



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 ۱۴۰۱/۱۴۰۰

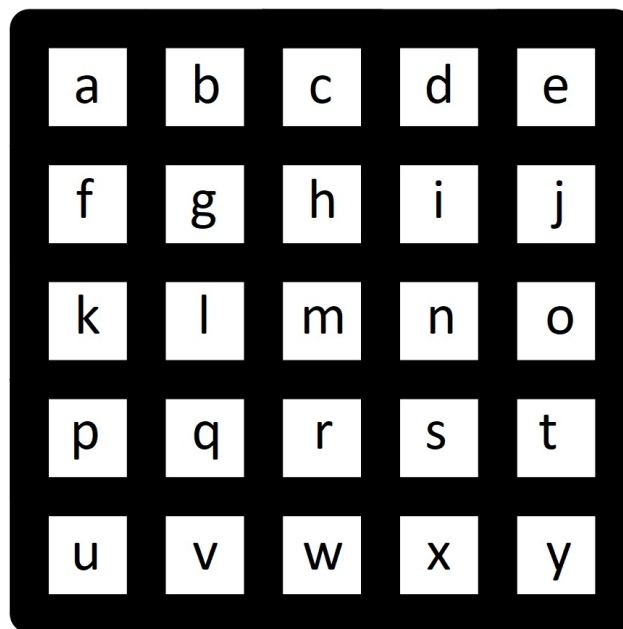
مدرس: دکتر نادر کریمی

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  $\alpha$ ،  $\beta$  و  $\gamma$  (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  $\alpha$ ،  $\beta$  و  $\gamma$  (نسبت ۱:۱:۱ وزن‌ها (متناظر فیلتر Prewitt در جهت عمودی) و نسبت ۱:۲:۱ (متناظر فیلتر Sobel در جهت عمودی) 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

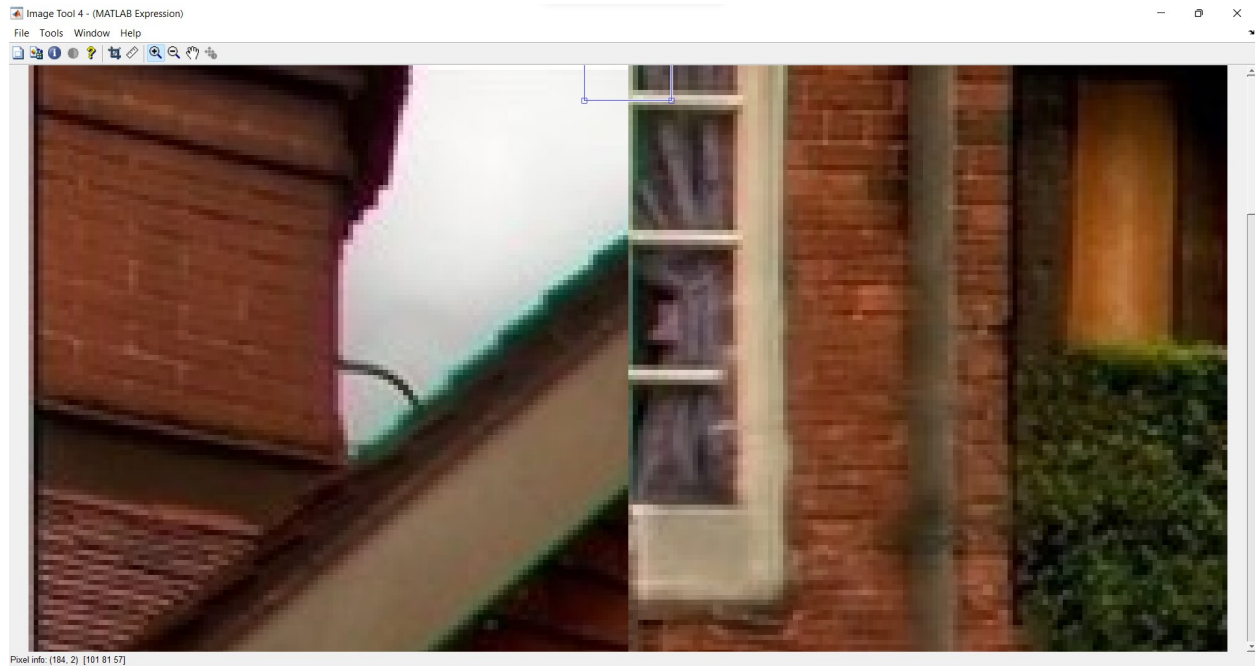
۳

**Algorithm ۱.۳**

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 ۲۱ فاصله اقلیدسی بردار ویژگی‌های تصویر خاکستری گونه‌ی همه‌ی قطعه‌های موجود را با تصویر خاکستری گونه‌ی 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.



