SAND REPORT

SAND2004-1870
Unlimited Release
Printed October 2004

Reading Color Barcodes using Visual Snakes

Hanspeter Schaub, ORION International Technologies

Telephone:	(865) 576-8401
Facsimile:	(865) 576-5728
E-Mail:	reports@adonis.osti.gov
Online ordering:	http://www.doe.gov/bridge

Telephone:	(800) 553-6847
Facsimile:	(703) 605-6900
E-Mail:	orders@ntis.fedworld.gov
Online ordering:	http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online

SAND2004-1870
Unlimited Release
Printed October 2004

Reading Color Barcodes using Visual Snakes

Hanspeter Schaub
ORION International Technologies
Albuquerque, NM 87106
schaub@vt.edu

Abstract

Statistical pressure snakes are used to track a mono-color target in an unstructured environment using a video camera. The report discusses an algorithm to extract a bar code signal that is embedded within the target. The target is assumed to be rectangular in shape, with the bar code printed in a slightly different saturation and value in HSV color space. Thus, the visual snake, which primarily weighs hue tracking errors, will not be deterred by the presence of the color bar codes in the target. The bar code is generate with the standard 3 of 9 method. Using this method, the numeric bar codes reveal if the target is right-side-up or up-side-down.

Intentionally Left Blank

Contents

Figures

Tables

Intentionally Left Blank

Reading Color Barcodes using Visual Snakes

Introduction

The statistical pressure snakes are a parametric curve described through a discrete number of control points.[1] These snakes are to be used to identify tool targets and allow a remote manipulator to autonomously align itself to this target and pick it up. The targets are to have a rectangular shape and be essentially a mono-color. To avoid having the hue-based snake wander past the target, it will be surrounded by a border of a distinctly different hue.[2] With this system, it should be possible to align the manipulator gripper (x, y, z) position relative to the target, as well has the heading angle. However, since the target is symmetric and rectangular (makes it easier to check for out-of-plane tilt angles), the heading angle of the target major principal axis is only known to within a sign.

When picking up tools, it can occur that the true heading of the tool needs to be known. Further, it would be convenient if the tool being targeted could be identified. This would allow the autonomous system to check that the manipulator is about to pick up the proper tool. An algorithm is presented in this paper which is able to pick up subtle differences in color on the target and decode the corresponding barcode information, to provide orientation and identification information.

Barcode Character System

To encode a barcode onto the target, the Code 39 set was used (sometimes also referred to as the 3-of-9 set). Other standard barcode sets exist, but typically have a denser set of bars and spaces. These barcodes would make it more difficult for the relatively low pixel resolution that the visual servo camera has. Most barcodes systems were generated to be used with a laser scanner. Such a device is able to scan the black and white barcode at very high rates and measure much finer changes in the barcode than is possible with a video camera. With the Code 39 set, several free fonts exist that can be installed on a computer to generate this barcode characters. Using these fonts, it is possible to generate the barcode with any simple drawing program by selecting the Code 39 font and typing in the desired digits.

Table 1. The Digits 0–9 Represented in the Code 39 Set

Character	Pattern	Bars	Spaces
1	■■□■□□■□■□■■	10001	0100
2	■□■■□□■□■□■■	01001	0100
3	■■□■■□□■□■□■	11000	0100
4	■□■□□■■□■□■■	00101	0100
5	■■□■□□■■□■□■	10100	0100
6	■□■■□□■■□■□■	01100	0100
7	■□■□□■□■■□■■	00011	0100
8	■■□■□□■□■■□■	10010	0100
9	■□■■□□■□■■□■	01010	0100
0	■□■□□■■□■■□■	00110	0100

Table 1 shows the barcode patterns for the digits 0–9. The alpha numeric characters exist as well in this set, but are not used in this application. Every character starts and stops with a bar, accounting for 5 bars total with four spaces in between. Each bar or space can have a width of 1 or 2 units. Only three double wide bars or spaces are present, making the total width of each character 12 units. A small single unit space is set between characters. The numeric representation of the bars and spaces in Table 1 uses 1 to indicate a double width unit, and 0 denote a single width unit. For example, a bars pattern of 10001 would indicate that the first and last bar was a double wide bar.

