دانشگاه صنعتی اصفهان

دانشکده مهندسی برق و کامپیوتر

عنوان: تکلیف چهارم درس مبانی بینایی کامپیوتر

نام و نام خانوادگی: علیرضا ابره فروش

شماره دانشجویی: ۹۸۱۶۶۰۳

نیم سال تحصیلی: بهار ۱۴۰۰/۱۴۰۱

مدرّس: دکتر نادر کریمی

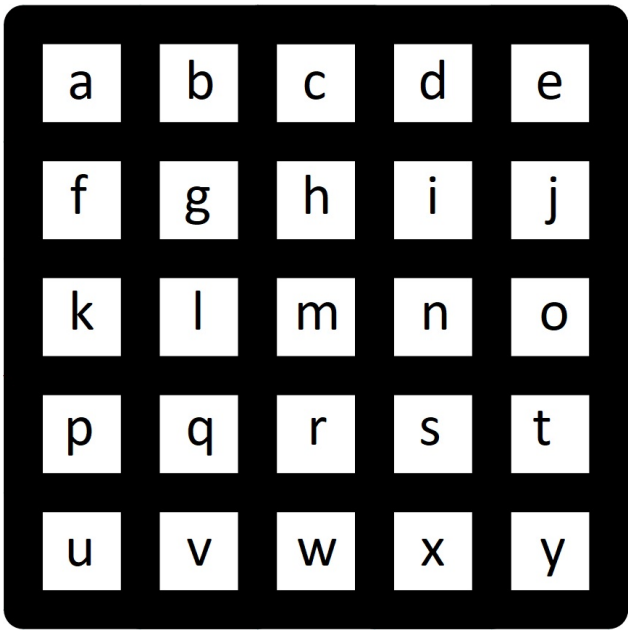
دستیاران آموزشی: بهنام ساعدی - محمدرضا مزروعی

۱

۱.۱ Prewitt

سطح روشنایی پیکسل‌های h، m و r در تصویر فرضی زیر پس از اعمال یک فیلتر هموارکننده‌ی افقی با وزن‌های α، β و γ (نسبت ۱:۱:۱ وزن‌ها (متناظر فیلتر Prewitt در جهت افقی) و نسبت ۱:۲:۱ (متناظر فیلتر Sobel در جهت افقی) حالات خاص هستند) به ترتیب برابر است با $\alpha g+\beta h+\gamma i$، $\alpha l+\beta m+\gamma n$ و $\alpha q+\beta r+\gamma s$. پس از اعمال فیلتر مشتق‌گیر عمودی سطح روشنایی پیکسل m برابر است با $\alpha(q-g)+\beta(r-h)+\gamma(s-i)$.

از طرفی با اعمال فیلتر Prewitt روی تصویر، مقدار پیکسل m برابر با $(q-g)+(r-h)+(s-i)$ می‌شود. از برابری سطح روشنایی‌های به دست آمده از دو روش نتیجه می‌گیریم که اعمال توالی فیلترهای یک بعدی هموارکننده افقی و بعد مشتق‌گیر عمودی معادل با اعمال فیلتر Prewitt در جهت عمودی خواهد بود.



شکل ۱

۲.۱ Sobel

به طریق مشابه سطح روشنایی پیکسل‌های l، m و n در تصویر پس از اعمال یک فیلتر هموارکننده‌ی عمودی با وزن‌های α، β و γ (نسبت ۱:۱:۱ وزن‌ها (متناظر فیلتر Prewitt در جهت افقی) و نسبت ۱:۲:۱ (متناظر فیلتر Sobel در جهت افقی) حالات خاص هستند) به ترتیب برابر است با $\alpha g+\beta l+\gamma q$، $\alpha h+\beta m+\gamma r$ و $\alpha i+\beta n+\gamma s$. پس از اعمال فیلتر مشتق‌گیر افقی سطح روشنایی پیکسل m برابر است با $\alpha(i-g)+\beta(n-l)+\gamma(s-q)$.