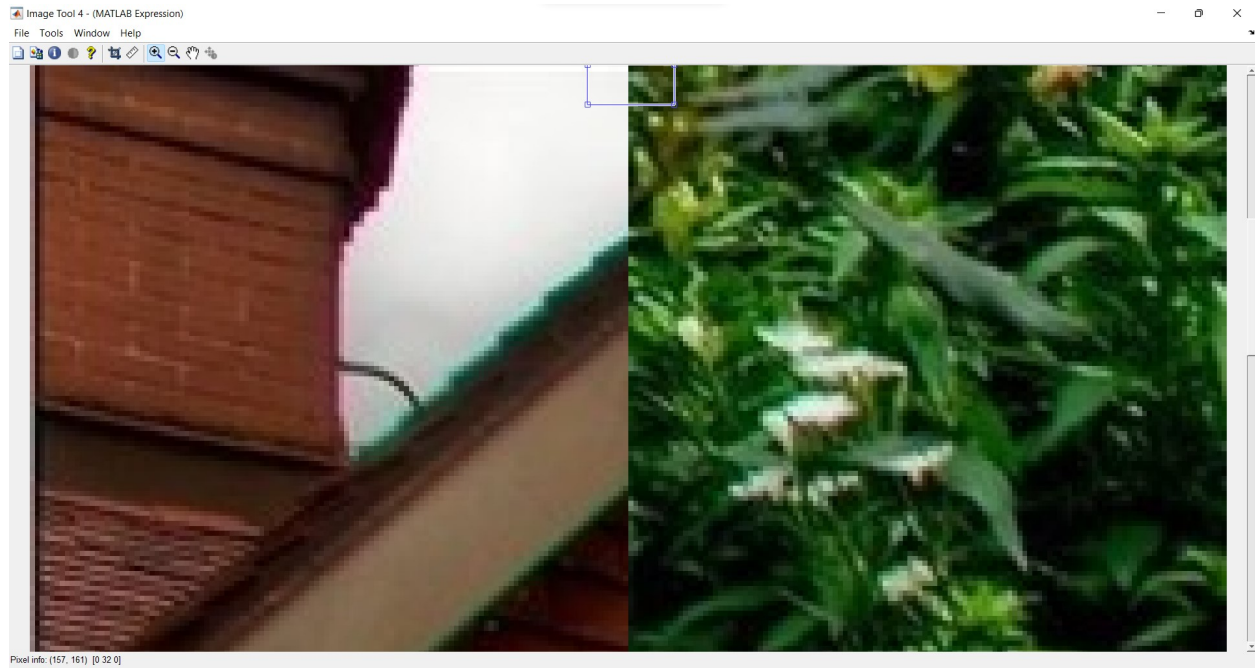
شکل ۲: مقایسه‌ی لبه‌ی راست قطعه‌ی گوشه‌ی چپ بالا با چند قطعه



شکل ۳: 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 ۴: لبه‌ها (نمای نزدیک)