Note that with the Code 39 set the spacing pattern is equal among all digits. Not shown in this table is that *only* the numerical digits have this particular spacing pattern with the second space always being a double-wide space. The common space pattern results in two benefits in this coding system. First, unless this 0100 spacing pattern is found consistently across all sensed characters, then an error must have occurred. Second, if a 0010 spacing pattern is found across all digits, then the characters must be read from right to left since the digits are upside down. This last feature is very convenient in that it allows us to sense the proper "up" direction of the sensor.

With standard Code 39 barcodes, an asterisk symbol is used as both the start and stop character. Doing so a laser scanner can determine if it needs to read the barcode from left to right or right to left. Since only numerical barcode digits are to be used in the visual targeting barcodes, the start and stop asterisk symbols are not required. In fact, by omitting them, we save the space required for 2 characters and are able to enlarge the dimension the remaining number characters more. The tested routine appeared to work reasonably well with 2 or 3 character barcodes.

Generating Barcodes

To generate the barcode, a standard drawing program like Freehand or CorelDRAW can be used. The Code 39 fonts can be found as freeware on the web, as well as commercial implementations. It is easiest to draw the color tool target with the encoded barcodes all at the same time using these drawing programs.

After selecting a convenient target color (typically a specific hue value with 100% saturation and 100% value in HSV color space), we need to specify the barcode color. To avoid having the snake not recognize the barcodes as being part of the same target, we select the same hue value for the primary barcode color as the target color. The color snakes are the most sensitive to variations in hue, and less sensitive to variations in saturation and value. It is possible to compute precise variations in saturation and value that will be ignored by the snake. However, one must also consider practical limitations of trying to recreate these precise color variations with the color printer. The following recipe was obtained through trial-and-error process by printing the sample color barcodes to an HP color laser printer. What seemed to work best with images obtained from the PXcam camera model (images obtained through the PXC200 frame grabber board) is to reduce the saturation level to 70% to that of the target color, and leave the value at the same level as the target color (typically 100%). When printed to standard laser printer paper, this appeared to produce reasonable results. The color barcodes were subtle enough to allow the standard HSV color snake to ignore it, but they were also pronounced enough to allow the decoding algorithm to have a strong enough signal to process. Three sample targets with color barcodes encoded in them are illustrated in Figure 1.

Decoding Algorithm

A general flow chart of the decoding algorithm is shown in Figure 2. The following text will explain more complex components in further detail.

After having acquired an image and the center of mass of the snake, as well as the heading angle and the estimated box target width and height, certain image pixels are sample to obtain a one-dimensional reading of the barcode information. Figure 3 illustrates which image pixels are sampled for the barcode information. Given the measured target box length l in units of pixels, the maximum line length that is searched for the barcode is computed using

$$cmax = (\text{int})\, l - 20 \tag{1}$$

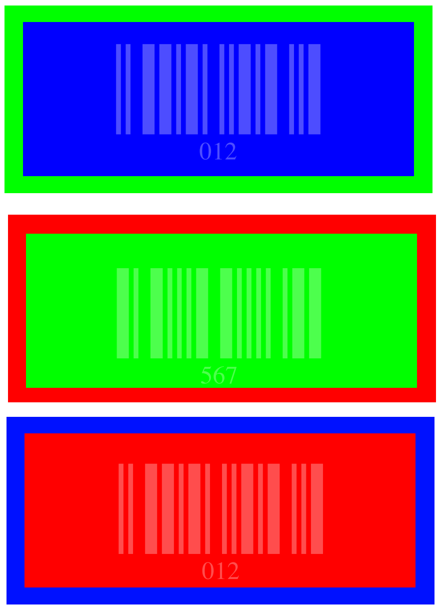


Figure 1. Illustration of a Blue, Green and Red Color Targets with Encoded Barcodes

Given the snake center of mass and major principal axis orientation, we are able to perform a one-dimensional bar code scan of the *cmax* pixel locations. However, due to the relatively crude pixel size, discretization issues are easily encountered. The thin barcode lines may not have a strong signal where it intersects the major principal axis line. To obtain a stronger barcode signal when computing the signal at each bar code pixel, we sample the *dmax* pixel above and below this pixel (i.e. along the minor principal axis direction). Given the target box height h, the integer pixel number *dmax* is computed using

$$dmax = (\text{int})\,\frac{h}{2} \tag{2}$$

By cross-sampling the pixels to obtain a one-dimensional barcode scan, a less noisy and stronger barcode signal is found. However, care should be taken that the cross-sampling value *dmax* is not too large. Increasing this term will make the algorithm more sensitive to heading θ computation errors.