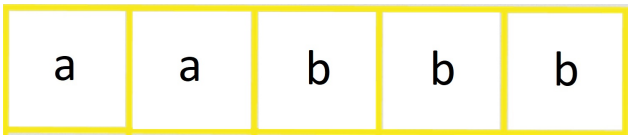
از طرفی با اعمال فیلتر Sobel روی تصویر، مقدار پیکسل m برابر با $(i-g)+2\times(n-l)+(s-q)$ می‌شود. از برابری سطح روشنایی‌های به دست آمده از دو روش نتیجه می‌گیریم که اعمال توالی فیلترهای یک بعدی هموارکننده عمودی و بعد مشتق‌گیر افقی معادل با اعمال فیلتر Sobel در جهت افقی خواهد بود.

۲

حین عبور از لبه‌ها در تصویر، یک جهش در اختلاف سطح روشنایی دو پیکسل مجاور رخ می‌دهد. کرنل زیر را در نظر بگیرید:

شکل ۲: کرنل مشتق دوم افقی

با گذاردن این کرنل از روی لبه، چون همواره پیکسل وسط کرنل یا روی سمت راست لبه است یا سمت چپ لبه، مقدار مشتق دوم در پیکسل قرار گرفته در لبه‌ی با سطح روشنایی کمتر (بیشتر)، مثبت (منفی). در تصویر زیر اگر داشته باشیم $a \geq b$ $(b \geq a)$ مقدار مشتق دوم برای پیکسل وسط برابر $a - b$ می‌شود که با توجه به فرض مقدار مثبت (منفی) است. همچنین مقدار مشتق دوم برای پیکسل دوم از سمت چپ برابر $b - a$ می‌شود که با توجه به فرض مقدار منفی (مثبت) است. بنابراین zero-crossing رخ می‌دهد.



شکل ۳

مثال زیر را در نظر بگیرید.

