دانشگاه صنعتی اصفهان

دانشکده مهندسی برق و کامپیوتر

عنوان: تکلیف سوم درس مبانی بینایی کامپیوتر

نام و نام خانوادگی: علیرضا ابره فروش

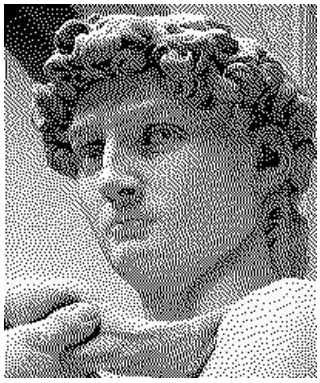
شماره دانشجویی: ۹۸۱۶۶۰۳

نیم سال تحصیلی: بهار ۱۴۰۰/۱۴۰۱

مدرّس: دکتر نادر کریمی

دستیاران آموزشی: بهنام ساعدی - محمدرضا مزروعی

۱



شکل ۱: خروجی الگوریتم *Floyd-Steinberg*

شکل ۲: خروجی الگوریتم حریصانه

۲

۳

۱.۳ Algorithm

ابتدا تصاویر آبی و قرمز معیار برای ارقام ۱ تا ۹ تولید می‌کنیم (پیش‌پردازش). هر تصویر را می‌خوانیم و در هر مرحله (تا زمانی که پیکسل آبی یا قرمز وجود داشته باشد) موقعیت مکانی رقم رنگی‌ای که بالاتر و چپ‌تر است را پیدا می‌کنیم و مقدار MSE (متناظرا نرم ۱) بین آن قطعه از تصویر و تمام ارقام معیار هم‌رنگ تولید شده در مرحله‌ی پیش‌پردازش (و دارای ابعاد برابر با قطعه‌ی پیدا شده (با استفاده از resize)) را محاسبه می‌کنیم. مینیمم این مقادیر با احتمال بسیار بالا مربوط به رقم مورد نظر است. آن را به آرایه‌ی ارقام درون تصویر اضافه می‌کنیم و سپس آن قطعه را تماما سیاه می‌کنیم تا در مرحله بعد رقم بعدی پیدا شود. به این شکل جمع ارقام موجود در همه تصاویر با دقت ۱۰۰ درصد به دست می‌آید.

۲.۳ Preprocessing

```
clc
clear
close all
imtool close all
%%%%%%%%%%%%%%%%
%generating template image of numbers to compare each element of image with
for k = 0: 9
    I = imread(['images\' num2str(k) '.png']);
    min_row = 1000;
    max_row = 0;
    min_column = 1000;
    max_column = 0;
    for i = 1: size(I, 1)
        for j = 1: size(I, 2)
            if (I(i, j, 1) < 200)
                if (i < min_row)
                    min_row = i;
                end
                if (i > max_row)
                    max_row = i;
                end
                if (j < min_column)
                    min_column = j;
                end
                if (j > max_column)
                    max_column = j;
                end
            end
        end
    end
    Red = uint8(zeros(max_row - min_row + 1, max_column - min_column + 1, 3));
    Blue = uint8(zeros(max_row - min_row + 1, max_column - min_column + 1, 3));
    for i = min_row: max_row
        for j = min_column: max_column
            Red(i - min_row + 1, j - min_column + 1, 1) = 255;
            Red(i - min_row + 1, j - min_column + 1, 2: 3) = I(i, j, 2: 3);
```

```
            Blue(i - min_row + 1, j - min_column + 1, 1: 2) = I(i, j, 1: 2);
            Blue(i - min_row + 1, j - min_column + 1, 3) = 255;
        end
    end

%     imtool(I);
%     imtool(Red);
%     imtool(Blue);
    imwrite(Red, ['images\p' int2str(k) '.png']);
    imwrite(Blue, ['images\n' int2str(k) '.png']);
end
```