With the OpenCV HSV colors, a saturation number of 0 is a white color and a full saturation is a rich color. A value number of 0 is black while a full value is a bright color. When using the above coloring scheme for the barcodes, and after printing them to paper using a color laser scanner, the acquired video image of the target showed the barcodes to

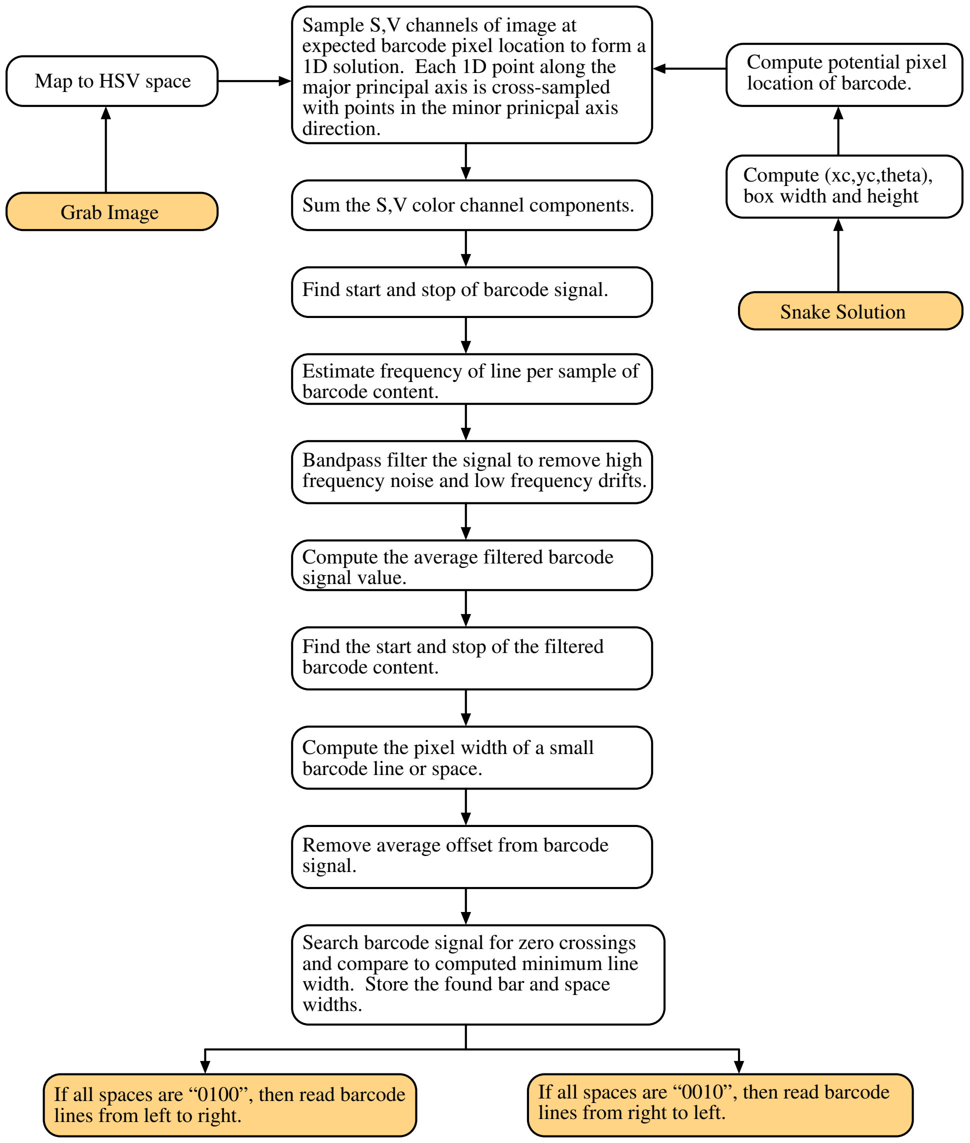

Grab Image
Map to HSV space
Sample S,V channels of image at expected barcode pixel location to form a 1D solution. Each 1D point along the major principal axis is cross-sampled with points in the minor prinicpal axis direction.
Snake Solution
Compute (xc,yc,theta), box width and height
Compute potential pixel location of barcode.
Sum the S,V color channel components.
Find start and stop of barcode signal.
Estimate frequency of line per sample of barcode content.
Bandpass filter the signal to remove high frequency noise and low frequency drifts.
Compute the average filtered barcode signal value.
Find the start and stop of the filtered barcode content.
Compute the pixel width of a small barcode line or space.
Remove average offset from barcode signal.
Search barcode signal for zero crossings and compare to computed minimum line width. Store the found bar and space widths.
