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

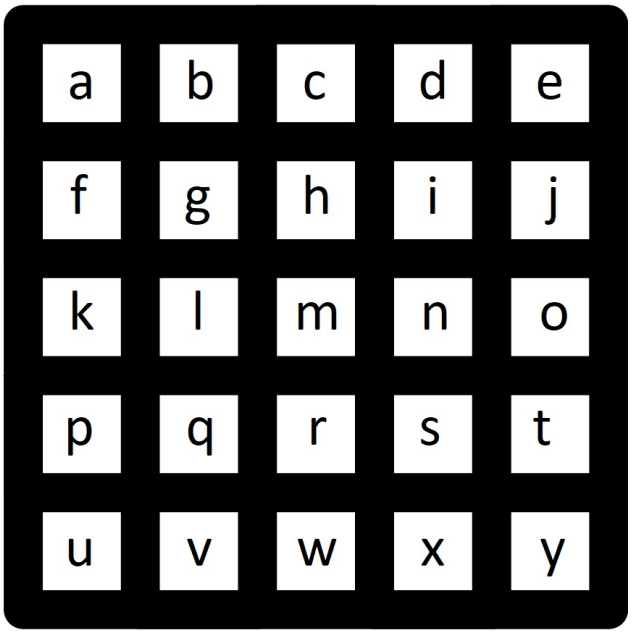
دستیاران آموزشی: بهنام ساعدی - محمدرضا مزروعی

۱

Prewitt ۱.۱

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 مشتق‌گیری عمودی معادل با اعمال فیلتر Prewitt در جهت عمودی خواهد بود.



شکل ۱: مقایسه‌ی لبه‌ی راست قطعه‌ی گوشه‌ی چپ بالا با چند قطعه

Sobel ۲.۱

۲

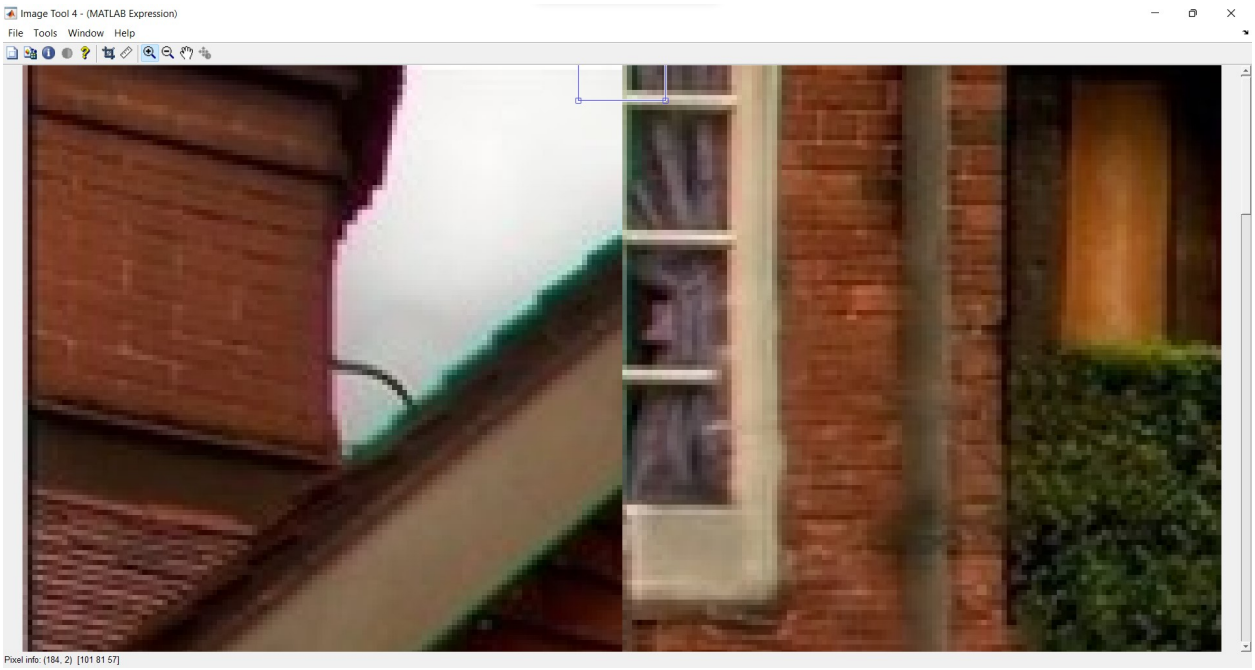
۳

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 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.