شکل ۴

شکل ۵

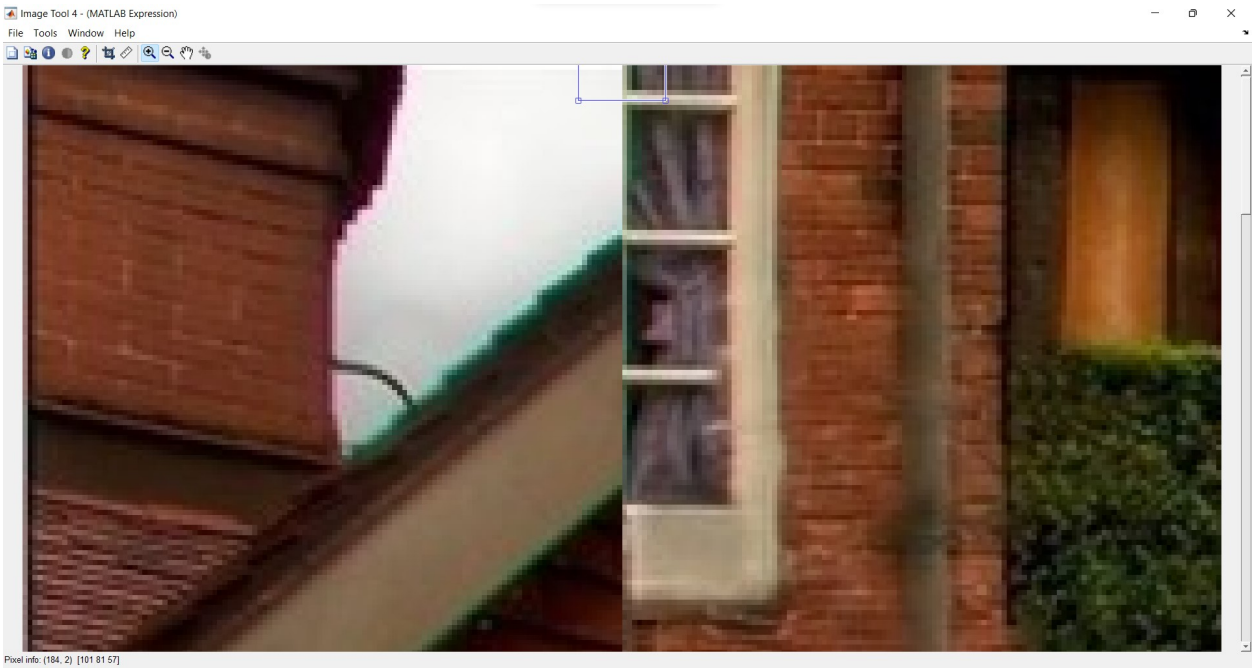
۳

۱.۳ Algorithm

ابتدا ۴ گوشه‌ی تصویر را فیکس می‌کنیم. برای پیدا کردن قطعه‌ی متناظر با موقعیت فعلی از میزان شباهت لبه‌ها با یکدیگر استفاده می‌کنیم. برای سنجش شباهت بین دو قطعه از پازل برای همسایگی در یک سطر (ستون)، بردار ویژگی‌های قطعه‌ی سمت چپ (بالا) را برابر سطح روشنایی‌های پیکسل‌های لبه‌ی راست (پایین) و بردار ویژگی‌های قطعه‌ی سمت راست (پایین) را برابر سطح روشنایی‌های

پیکسل‌های لبه‌ی چپ (بالا) در نظر می‌گیریم. پیمایش تصویر را از گوشه‌ی چپ و بالا شروع می‌کنیم و توان ۲ی فاصله اقلیدسی بردار ویژگی‌های تصویر خاکستری‌گونه‌ی همه‌ی قطعه‌های موجود را با تصویر خاکستری‌گونه‌ی قطعه‌ی فیکس شده حساب می‌کنیم. مینیمم همه این مقادیر مربوط به تصویری است که لبه‌ی آن بیشترین شباهت را از لحاظ سطح روشنایی با لبه‌ی تصویر فیکس شده دارد.

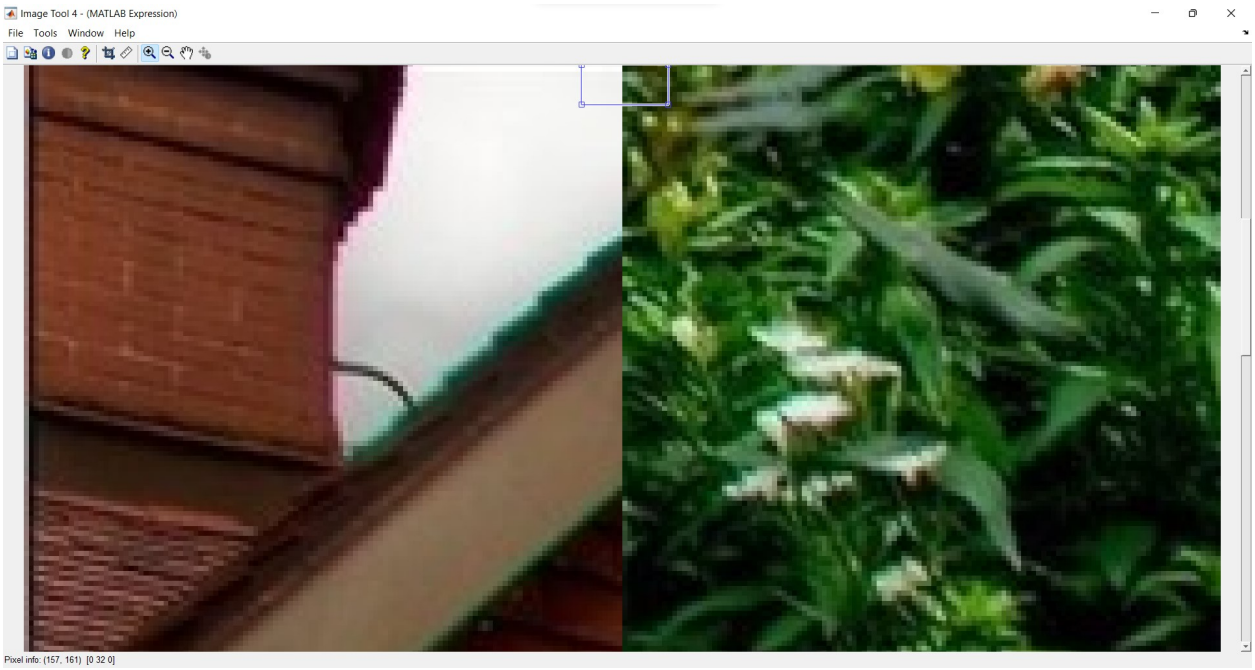
شکل ۶: مقایسه‌ی لبه‌ی راست قطعه‌ی گوشه‌ی چپ بالا با چند قطعه



