오진수를 이용한 RGB QR코드 구현에 관한 연구

Study on the Implementation of RGB QR Codes Using Quinary Number

세종고등학교
30707 김지성, 20801 강희원, 20104 김우진, 21004 문평안, 20526 홍순규

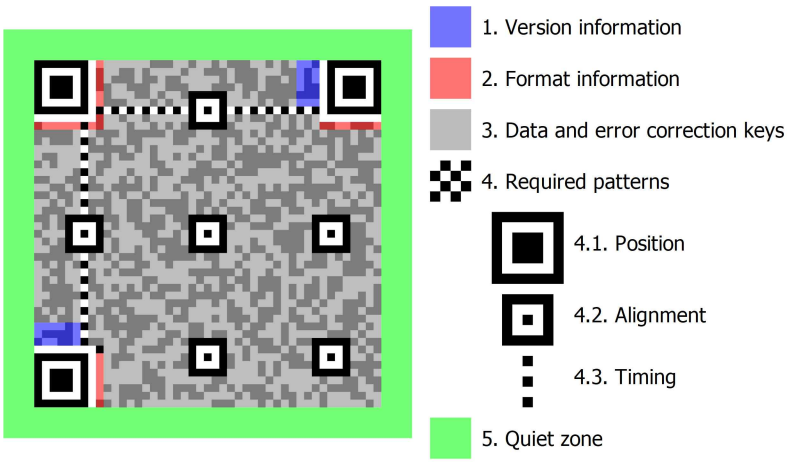
2021.01

I. 사전 탐구

I -1) QR코드

바코드는 컴퓨터가 정보를 인식하기 쉽게 굵기가 다른 흑백의 막대를 조합하여 문자나 숫자를 코드화 한 것이다. 이러한 바코드는 저장할 수 있는 정보량의 한계가 있었다. 점점 저장해야 할 정보량이 늘어남에 따라 여러 바코드를 나열하는 대안들이 제시되었고, 이는 2차원 매트릭스 형태의 바코드로 발전하였다. 이렇게 발전한 2차원 바코드 중 하나가 QR 코드이다. 초기에는 상품관리에 이용되어 기존의 바코드를 대체하는 개념으로 많이 보급되었다가 다양한 인쇄 매체에 인쇄해 인터넷 정보를 쉽게 확인할 수 있는 수단으로 발전하였다. 최근에는 결제 수단으로 활용되기도 한다.

데이터의 표현과 읽기를 수월하게 하기 위해 위치 검출 패턴(모든 방향에서 판독이 가능하도록 하는 영역), 얼라인먼트 패턴(코드의 크기가 커질 경우 왜곡을 줄이기 위한 영역), 데이터 영역(에러 정정 코드 영역) 등의 영역이 나뉘어 있다.



I -2) 아스키코드

아스키코드는 1962년 안시가 정의한 미국 표준 정보 교환 코드이다. 지금의 미국 국가 표준 협회(ANSI)의 전신인 미국 표준 협회(ASA)가 주도한 X3 위원회가 개발했다. 그 아래

의 X3.2 소위원회는 1960년 10월 6일 아스키 표준화 작업을 시작하여, 1963년 표준화 초판을 발간했고, 1967년 개정하였으며, 가장 최근의 업데이트는 1986년에 있었다. 아스키코드는 7비트의 이진수 조합으로 만들어져 총 128개의 부호를 표현한다. 아스키코드는 영문 알파벳을 사용하는 문자 인코딩이며, 컴퓨터와 통신 장비를 비롯한 문자를 사용하는 많은 장치에서 사용되며, 대부분의 문자 인코딩이 아스키코드에 기초를 두고 있다.

10진수	부호	10진수	부호	10진수	부호	10진수	부호
032		056	8	080	P	104	h
033	!	057	9	081	Q	105	i
034	"	058	:	082	R	106	j
035	#	059	;	083	S	107	k
036	$	060	<	084	T	108	l
037	%	061	=	085	U	109	m
038	&	062	>	086	V	110	n
039	'	063	?	087	W	111	o
040	(	064	@	088	X	112	p
041	)	065	A	089	Y	113	q
042	*	066	B	090	Z	114	r
043	+	067	C	091	[	115	s
044	,	068	D	092	\	116	t
045	-	069	E	093	]	117	u
046	.	070	F	094	^	118	v
047	/	071	G	095	_	119	w
048	0	072	H	096	`	120	x
049	1	073	I	097	a	121	y
050	2	074	J	098	b	122	z
051	3	075	K	099	c	123	{
052	4	076	L	100	d	124	|
053	5	077	M	101	e	125	}
054	6	078	N	102	f	126	~
055	7	079	O	103	g		

