ESI for

Quantitative analysis of s-PB/SBR blends dispersion morphology using computer image processing-assisted Raman spectroscopic technologies

Meng Ge [a,d], Junqing Wu [b,d], Qingqing Hong [c,d], Lifeng Zhang [a,d,*], Ming Zhang [b,d],
Lei Yu [b,d,*]

[a] Department of Electrical and Electronic Engineering, Kyushu Institute of Technology, Kitakyushu, Fukuoka 8048550, Japan

[b] School of Chemistry and Chemical Engineering, Yangzhou University, Yangzhou, Jiangsu 225002, China

[c] School of Information Engineering, Yangzhou University, Yangzhou, Jiangsu 225002, China

[d] Joint Laboratory of Yangzhou University and Kyushu Institute of Technology, Yangzhou University, Yangzhou, Jiangsu 225002, China

Fax : +86-514-87979516; Tel: +86-514-87979516; Emails: zhang@elcs.kyutech.ac.jp (L. Zhang) ; yulei@yzu.edu.cn (L. Yu).

CONTENTS

1. Computer programs to quantify homogeneity

```
import cv2
from numpy import *
import numpy as np
import math

import random
from random import uniform, randint

def findrate(square):
    h, w = square.shape[:2]
    s = 0.0
    amount = 0.0
    for i in range(0,h):
        for j in range(0,w):
            amount += 1
            if square[i,j] == 255:
                s = s+1
    rate = s/amount
    return rate

def meanrate(fdomain, r=20):
    h, w = fdomain.shape[:2]
    coord = 0
    coor = 0
    for i in range(0,h-r+1,10):
        for j in range(0, w-r+1, 10):
            coor += 1
    sample_rate = np.empty(coor)
```

```
        for i in range(0,h-r+1,10):
                for j in range(0, w-r+1, 10):


                        sample = fdomain[i:i+r-1, j:j+r-1]
                        sample_rate[coord] = findrate(sample)


                        coord = coord + 1


        var = np.cov(sample_rate)
        a = findrate(fdomain)
        k = 100*(math.sqrt(var))/a
        return k


img = cv2.imread('image path')


cv2.namedWindow('Original', 0)
cv2.resizeWindow('Original',500,500)
cv2.imshow('Original', img)
imag = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
rows, cols = imag.shape[:2]
lower_red_1 = np.array([0,50,50])
upper_red_1 = np.array([34,255,255])
mask_1 = cv2.inRange(imag, lower_red_1, upper_red_1)


lower_red_2 = np.array([160,50,50])
upper_red_2 = np.array([180,255,255])
mask_2 = cv2.inRange(imag, lower_red_2, upper_red_2)
nimg = np.zeros((rows, cols))
```

```
for m in range(0,rows):

        for n in range(0,cols):

                nimg[m,n] = mask_1[m,n] or mask_2[m,n]


homo = meanrate(mask_1)

print homo

cv2.namedWindow('Final', 0)

cv2.resizeWindow('Final',500,500)

cv2.imshow('Final', nimg)

cv2.waitKey(0)

cv2.destroyAllWindows()
```