۳.۳ Function

```
function sum_of_values = sumOnImage(I)
%SUMONIMAGE Summary of this function goes here
    X = I;
    values = 0;
    digit = 1;
    while(true)
        [min_row, max_row, min_column, max_column, digit] = getDigit(X);
        if (digit == 0)
            break;
        end
        X(min_row: max_row, min_column: max_column, :) = 0;
        %imtool(X);
        values(end + 1) = digit;
    end
    sum_of_values = sum(values);
end

function [min_row, max_row, min_column, max_column, digit] = getDigit(X)
%GETDIGIT Summary of this function goes here
    [min_row, max_row, min_column, max_column, sign] = getBoundaries(X);
    if (min_row == 100000)
        digit = 0;
    else
```

```
        J = X(min_row: max_row, min_column: max_column, :);
        mse_of_numbers = zeros(1, 9);
        if sign == 0
            for i = 1: 9
                num_pic = imread(['images\p' num2str(i) '.png']);
                mse_of_numbers(i) = immse(J, imresize(num_pic, size(J(:, :, 1))));
            end
        else
            for i = 1: 9
                num_pic = imread(['images\n' num2str(i) '.png']);
                mse_of_numbers(i) = immse(J, imresize(num_pic, size(J(:, :, 1))));
            end
        end
        [min_value, digit] = min(mse_of_numbers);
        if sign == 1
            digit = -digit;
        end
    end
end

function [min_row, max_row, min_column, max_column, sign] = getBoundaries(I)
%POPDIGIT Summary of this function goes here
    min_row = 100000;
    max_row = -100000;
    min_column = 100000;
    max_column = -100000;
    sign = 0;
    white_pixel = 255 * ones(1, 1, 3);
    black_pixel = zeros(1, 1, 3);
    top_row = 0;
    top_column = 0;
    break_flag = 0;
    for i = 1: size(I, 1)
        for j = 1: size(I, 2)
            if (~isequal(I(i, j, :), white_pixel) && ~isequal(I(i, j, :), black_pixel))
                top_row = i;
```

```
                top_column = j;
                if (I(i, j, 1) == 255)
                    sign = 0;
                end
                if (I(i, j, 3) == 255)
                    sign = 1;
                end
                break_flag = 1;
                break;
            end
        end
        if (break_flag)
            break;
        end
    end
    break_flag = 0;
    if sign == 0
        for i = top_row: top_row + 50
            if (i <= 0)
                continue;
            end
            for j = top_column - 35: top_column + 35
                if (j <= 0)
                    continue;
                end
                if (~isequal(I(i, j, :), white_pixel) && ~isequal(I(i, j, :), black_pixel
    ))
                    if (i < min_row)
                        min_row = i;
                    end
                    if (i > max_row)
                        max_row = i;
                    end
                    if (j < min_column)
                        min_column = j;
                    end
```

```
                    if (j > max_column)
                        max_column = j;
                    end
                end
            end
        end
    else
        for i = top_row: top_row + 50
            if (i <= 0)
                continue;
            end
            for j = top_column - 35: top_column + 35
                if (j <= 0)
                    continue;
                end
                if (~isequal(I(i, j, :), white_pixel) && ~isequal(I(i, j, :), black_pixel
    ))
                    if (i < min_row)
                        min_row = i;
                    end
                    if (i > max_row)
                        max_row = i;
                    end
                    if (j < min_column)
                        min_column = j;
                    end
                    if (j > max_column)
                        max_column = j;
                    end
                end
            end
        end
    end
end
```

۴.۳ Driver code

```
clc
clear
close all
imtool close all
%%%%%%%%%%%%%%%%
dirinfo = dir("images\Q3");
name_of_images = {dirinfo.name};
ground_truth = zeros(1, 100);
my_result = zeros(1, 100);
box_color = {'green'};
for i = 3: size(name_of_images, 2)
    current_image_name = char(name_of_images(i));
    temp = strsplit(current_image_name, "_");
    kemp = strsplit(char(temp(3)), ".");
    g_t = str2double(kemp(1));
    ground_truth(i) = g_t;
    I = imread(['images\Q3\' current_image_name]);
    J = I;
    m_r = sumOnImage(I);
    my_result(i) = m_r;
    J = insertText(J, [350 756;], num2str(m_r), 'boxColor', box_color, 'FontSize', 22);
    imwrite(J, ['images\A3\' current_image_name]);
end
correct_guesses = my_result == ground_truth;
percentage = 100 * (sum(correct_guesses) - 2) / (size(my_result, 2) - 2)
```