<출력 가능한 아스키코드 문자표>

Ⅱ. 탐구 설계

우선 기존 2차원 바코드의 저장 용량 한계를 개선하기 위해서 검정색, 하얀색만을 사용하는 기존의 방식에 빨간색, 초록색, 파란색의 3가지 색상을 추가하여 총 5가지 색을 사용하기로 하였다. 더 많은 색을 사용하지 않고 검정, 하양, 빨강, 초록, 파랑의 5가지 색만

사용하는 이유는, 실제 카메라로 인식할 때 각 셀 간의 색 차이를 확실하게 두어 안정적으로 인식시키기 위함이다.

2가지의 색을 사용하는 것보다 5가지 색을 사용함으로써 한 문자를 표현할 때 필요한 셀의 수를 줄일 수 있다. 95개의 문자를 표현하는데 기존의 이진 QR코드로는 7셀($2^6 < 95 < 2^7$)이 필요하지만, RGB 데이터 매트릭스는 3셀($5^2 < 95 < 5^3$)로 이를 모두 표현할 수 있다.

입력 데이터는 아스키코드 내의 문자들로 이루어진 문자열로 한정하고 문자열을 5색 QR 코드로 바꾸는 과정을 구상해보았다. 먼저, 입력한 문자열의 각 문자를 ASCII 코드를 이용하여 십진 숫자 데이터로 변환한다. 이 경우 ASCII 코드로 표현 가능한 문자의 개수가 95개이므로 데이터의 범위는 0~94이다. 다음 이 10진 데이터를 5진 데이터로 진수 변환을 한다. 0~94의 데이터를 5진수로 변환하면 한 문자당 3자리 5진수로 변환이 된다. 이 세 자리 숫자들을 일렬로 배열하고 0~4에 각각 하나의 색상을 대응시켜 이미지로 변환한다.

Ⅲ. 탐구 결과

Ⅲ-1) 소스 코드

<인코딩 코드>

```
import cv2
import numpy as np

NOTATION = '0123456789ABCDEF'
def numeral_system(number, base):  #5진수 변환
    q, r = divmod(number, base)
    n = NOTATION[r]
    return numeral_system(q, base) + n if q else n

def red(x, y) :
    img.itemset((y, x, 2), 255)
def green(x, y) :
    img.itemset((y, x, 1), 255)
def blue(x, y) :
    img.itemset((y, x, 0), 255)
def black(x, y) :
    pass
def white(x, y) :
    img.itemset((y, x, 0), 255)
    img.itemset((y, x, 1), 255)
    img.itemset((y, x, 2), 255)
```

```
#####main#####

sentence = input()      #입력 문자열
Data = []               #ASCII 변환 5진 리스트
DivData = []            #각 자리 쪼갠 리스트
length = len(sentence) #문자열 길이
size = int((length*3)**0.5+1)

for i in range(length) :  #ASCII 변환 후 5진수로 변환하여 Data에 저장
    Data.append(numeral_system(ord(sentence[i])-32, 5)) #32~126 -> 0~94로 변환

for i in range(length) :  #자리 쪼개어 DivData에 각각 3칸씩 저장
    if int(Data[i])<100 :
        DivData.append('0')
        DivData.append(str(Data[i])[0])
        DivData.append(str(Data[i])[1])
    else :
        DivData.append(str(Data[i])[0])
        DivData.append(str(Data[i])[1])
        DivData.append(str(Data[i])[2])

img = np.zeros((size, size, 3), np.uint8)  #qr코드 담을 빈 배열 생성
cnt = 0
for y in range(size) :                      #배열 전체 돌며 픽셀 별 데이터 입력
    for x in range(size) :
        if cnt<length*3 :
            color = int(DivData[cnt])
            cnt+=1
        else :
            break
        if color == 0 :
            red(x, y)
        elif color == 1 :
            green(x, y)
        elif color == 2 :
            blue(x, y)
        elif color == 3 :
            black(x, y)
        elif color == 4 :
            white(x, y)
```

```
if size**2 < length*3 :                          #초과 입력 알림
    print("Data loss")

bigimg = cv2.resize(img, dsize = (500, 500), interpolation = cv2.INTER_NEAREST)
#qr 코드 (500, 500) 으로 확대

cv2.imshow('Image', bigimg)  #qr코드 표시
cv2.waitKey(0)  #키보드 입력 대기
cv2.destroyAllWindows()
cv2.imwrite('images/image.bmp', bigimg)  #.bmp 저장
print('Saved')
```