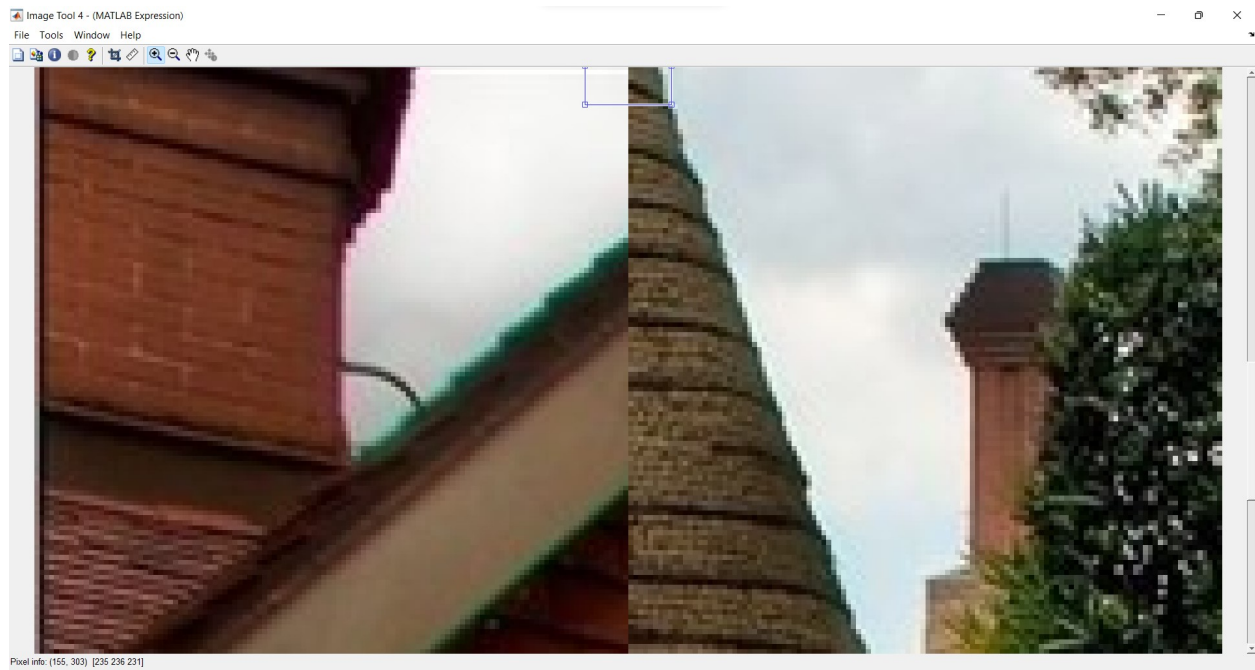


شکل ۵: لبه‌ها (نمای دور)

R:240 G:240 B:240	R: 8 G: 27 B: 0
R:254 G:254 B:254	R: 16 G: 42 B: 7
R:241 G:241 B:241	R: 96 G:126 B: 90
R:243 G:243 B:243	R:116 G:150 B:113
R:241 G:241 B:239	R: 39 G: 75 B: 37
R:241 G:241 B:239	R: 37 G: 74 B: 33
R:242 G:242 B:240	R: 44 G: 82 B: 41
R:242 G:242 B:240	R: 24 G: 60 B: 16

شکل ۶: لبه‌ها (نمای نزدیک)