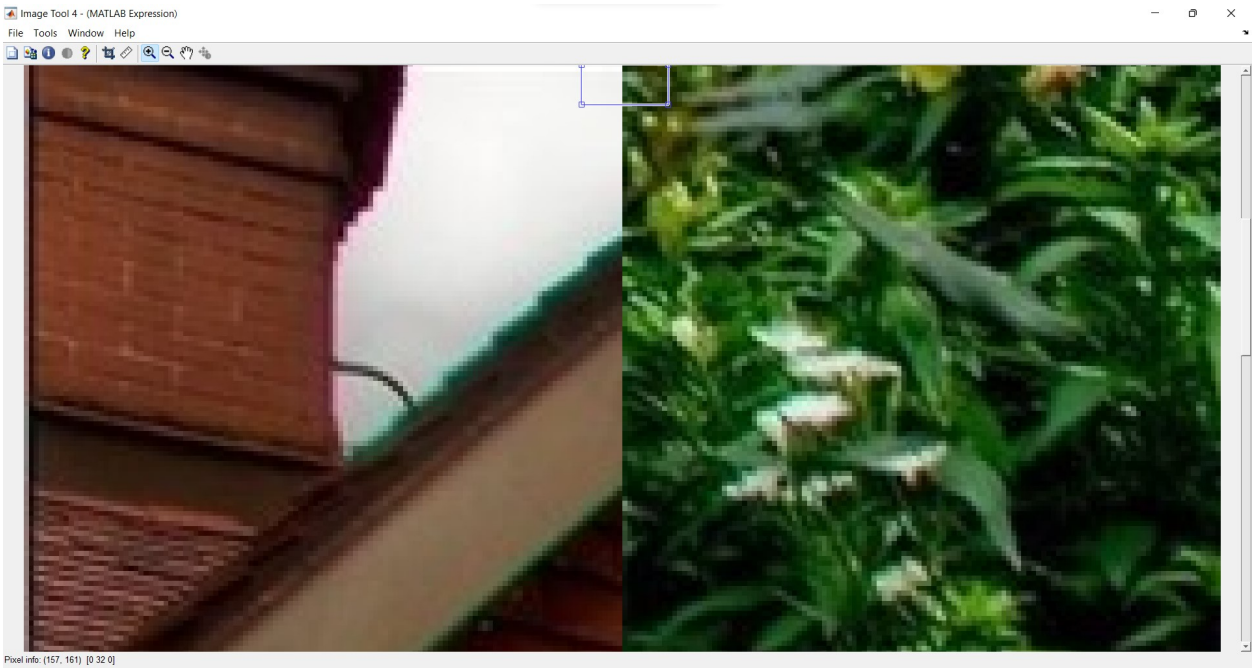
شکل ۲: مقایسه‌ی لبه‌ی راست قطعه‌ی گوشه‌ی چپ بالا با چند قطعه



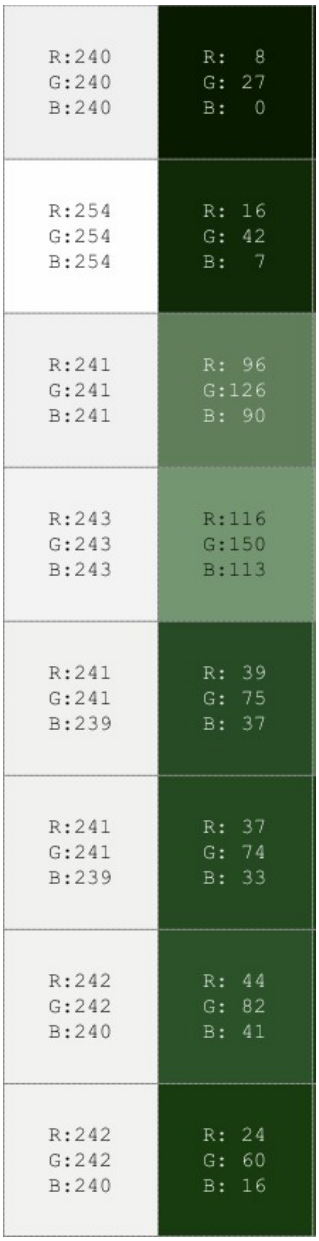
شکل ۳: لبه‌ها (نمای دور)

