



Title: Credit Card Character Recognition

Purpose

This application note demonstrates how the Gocator can be used to inspect and recognize characters on a credit card.

Equipment Used

Gocator 2330 with Firmware Release 3.2 or later
Halcon Version 10.0 or later

Sensor Features Used

Whole Part, GenTL driver

Table of Contents

1 Overview	2
2 Hardware and Setup	3
3 Halcon OCR Example.....	4
4 Conclusion	6



1 Overview

Letters and numbers are embossed on credit cards to work with non-electronic credit card readers. The embossed characters must maintain a consistent and authentic look and feel and are subject to tight conformance tolerances. Traditionally, classic 2D vision has been used to inspect embossed characters on credit cards, but it lacks the ability to verify the raised profile of the characters.

The 3D point cloud (height map) from the Gocator can be used to inspect the dimensions (width, length and depth) and locations of the embossed characters. This document explains how to recognize the characters on the credit card from the height map data.

This application uses Halcon to process the height map data. Halcon is a comprehensive software package for machine vision applications with an integrated development environment. The Gocator includes a GenICam / GenTL driver that can be used to stream data into Halcon in real-time.

This document assumes the user is familiar with setting up the Gocator to work with Halcon. Refer to the application note *Interfacing a Gocator to Halcon* for more details.



2 Hardware and Setup

Embossed numbers are typically 3mm wide, 5mm long and 500um high (depth). The Gocator 2330 is the most suitable device within the 2300 family because it has the highest X and Z resolutions (X resolution: 44um, Z resolution: 6um).

The Gocator 2330 is setup to generate height maps as the credit cards pass through the sensor.



Figure 1. Physical Setup

A height map is a 2D image with each pixel value representing the height of the surface. The position of each pixel corresponds to a fixed X and Y location in real-world coordinates.

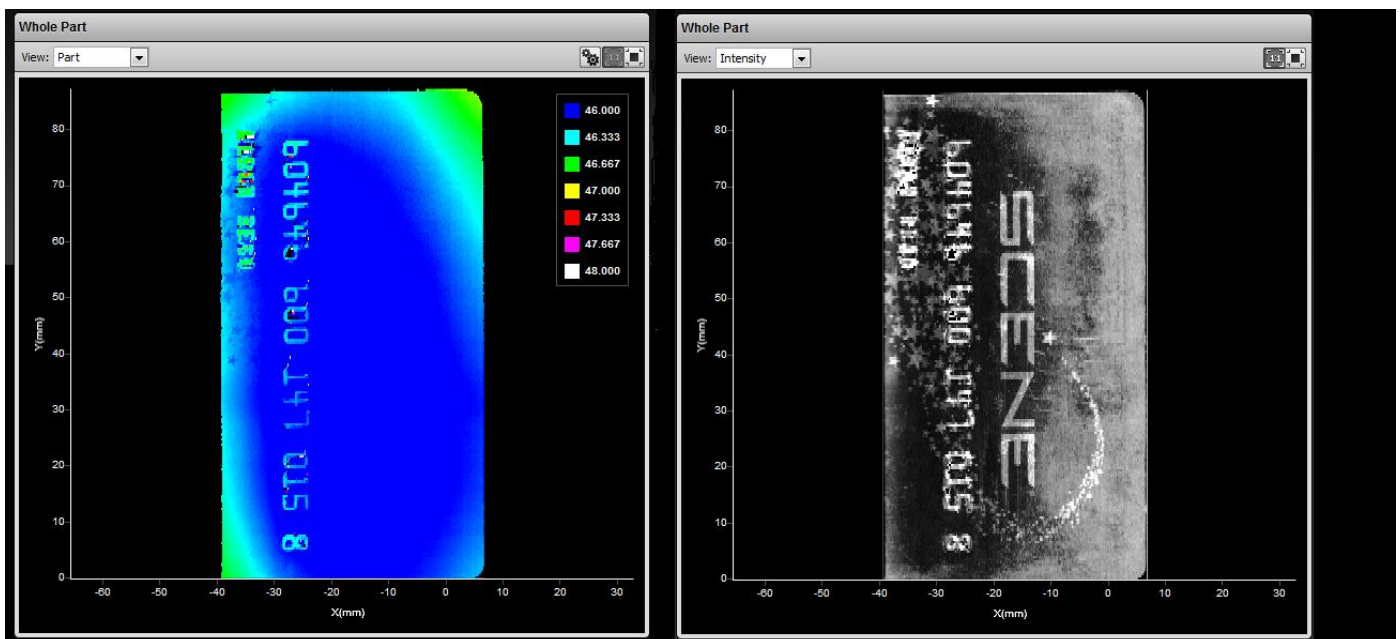


Figure 2. Height Map (Left) and Intensity (Right)