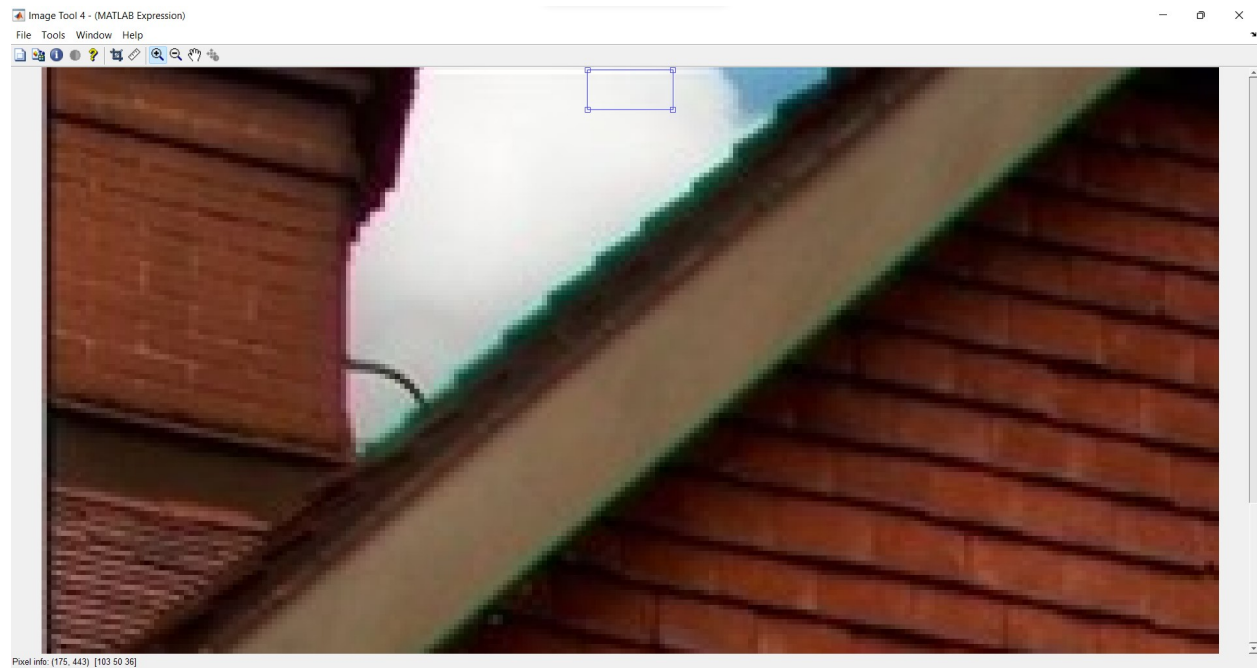




شکل ۷: لبه‌ها (نمای دور)

R:240 G:240 B:240	R: 87 G: 80 B: 61
R:254 G:254 B:254	R:103 G: 96 B: 77
R:241 G:241 B:241	R:114 G:107 B: 88
R:243 G:243 B:243	R: 62 G: 54 B: 35
R:241 G:241 B:239	R: 24 G: 11 B: 0
R:241 G:241 B:239	R: 82 G: 67 B: 48
R:242 G:242 B:240	R:122 G:103 B: 86
R:242 G:242 B:240	R: 86 G: 65 B: 44

شکل ۸: لبه‌ها (نمای نزدیک)



شکل ۹: 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 ۱۰: لبه‌ها (نمای نزدیک)

### ۲.۳ Function

```

1 function diff = borderDiff(X, Y, direction)
2 %BORDERDIFF Summary of this function goes here
3     X = int64(X);
4     Y = int64(Y);
5     r = size(X, 1);
6     diff = uint64(0);

```

```

7     if (direction == 0)
8         diff = sum((X(r, :) - Y(1, :)) .^ 2);
9     else
10        diff = sum((X(:, r) - Y(:, 1)) .^ 2);
11    end
12 end

```

Driver code ۳.۳

الف ۱.۳.۳

```

1  clc
2  clear
3  close all
4  imtool close all
5  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
6  temp = imread("images\Q3\Puzzle_1_40\Corner_1_1.tif");
7  r = size(temp, 1);
8  temp = imread("images\Q3\Puzzle_1_40\Shuffled_Patches.tif");
9  row = size(temp, 1);
10 column = size(temp, 2);
11 number_of_pieces = (row / r) * (column / r) - 4;
12 C11 = imread("images\Q3\Puzzle_1_40\Corner_1_1.tif");
13 C18 = imread("images\Q3\Puzzle_1_40\Corner_1_8.tif");
14 C51 = imread("images\Q3\Puzzle_1_40\Corner_5_1.tif");
15 C58 = imread("images\Q3\Puzzle_1_40\Corner_5_8.tif");
16 J = uint8(zeros(row, column, 3));
17 J(1: r, 1: r, :) = C11;
18 J(1: r, column - r + 1: column, :) = C18;
19 J(row - r + 1: row, 1: r, :) = C51;
20 J(row - r + 1: row, column - r + 1: column, :) = C58;
21
22 patches = uint8(zeros(number_of_pieces, r, r, 3));
23 for i = 1: number_of_pieces
24     patches(i, :, :, :) = imread(['images\Q3\Puzzle_1_40\' 'Patch_' num2str(i) '.tif']);
25 end
26 solution = zeros(row / r, column / r);
27 for i = 1: r: row

