL. This is a paid license that provides all the features to recognize markers Small Aztec, including the distribution, together with your software and/or hardware.

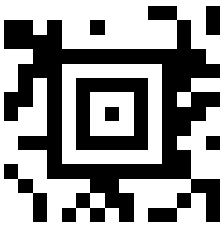
2. INSTALLATION

No installation is required. Need to specify the path to the Dll, lib and h files.

3. DESCRIPTION

a. SUPPORTED MARKERS AND THEIR CODING

In version 1.0. supporting marker type only Small Aztec. This type of bar codes is young enough (standard in 2008), but one of the most effective. Convenient central target allows the detection code even in contact with the edges of the other graphic images. Small Aztec code example:



In the folder images\small aztec\ are 20 examples of markers based Small Aztec. To encode other markers can be used on-line service, or utility zint. You can download the utility at http://sourceforge.net/projects/zint/. For encoding in the program, you can use the command line utility:

zint -b 92 -o t.png -d "Sample Text"

b. LIBRARY FUNCTIONS

*Recognition markers (*OMark.h)

FindMarkers

Function finding optical markers on the input image.

```
int FindMarkers(
        char* Image_Data,
        int Image_Width,
        int Image_Height,
        int* All_Markers,
        char** Data_Result,
        int** Size_Data_Result,
        int** Points4_Mark,
```

int Type_Marker,

int Param

);

Parameters:

Image_Data – input image as an array of image data standard (not inverted, as in BMP) character can be 8, 24, 32 bit, the line is a step in the BMP - a multiple of 4 bytes;

Image_Width – image width;

Image_Height – image height;

All_Markers – a pointer to the maximum number of available output marker, function changes the value by the number of detected markers;

Data_Result – a pointer to a pointer array of character type, in which each marker will return the decoded text, pointers must point to the previously allocated memory area;

Size_Data_Result – pointer to an array of pointers to the buffer sizes **Data_Result**, after performing these values vary by the size of the decoded text;

Points4_Mark – pointer to an array of pointers to arrays of type int with 8 elements, which, after execution of the function will contain the coordinates of the marker on the image in order: X0, Y0, X1, Y1, X2, Y2, X3, Y3;

Type_Marker – type of marker that you want to find in the image, now only supported Small Aztec, so be sure to be – OMARKS_TYPE_SMALL_AZTEC;

Param – flags:

OMARKS_IMAGE8 – 8 bit image (Gray);

OMARKS_IMAGE24 – 24 bit image (BGR);

OMARKS_IMAGE32 – 32 bit image (BGR);

OMARKS_QUALITY_1 – quality recognition 1;

OMARKS_QUALITY_2 – quality recognition 2;

OMARKS_QUALITY_3 – quality recognition 3;

1 to 3 increases the quality, but reduces performance.

The function returns 0 if the markers were not found. 1 - if the detected one or more markers, 2 - unknown format images, 3 - unknown marker type, 4 - unknown error, 5 - recognition error.

*Dynamic properties of the markers (*OMark.h)

These features have not been fully tested and designed for augmented reality, in particular smoothing tremble markers and detection of motion.

DMarksInit

Function initializes the dynamic object.

```
void* DMarksInit(
);
```

The function returns a pointer to the initialized memory area, to work with the dynamic characteristics.

DMarksRelease

Frees the used memory from earlier functions DMarksInit and DMarksProcessed.

```
void DMarksRelease(
    void** Marks
);
```

Parameters:

Marks – a pointer to a previously initialized memory pointer function DMarksInit.

DMarksProcessed

The function to call after call FindMarkers to compare with previous markers and eliminate treble and the disappearance of markers.

```
void DMarksProcessed(
    char* Image_Data,
    int Image_Width,
    int Image_Height,
    void** Marks,
    int* All_Markers,
    char** Data_Result,
    int** Size_Data_Result,
    int** Points4_Mark,
    int flags
);
```

Parameters:

Image_Data – a pointer to the image, such as in FindMarkers;

Image_Width – image width;

Image_Height – image height;

Marks – a pointer to the initialized function DMarksInit memory area;

All_Markers – a pointer to the number of markers found with FindMarkers;

Data_Result – returned array of pointers (function FindMarkers);

Size_Data_Result – returned FindMarkers decoded size markers;

Points4_Mark – returned FindMarkers points of markers;

flags – flags:

OMARKS_IMAGE8 – 8 bit image (Gray);

OMARKS_IMAGE24 – 24 bit image (BGR);

OMARKS_IMAGE32 – 32 bit image (BGR);

DMARK_TREMBLE – eliminate tremble marker between shots;

DMARK_MOTION – recover lost markers in the absence of motion.

Function changes the dynamic object **Marks** and corrects markers returned by the function FindMarkers.

c. SAMPLE OF USE WITH THE OPENCV LIBRARY

The directory samples\CameraOpenCV\ example of interaction with the library OpenCV OMarks. Interface between the library and the example is OMarks file OMarksOpenCV.h. These files show how to work with the library OMarks.

4. TECHNICAL INFORMATION

It is implemented under the Windows platform and has been tested on Windows 7 64 bit and Windows XP. A requirement for the program is to install Microsoft Visual C++ 2010 SP1 Redistributable Package (x86):

http://www.microsoft.com/en-us/download/details.aspx?id=8328

5. CREDITS

IntBuSoft Ltd

http://intbusoft.com