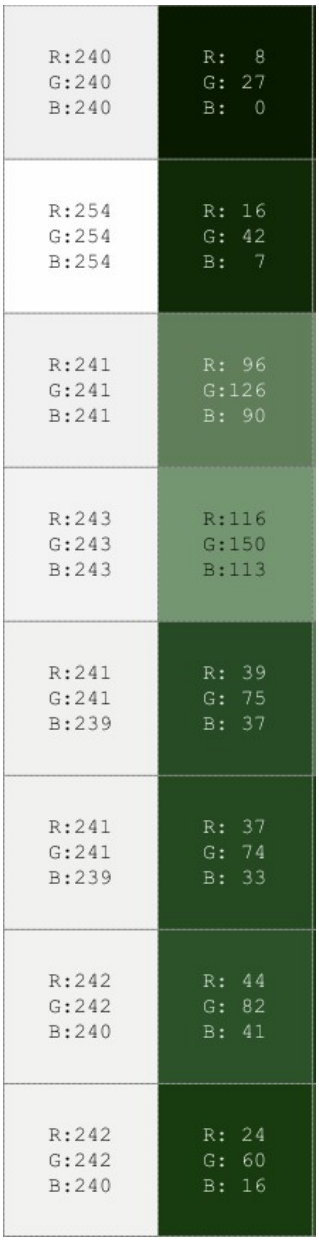
شکل ۷: لبه‌ها (نمای دور)

R:240 G:240 B:240	R:173 G:176 B:159
R:254 G:254 B:254	R:172 G:175 B:156
R:241 G:241 B:241	R:175 G:178 B:157
R:243 G:243 B:243	R:176 G:178 B:156
R:241 G:241 B:239	R:177 G:179 B:155
R:241 G:241 B:239	R:182 G:185 B:158
R:242 G:242 B:240	R:196 G:196 B:168
R:242 G:242 B:240	R:202 G:205 B:176