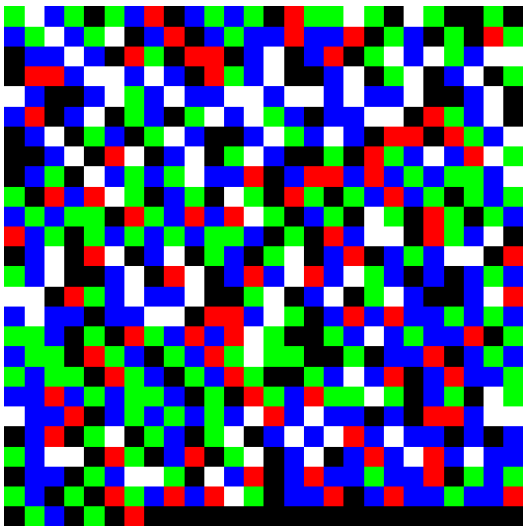
<디코딩 핵심 코드>

```
def decorder(image, x, y):
    if(image.item(y, x, 2)==255 and image.item(y, x, 1)==0):
        return 0
    elif(image.item(y, x, 1)==255 and image.item(y, x, 2)==0):
        return 1
    elif(image.item(y, x, 0)==255 and image.item(y, x, 2)==0):
        return 2
    elif(image.item(y, x, 0)==255 and image.item(y, x, 1)==255 and
                                                    image.item(y, x, 2)==255):
        return 4
    elif(image.item(y, x, 0)==0 and image.item(y, x, 1)==0 and image.item(y, x, 2)==0):
        return 3


Decode=str()
size = 21                          #사이즈 고정(위 인코딩 코드와 안 맞음)
img = cv2.imread('image.png')
cv2.imshow('ss', img)

orgimg = cv2.resize(img, dsize = (size, size), interpolation = cv2.INTER_NEAREST)
#확대된 이미지 원본으로 축소

for y in range(size):#번역을 위한 반복문
    for x in range(0,7) :
        Decode = Decode + chr(decorder(img, 3*x, y)*25 + decorder(img, 3*x+1, y)*5
+ decorder(img, 3*x+2, y))) #번역
print(Decode)
```

Ⅲ-2) 프로그램 구성

먼저 인코딩부분을 보면 QR코드로 변환할 문자열을 입력받고, 문자열의 문자 각각을 아스키코드를 이용해 0~94의 십진수의 숫자들로 변환하고, 이를 다시 3자리의 5진수 숫자들로 변환한다. 다음, 5진수 각각의 자리를 쪼개어 한 5진수를 3개의 0~4의 수로 분리하고 배열에 저장한다. 이후, 숫자 0은 빨강, 1은 초록, 2는 파랑, 3은 검정, 4는 하양으로 설정하고, OpenCV 라이브러리를 이용하여 각 픽셀에 정해진 색들을 채워 넣는다. 이때 이미지의 사이즈는 $\sqrt{3\times(\text{문자열 길이})}+1$의 정수부분을 한 변으로 하는 정사각형이 된다. 이 이미지를 사용하기 편하게 500*500의 해상도로 INTER_NEAREST 보간법을 이용해 확대하고 저장한다.

디코딩 부분을 보면, 먼저 디코딩할 이미지를 불러오고 확대된 이미지를 원래 사이즈로 축소시킨다. 그 후 세 픽셀씩 모아 미리 설정해둔 색에 대응하는 숫자로 5진수를 10진수로 변환시킨다. 이를 아스키코드를 이용해 문자로 변환시키고 이들을 모아 문자열을 출력한다.



<임의의 문자열 220자를 QR코드로 인코딩한 결과>

Ⅳ. 결론 및 제언

본 탐구는 QR코드의 정보 저장의 효율성을 높이기 위해 검정, 하양만을 사용하던 기존 이진 체계에서 색상을 다양화하여 5가지 색상을 사용한 5진 체계를 도입하였다. 앞서 공부한 영상처리, 진수변환, Python 프로그래밍 등의 지식을 활용하여 RGB QR코드를 구현하기 위해 어떠한 과정을 거쳐야할지에 대한 고민이 많았다. 특히 정보의 효율적인 저장을 위해 압축 기법에 대해 고민해보았지만 유의미한 성과가 나오지 못한 것은 아쉬운 부분이다. 아직 인코딩과 디코딩의 기본적인 단계까지 밖에 구현하지 않았지만 이후 컴퓨터 공학

적 지식을 많이 쌓아 카메라를 활용한 RGB QR코드 인식, 압축 알고리즘 적용, 얼라인먼트 마크 적용 등을 구현하고 싶다.