۴

۱.۴ Algorithm

برای هر پیکسل دارای نویز فلفل نمکی (سطح روشنایی ۰ و ۲۵۵) یک کرنل ۳ در ۳ نظر می‌گیریم و آن را با میانگین سطح روشنایی پیکسل‌های همسایه‌اش در کرنل ۳ در ۳ که دارای نویز نیستند (در صورت وجود) مقداردهی می‌کنیم.سطح روشنایی سایر پیکسل‌ها (که یا همه‌ی همسایه‌هاشان نویز هستند یا خودشان فاقد نویز) را بدون تغییر می‌گذاریم. با فرض اینکه تصویر اصلی فاقد سطح روشنایی ۰ یا ۲۵۵ باشد (یا به تعداد کم) الگوریتم را تا جایی که هیچ پیکسل با سطح روشنایی ۰ یا ۲۵۵ باقی نماند تکرار می‌کنیم. این کار در تصاویر با نویز بالا که در آن بسیاری از پیکسل‌ها در همسایگی‌های ۳ در ۳ی خود همسایه‌ی غیر نویز ندارند می‌تواند تا حدی کیفیت تصاویر را ارتقا دهد. توجه شود که در صورتی که تصویر اصلی تعداد زیادی پیکسل ۰ یا ۲۵۵ داشته باشد آنگاه این الگوریتم ناکارآمد است. همچنین با توجه به اینکه ممکن است الگوریتم به تعداد زیادی اجرا شود (با توجه به درصد نویز و میزان

پراکندگی آن) در کاربردهایی که زمان اهمیت دارد می‌تواند نا کارآمد باشد.

۲.۴ Function

```
function K = removeNoise(J)
%REMOVENOISE Summary of this function goes here
    K = J;
    for i = 1: size(K, 1)
        for j = 1: size(K, 2)
            if (K(i, j) == 0 || K(i, j) == 255)
                arr = [];
                for k = i - 1: i + 1
                    for l = j - 1: j + 1
                        if (k > 0 && k < size(K, 1) && l > 0 && l < size(K, 2))
                            if (K(k, l) > 0 && K(k, l) < 255)
                                arr(end + 1) = K(k, l);
                            end
                        end
                    end
                end
                K(i, j) = mean(arr);
            end
        end
    end
end
```

۳.۴ Driver code

```
clc
clear
close all
imtool close all
%%%%%%%%%%%%%%%%%
I = imread("images\Q4\House.tif");
my_psnr_values = [];
med_psnr_values = [];
for d = 0.1: 0.1: 0.9
    J = imnoise(I, 'salt & pepper', d);
```

```
    K = removeNoise(J);
    while (ismember(0, K) || ismember(255, K))
        K = removeNoise(K);
    end
    L = medfilt2(J);
    my_psnr_values(end + 1) = psnr(K, I);
    med_psnr_values(end + 1) = psnr(L, I);
end
my_psnr_values
mean(my_psnr_values)
med_psnr_values
mean(med_psnr_values)
```

۴.۴ Results

مقدار نویز	مقدار PSNR							
	Bridge		**Boat**		**Peppers**		**House**	
	Median	روش شما	**Median**	روش شما	**Median**	روش شما	**Median**	روش شما
۱۰٪	26.4491	35.4940	29.5452	38.3188	32.9933	40.1790	31.6136	41.3155
۲۰٪	24.7149	32.3510	26.8926	35.1236	28.5689	37.1491	27.1770	37.8877
۳۰٪	21.6692	30.3924	22.8737	32.9896	23.3094	34.8541	22.1984	35.5075
۴۰٪	17.8907	28.7389	18.6787	31.4095	18.7179	33.2457	18.3001	33.9004
۵۰٪	14.6175	27.3220	15.2068	29.9644	15.2770	31.7296	15.1437	31.9559
۶۰٪	11.9099	25.8672	12.3400	28.4694	12.1137	30.0131	12.3490	29.7612
۷۰٪	9.6961	24.3418	9.9814	26.6358	9.8964	28.2534	9.8242	28.0489
۸۰٪	7.8965	22.5651	8.1553	24.7047	7.9913	25.8364	8.0907	25.7710
۹۰٪	6.4464	20.1695	6.6309	21.7863	6.4746	22.4505	6.6512	22.4929
میانگین	**15.6989**	**27.4713**	**16.7005**	**29.9336**	**17.2603**	**31.5234**	**16.8164**	**31.8490**

منابع