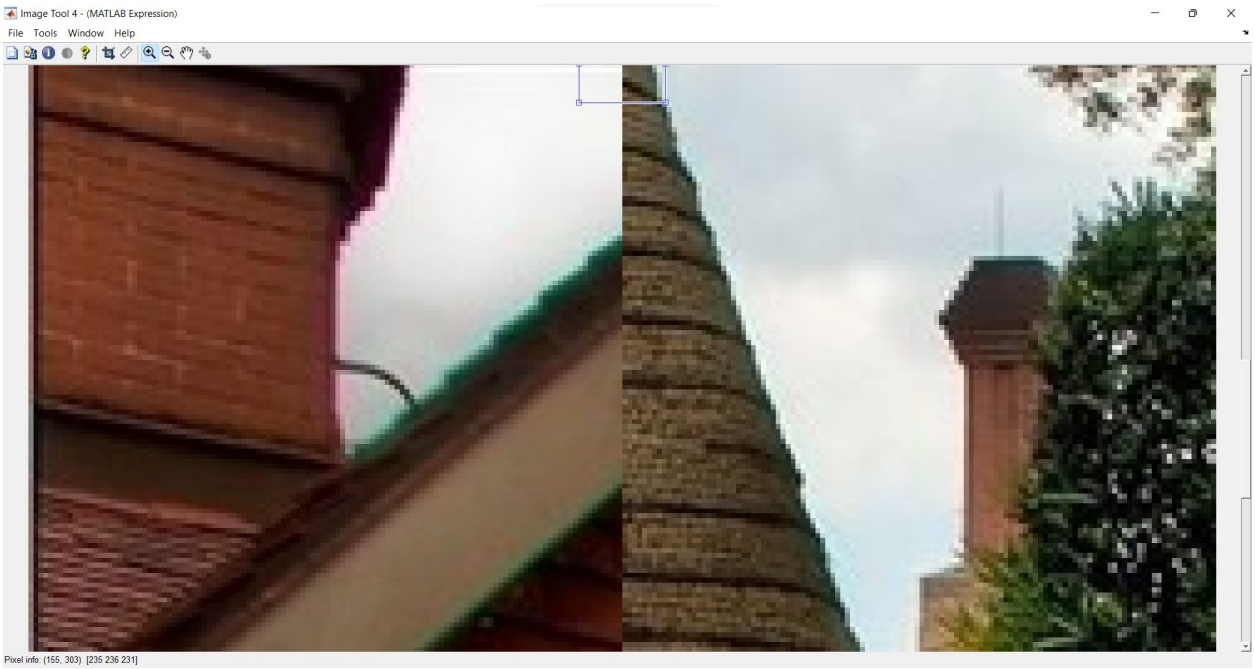
شکل ۸: لبه‌ها (نمای نزدیک)



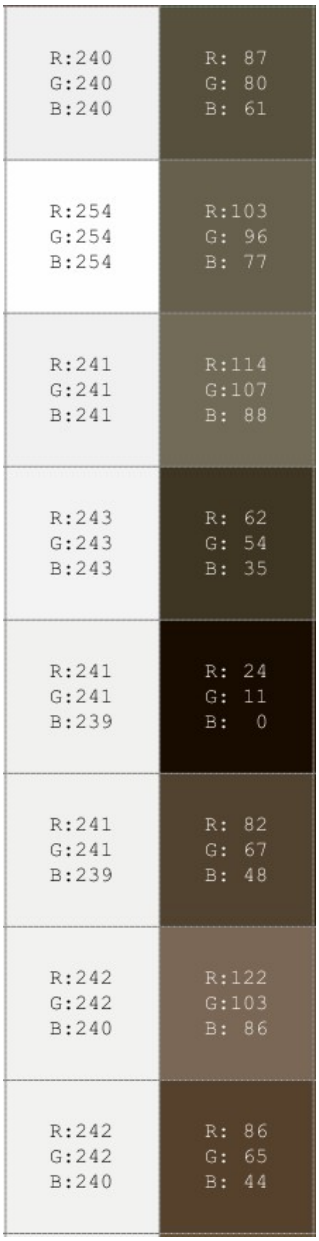
شکل ۹: لبه‌ها (نمای دور)