If all spaces are "0100", then read barcode lines from left to right.
If all spaces are "0010", then read barcode lines from right to left.

Figure 2. General Flowchart of Barcode Reading Algorithm

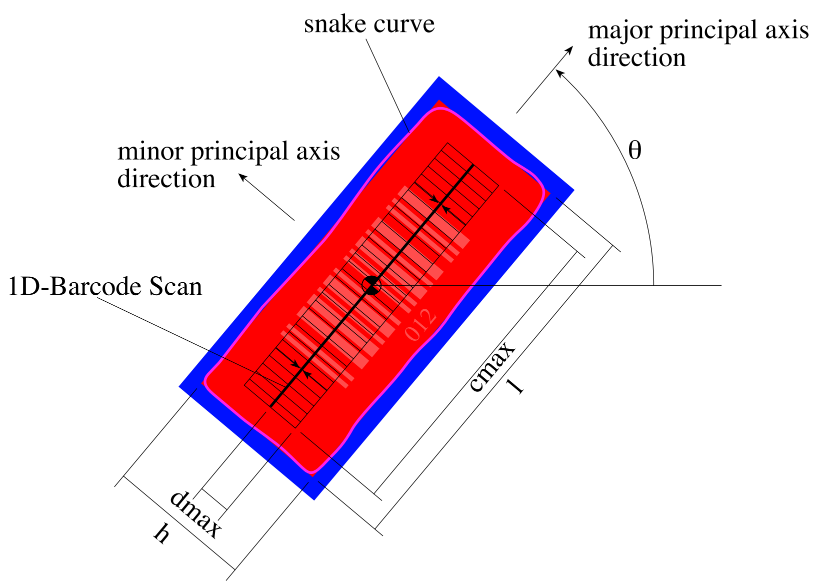

snake curve
major principal axis direction
θ
minor principal axis direction
1D-Barcode Scan
012
cmax
l
dmax
h

Figure 3. Illustration of the Method used to Sample the Image Pixels for the Barcode Signal.

have a lower saturation value S and a higher brightness V. The average barcode reading $\bar{\sigma}$ is then computed using

$$\bar{\sigma}[i] = V[i] + (255 - S[i]) \tag{3}$$

Since the HSV colors are stored as 2 byte values, 255 is a full value while 0 is an empty color value. Using this scheme, a barcode line will always show up as a positive increase in $\bar{\sigma}$.

Next, the start and end of the barcode signal is searched within the array of barcode signals $\bar{\sigma}[i]$. Starting from either end, the current values $\bar{\sigma}[i]$ are compared to values 5 pixels back (either $\bar{\sigma}[i-5]$ or $\bar{\sigma}[i+5]$, depending from which end we started). If a positive jump of 15 pixels is detected, then this is declared the beginning pixel $c1$ or the end pixel $c2$. This provides us with a first estimate how long (in units of pixels) the measured barcode is.