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

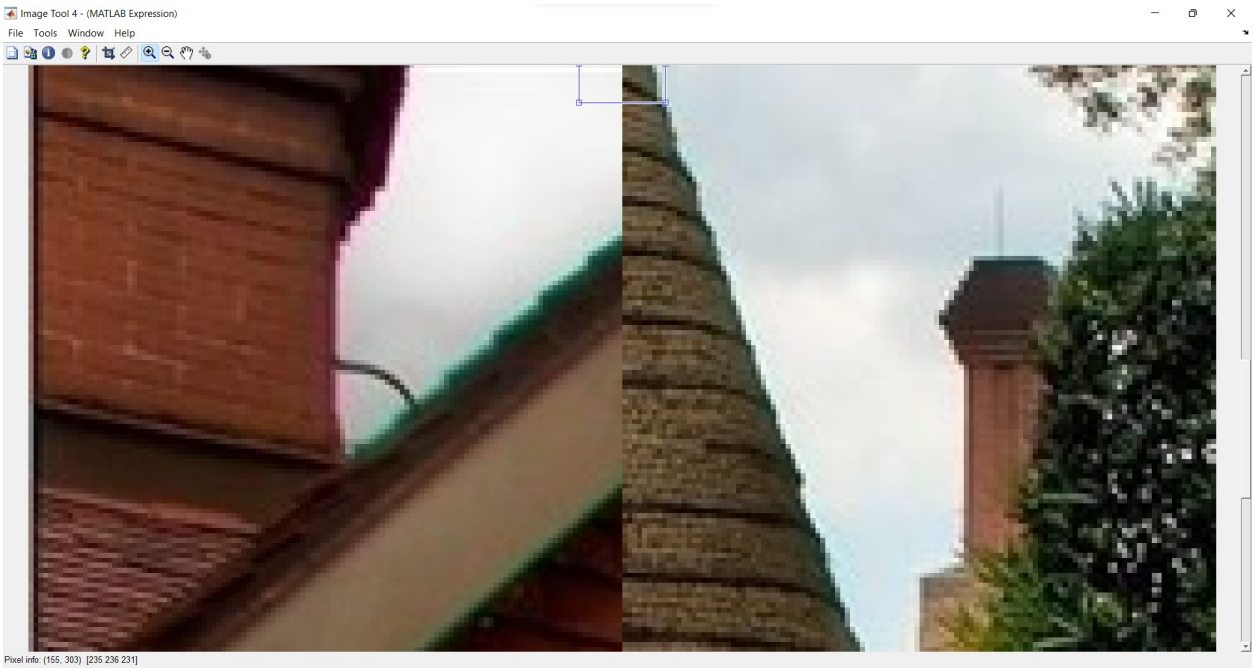
شکل ۴: لبه‌ها (نمای نزدیک)



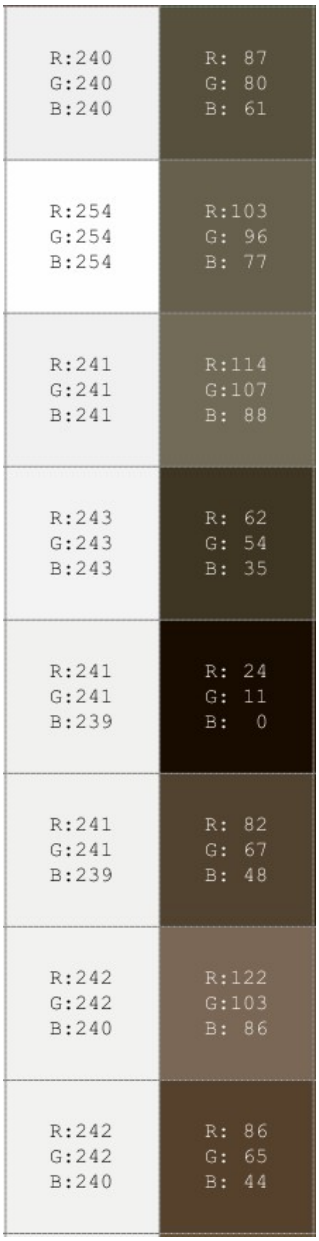
شکل ۵: لبه‌ها (نمای دور)