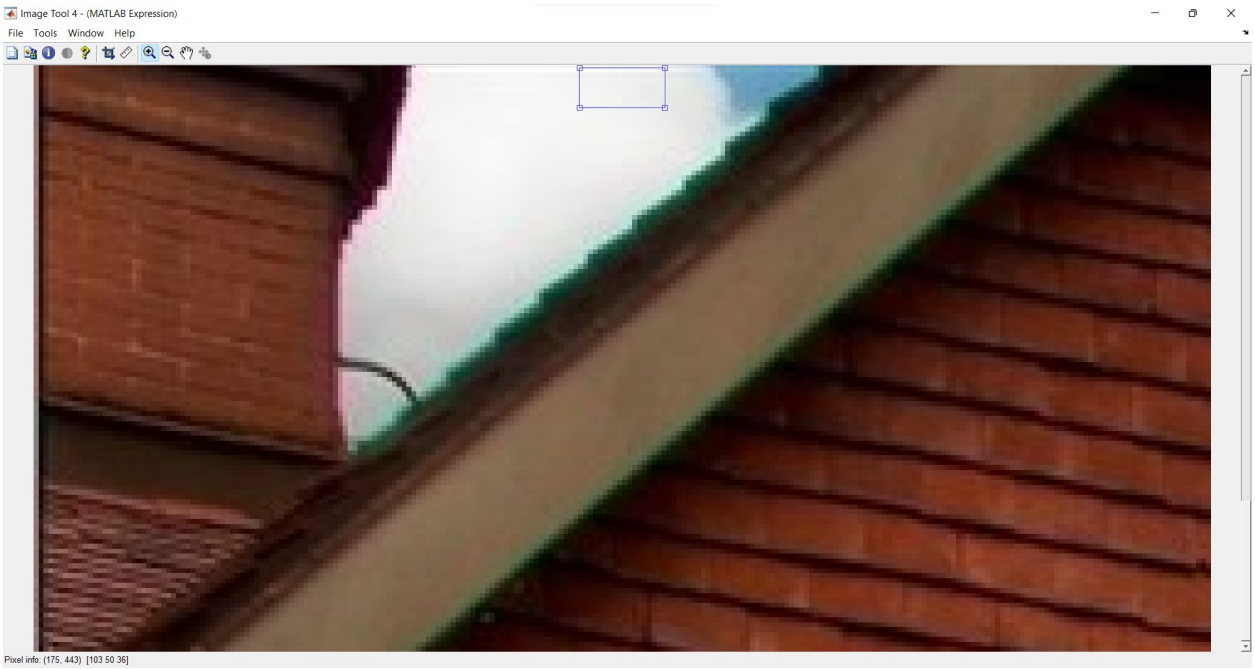
شکل ۱۰: لبه‌ها (نمای نزدیک)



شکل ۱۱: لبه‌ها (نمای دور)



شکل ۱۲: لبه‌ها (نمای نزدیک)



شکل ۱۳: لبه‌ها (نمای دور)

R:240 G:240 B:240	R:237 G:239 B:238
R:254 G:254 B:254	R:250 G:252 B:251
R:241 G:241 B:241	R:238 G:240 B:239
R:243 G:243 B:243	R:240 G:242 B:241
R:241 G:241 B:239	R:240 G:240 B:240
R:241 G:241 B:239	R:241 G:241 B:241
R:242 G:242 B:240	R:240 G:240 B:238
R:242 G:242 B:240	R:239 G:239 B:237

شکل ۱۴: لبه‌ها (نمای نزدیک)

۲.۳ Function

```
function diff = borderDiff(X, Y, direction)
%BORDERDIFF Summary of this function goes here
    X = int64(X);
    Y = int64(Y);
    r = size(X, 1);
    diff = uint64(0);
```

```
    if (direction == 0)
        diff = sum((X(r, :) - Y(1, :)) .^ 2);
    else
        diff = sum((X(:, r) - Y(:, 1)) .^ 2);
    end
end
```