The next step is to pass the measured signals $\bar{\sigma}$ through a bandpass filter. The entire array is filtered using a first order digital bandpass filter of center frequency fc and bandwidth BW. Let nc be the number of barcode characters, then the center frequency is computed using

$$fc = \frac{10nc - 1)}{2(c2 - c1)} \tag{4}$$

The bandpass bandwidth is set to BW = 0.5 Hz.

After filtering the data, the barcode start and stop pixel locations $c1$ and $c2$ are recomputed using the above scheme. Since each barcode character is 12 units long (5 bars and 4 spaces, three of which are double wide), and two characters are separated by a single width space, the unit length of a character dc is estimated using

$$dc = \frac{c2 - c1 + 1}{12nc + (nc - 1)} \tag{5}$$

The next process is the most crucial step in the decoding algorithm. We we take the filtered barcode signals and attempt to extract if a bar or a space has been measured, as well as the width of these items. We know that at $\bar{\sigma}[c1]$ the barcode signal begins and that it ends at $\bar{\sigma}[c2]$. A loop is created to check all the values between $c1$ and $c2$. Since we are starting out with a line signal, to determine the width of the line we are looking for a zero crossing. As this occurs, the line width δ is computed to sub-pixel accuracy by using a linear interpolation between the two pixel values at the zero crossing. This width is then compared to the computed minimum line width dc. If the bar is two units wide, then the width should be $2 * dc$, if the unit is 1 unit wide, then the width should be simple dc. To account for printing errors and video image based width estimation errors, the following algorithm is used:

$\delta > 1.5dc$	wide line or space
$\delta < 1.5dc$	narrow line or space

After the first zero crossing has been detected, then we check for the next zero crossing to determine the space width. This process is continued until the pixel location $c2$ is reached. Figure 4 illustrates this zero crossing checking using some actual data. Here the upside down “012” characters were sensed using the above algorithm. The plot shows the filtered barcode sensor data. Every positive hump represents a barcode line, while each dip represents a barcode space.

Having determined how many lines and spaces are present, and what their respective widths are, we are ready to decode the characters. The first check is to see the behavior of the barcode spaces. They should all be the same for numerical digits. If the second slot is always the sole double-wide space, then the numerical digits should be read from left to right. If the third slot is the sole double-wide space, then the digits are to be read from