3 Halcon OCR Example

The example code included with this application note uses Halcon's OCR engine to perform character recognition. The example executes the following steps for each credit card scan:

Note: The example code and the associated data files are included in the application note package.

1. Acquire Gocator data using `grab_image_async`.

```
grab_image_async (ScanImage, AcqHandle, -1)
```

2. Extract the height map, encoder position and resolution information from the data using `Go2GenTL_ParseData`. `Go2GenTL_ParseData` is a procedure supplied with the Gocator GenTL example code.

```
Go2GenTL_ParseData(ScanImage, ScanHeightMapNonRotate, ScanIntensityNonRotate, frameCount, timestamp,  
encoderPosition, encoderIndex, inputs, xOffset, xResolution, yOffset, yResolution, zOffset, zResolution,  
partWidth, partHeight)
```



Figure 3. Height map after thresholding

3. Reveal the embossed characters by thresholding the height map. The example uses the median height of the surface as the base of the threshold ranges. Median reliably represents the surface height of the card when the card is flat. The threshold limits are set to:

```
Upper Height Limit = Surface height + 2 * expected character height  
Lower Height Limit = Surface height + expected character height / 5
```

Note: This threshold scheme works well for cards that are flat or very slightly curved. More sophisticated thresholding algorithm will be required if the card surface is curved.

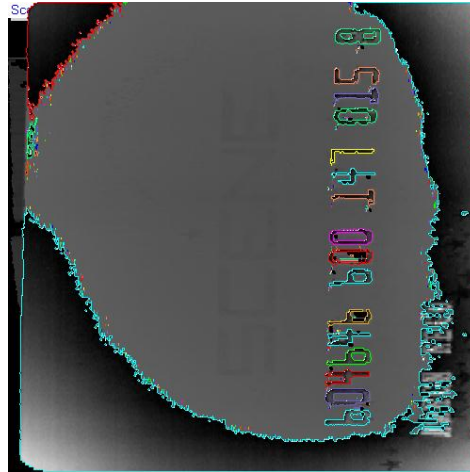


Figure 4. Characters revealed after thresholding

4. Pass the revealed character to the OCR engine. The OCR engine is created with a credit card's font file (*ocr.omc*)

```
do_ocr_multi_class_mlp(ScanHeightMapConnectedRoi, ScanHeightMapReduced, OCRHandle, Class,Confidence)
```

5. Filter results based on the expected character dimensions.

* Only accept connected regions where the dimension is correct.

```
for k :=0 to RegionCount-1 by 1
    tuple_select(rectRow1, k, boxRow1)
    tuple_select(rectRow2, k, boxRow2)
    tuple_select(rectCol1, k, boxCol1)
    tuple_select(rectCol2, k, boxCol2)

    characterBoxWidth := abs(boxRow2 - boxRow1)
    characterBoxLength := abs(boxCol2 - boxCol1)

    if (characterBoxWidth > CharacterWidthPixelLimit)
        if (characterBoxLength > CharacterLengthPixelLimit)
            if (characterBoxWidth < 2*CharacterWidthPixelLimit)
                if (characterBoxLength < 2*CharacterLengthPixelLimit)
                    * Accept results
                    disp_rectangle1(WindowHandle, boxRow1, boxCol1, boxRow2, boxCol2)
                    disp_message(WindowHandle, Class[k], 'image', boxRow1, boxCol1-50, 'blue', 'true')
                endif
            endif
        endif
    endif
endfor
```



Figure 5. OCR Results

4 Conclusion

The example application recognizes embossed characters using the height map from a Gocator sensor. After the characters are recognized, the dimensions and position of each character can be easily extracted from the height map and then inspected for conformance.