۳.۳ Driver code

۱.۳.۳ الف

```
clc
clear
close all
imtool close all
%%%%%%%%%%%%%%%%%%
temp = imread("images\Q3\Puzzle_1_40\Corner_1_1.tif");
r = size(temp, 1);
row = 1200;
column = 1920;
number_of_pieces = (row / r) * (column / r) - 4;
C11 = imread("images\Q3\Puzzle_1_40\Corner_1_1.tif");
C18 = imread("images\Q3\Puzzle_1_40\Corner_1_8.tif");
C51 = imread("images\Q3\Puzzle_1_40\Corner_5_1.tif");
C58 = imread("images\Q3\Puzzle_1_40\Corner_5_8.tif");
J = uint8(zeros(row, column, 3));
J(1: r, 1: r, :) = C11;
J(1: r, column - r + 1: column, :) = C18;
J(row - r + 1: row, 1: r, :) = C51;
J(row - r + 1: row, column - r + 1: column, :) = C58;

patches = uint8(zeros(number_of_pieces, r, r, 3));
for i = 1: number_of_pieces
    patches(i, :, :, :) = imread(['images\Q3\Puzzle_1_40\' 'Patch_' num2str(i) '.tif']);
end
solution = zeros(row / r, column / r);
for i = 1: r: row
    for j = 1: r: column
```

```
28          if (((i == 1) && (j == 1)) || ((i == row - r + 1) && (j == 1)) || ((i == 1) && (j
        == column - r + 1)) || ((i == row - r + 1) && (j == column - r + 1)))
29              continue;
30          end
31          if (i == 1)
32              base = J(i: i + r - 1, j - r: j - 1, :);
33              values = uint32(zeros(1, number_of_pieces));
34              for k = 1: number_of_pieces
35                  values(k) = borderDiff(rgb2gray(base), rgb2gray(squeeze(patches(k, :, :,
        :))), 1);
36              end
37              [min_value, min_index] = min(values);
38              J(i: i + r - 1, j: j + r - 1, :) = patches(min_index, :, :, :);
39              solution(ceil(i / r), ceil(j / r)) = min_index;
40              imshow(J, []);
41          else
42              base = J(i - r: i - 1, j: j + r - 1, :);
43              values = uint32(zeros(1, number_of_pieces));
44              for k = 1: number_of_pieces
45                  values(k) = borderDiff(rgb2gray(base), rgb2gray(squeeze(patches(k, :, :,
        :))), 0);
46              end
47              if (j > 1)
48                  base = J(i: i + r - 1, j - r: j - 1, :);
49                  for k = 1: number_of_pieces
50                      values(k) = values(k) + borderDiff(rgb2gray(base), rgb2gray(squeeze(
        patches(k, :, :, :))), 1);
51                  end
52              end
53              [min_value, min_index] = min(values);
54              J(i: i + r - 1, j: j + r - 1, :) = patches(min_index, :, :, :);
55              solution(ceil(i / r), ceil(j / r)) = min_index;
56              imshow(J, []);
57          end
58      end
59  end
```

۲.۳.۳ ب

```
clc
clear
close all
imtool close all
%%%%%%%%%%%%%%%%
temp = imread("images\Q3\Puzzle_2_160\Corner_1_1.tif");
r = size(temp, 1);
row = 1200;
column = 1920;
number_of_pieces = (row / r) * (column / r) - 4;
C11 = imread("images\Q3\Puzzle_2_160\Corner_1_1.tif");
C18 = imread("images\Q3\Puzzle_2_160\Corner_1_16.tif");
C51 = imread("images\Q3\Puzzle_2_160\Corner_10_1.tif");
C58 = imread("images\Q3\Puzzle_2_160\Corner_10_16.tif");
J = uint8(zeros(row, column, 3));
J(1: r, 1: r, :) = C11;
J(1: r, column - r + 1: column, :) = C18;
J(row - r + 1: row, 1: r, :) = C51;
J(row - r + 1: row, column - r + 1: column, :) = C58;

patches = uint8(zeros(number_of_pieces, r, r, 3));
for i = 1: number_of_pieces
    patches(i, :, :, :) = imread(['images\Q3\Puzzle_2_160\' 'Patch_' num2str(i) '.tif']);
end
solution = zeros(row / r, column / r);
for i = 1: r: row
    for j = 1: r: column
        if (((i == 1) && (j == 1)) || ((i == row - r + 1) && (j == 1)) || ((i == 1) && (j
     == column - r + 1)) || ((i == row - r + 1) && (j == column - r + 1)))
            continue;
        end
        if (i == 1)
            base = J(i: i + r - 1, j - r: j - 1, :);
            values = uint32(zeros(1, number_of_pieces));
            for k = 1: number_of_pieces
```