```

```

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

```

۲.۳.۳ ب

```

60 end

1  clc
2  clear
3  close all
4  imtool close all
5  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
6  temp = imread("images\Q3\Puzzle_2_160\Corner_1_1.tif");
7  r = size(temp, 1);
8  temp = imread("images\Q3\Puzzle_2_160\Shuffled_Patches.tif");
9  row = size(temp, 1);
10 column = size(temp, 2);
11 number_of_pieces = (row / r) * (column / r) - 4;
12 C11 = imread("images\Q3\Puzzle_2_160\Corner_1_1.tif");
13 C18 = imread("images\Q3\Puzzle_2_160\Corner_1_16.tif");
14 C51 = imread("images\Q3\Puzzle_2_160\Corner_10_1.tif");
15 C58 = imread("images\Q3\Puzzle_2_160\Corner_10_16.tif");
16 J = uint8(zeros(row, column, 3));
17 J(1: r, 1: r, :) = C11;
18 J(1: r, column - r + 1: column, :) = C18;
19 J(row - r + 1: row, 1: r, :) = C51;
20 J(row - r + 1: row, column - r + 1: column, :) = C58;
21
22 patches = uint8(zeros(number_of_pieces, r, r, 3));
23 for i = 1: number_of_pieces
24     patches(i, :, :, :) = imread(['images\Q3\Puzzle_2_160\' 'Patch_' num2str(i) '.tif']);
25 end
26 solution = zeros(row / r, column / r);
27 for i = 1: r: row
28     for j = 1: r: column
29         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)))
30             continue;
31         end
32         if (i == 1)

```

```

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

```



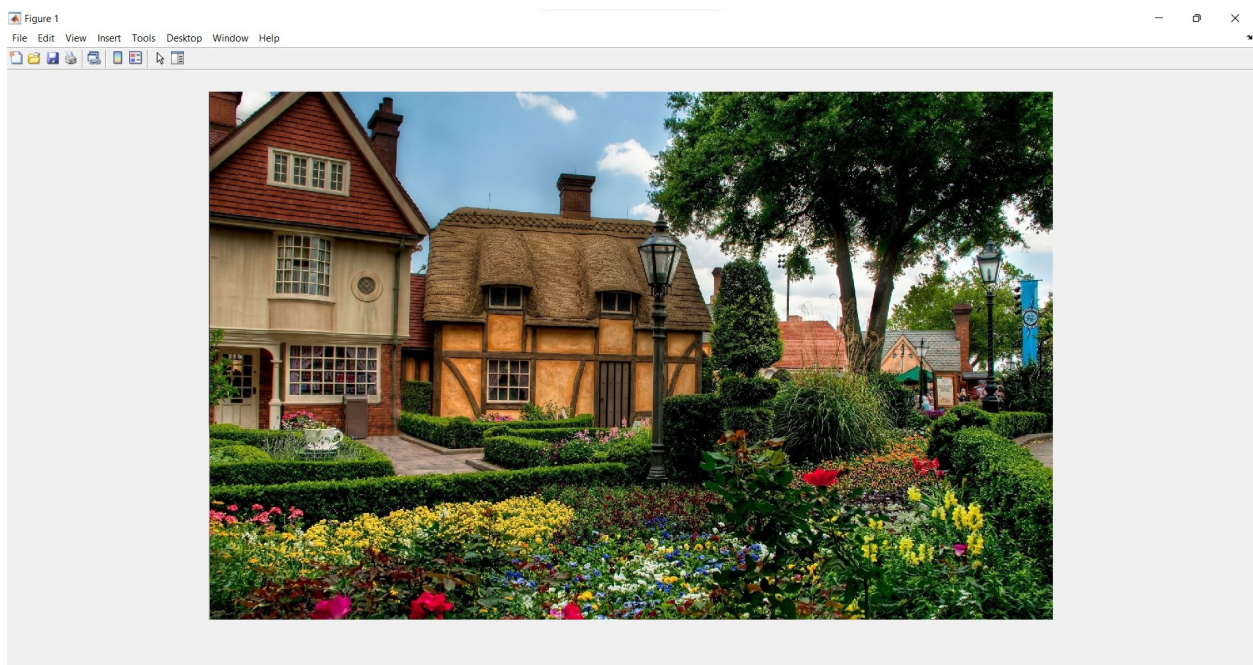
Results ۴.۳

الف ۱.۴.۳



شکل ۱۱

ب ۲.۴.۳



شکل ۱۲

## منابع