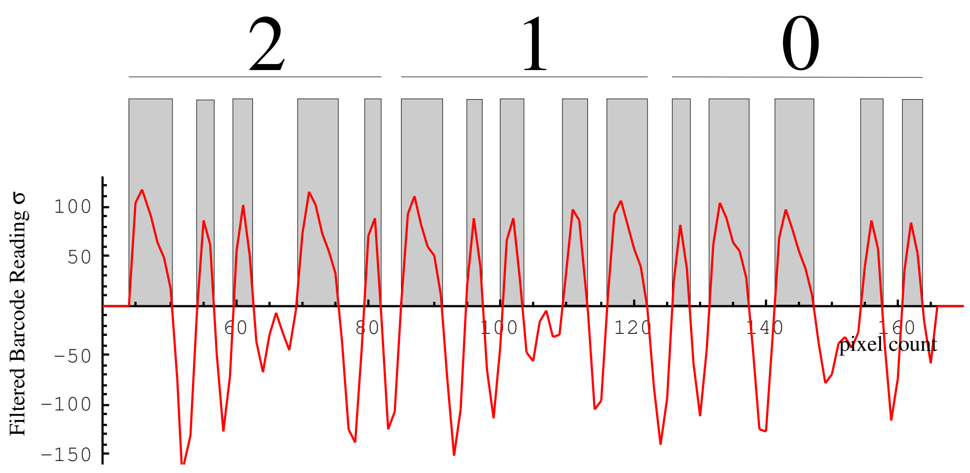


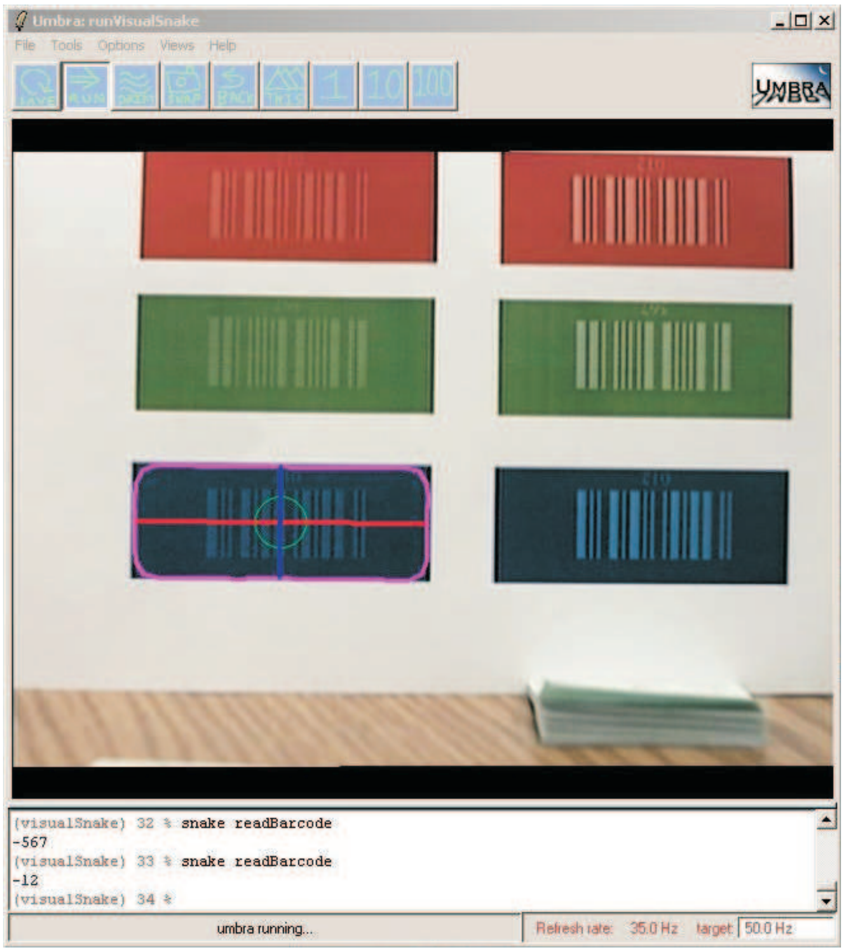
Figure 4. Illustration of the Barcode Line Detection Algorithm.

right to left because the image is upside down. If this barcode spacing is not found, then a sensing error has occurred. The most likely reason is that the barcode is too small on the screen, thus blurring the lines together. Another common reason is an out of focus camera. Finally, the lighting might also have an influence if sharp changes occur across the barcode.

Figure 5 shows how the final UMBRA snake module is able to acquire color bar codes. The target bar codes are placed here upside down. This is the reason why the returned barcode values are negative. The barcode reading shown in Figure 4 is the top left red barcode shown in this image.

Conclusion

This paper presents a technique to use read color based numerical barcodes with the help of color statistical pressure snakes. The barcode is setup by varying the saturation value only of the target color. The actual printout typically has both saturation and value color changes in HSV space. The presented algorithm is able to routinely read at least 2-3 targets of a typical tool pick up sized target.



```
(visualSnake) 32 % snake readBarcode
-567
(visualSnake) 33 % snake readBarcode
-12
(visualSnake) 34 %
```

Figure 5. Illustration of an UMBRA module having a snake segment a target region and the barcode routine successfully reading the encoded digits.

References

[1] Perrin, D. P. and Smith, C. E., “Rethinking Classical Internal Forces for Active Contour Models,” *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, December 2001, pp. 615–620.

[2] Schaub, H., “Statistical Pressure Snakes Based on Color Images,” Technical report, Sandia National Labs, Albuquerque, NM, 2003.

DISTRIBUTION:

1 Virginia Polytechnic Institute
Attn: Hanspeter Schaub
Aerospace and Ocean Engineering Department
228 Randolph Hall
Blacksburg, VA 24061-0203

1 MS 1125
Phil Bennett, 15252

1 MS 1125
Dan Marrow, 15252

1 MS 1125
Robert J. Anderson, 15252

1 MS 9018
Central Technical Files, 8940-2

2 MS 0899
Technical Library, 4916