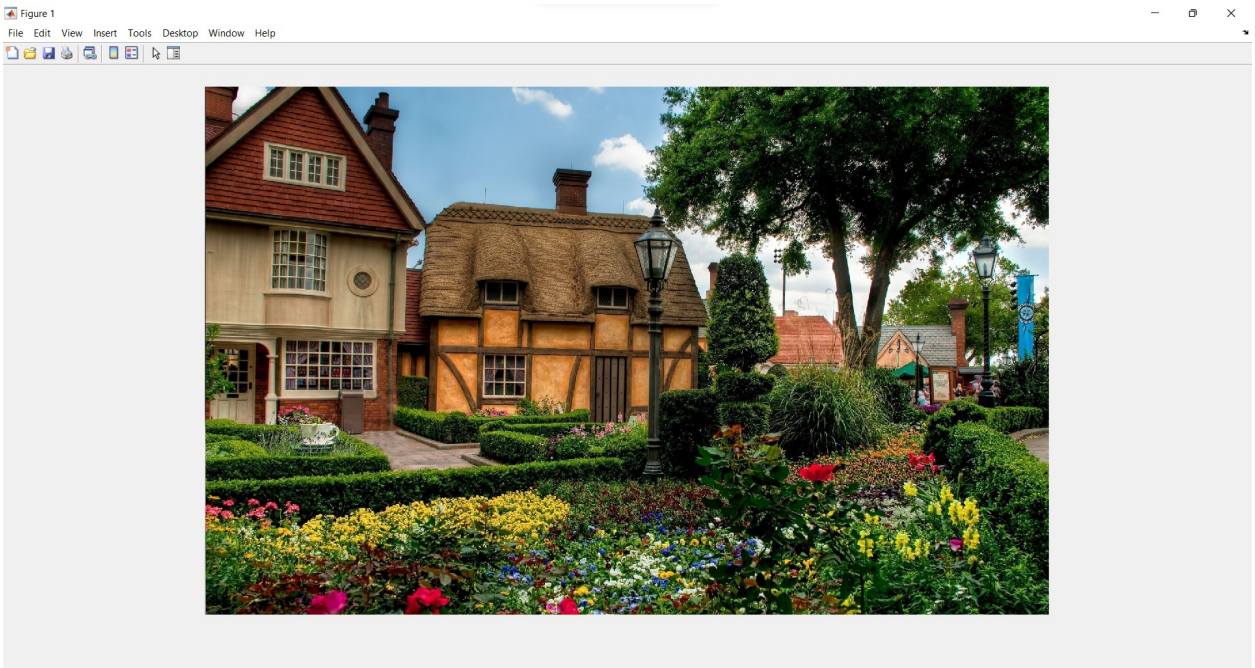
```
35                values(k) = borderDiff(rgb2gray(base), rgb2gray(squeeze(patches(k, :, :,
    :))), 1);
36            end
37            [min_value, min_index] = min(values);
38            J(i: i + r - 1, j: j + r - 1, :) = patches(min_index, :, :, :);
39            solution(ceil(i / r), ceil(j / r)) = min_index;
40            imshow(J, []);
41        else
42            base = J(i - r: i - 1, j: j + r - 1, :);
43            values = uint32(zeros(1, number_of_pieces));
44            for k = 1: number_of_pieces
45                values(k) = borderDiff(rgb2gray(base), rgb2gray(squeeze(patches(k, :, :,
    :))), 0);
46            end
47            if (j > 1)
48                base = J(i: i + r - 1, j - r: j - 1, :);
49                for k = 1: number_of_pieces
50                    values(k) = values(k) + borderDiff(rgb2gray(base), rgb2gray(squeeze(
    patches(k, :, :, :))), 1);
51                end
52            end
53            [min_value, min_index] = min(values);
54            J(i: i + r - 1, j: j + r - 1, :) = patches(min_index, :, :, :);
55            solution(ceil(i / r), ceil(j / r)) = min_index;
56            imshow(J, []);
57        end
58    end
59 end
```

۴.۳ Results

۱.۴.۳ الف

شکل ۱۵

۲.۴.۳ ب



شکل ۱۶

منابع