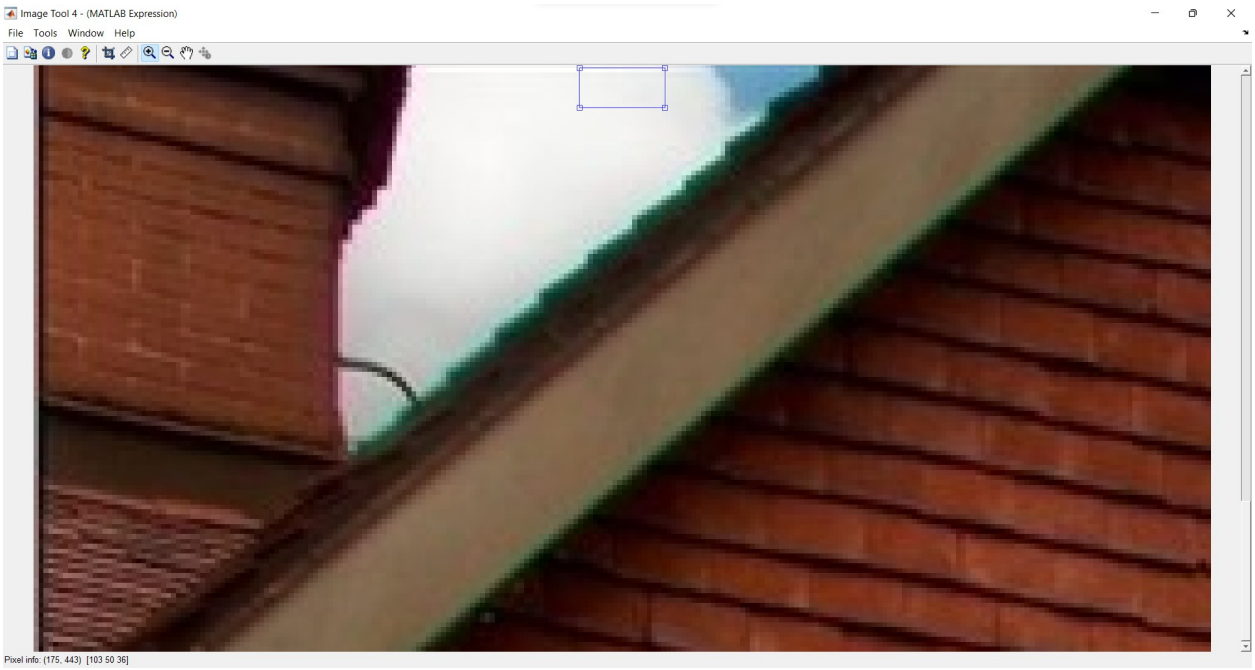
شکل ۶: لبه‌ها (نمای نزدیک)



شکل ۷: لبه‌ها (نمای دور)



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

```
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);
temp = imread("images\Q3\Puzzle_1_40\Shuffled_Patches.tif");
row = size(temp, 1);
column = size(temp, 2);
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
        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
                values(k) = borderDiff(rgb2gray(base), rgb2gray(squeeze(patches(k, :, :,
:))), 1);
            end
            [min_value, min_index] = min(values);
            J(i: i + r - 1, j: j + r - 1, :) = patches(min_index, :, :, :);
            solution(ceil(i / r), ceil(j / r)) = min_index;
            imshow(J, []);
        else
            base = J(i - r: i - 1, j: j + r - 1, :);
            values = uint32(zeros(1, number_of_pieces));
            for k = 1: number_of_pieces
                values(k) = borderDiff(rgb2gray(base), rgb2gray(squeeze(patches(k, :, :,
:))), 0);
            end
            if (j > 1)
                base = J(i: i + r - 1, j - r: j - 1, :);
                for k = 1: number_of_pieces
                    values(k) = values(k) + borderDiff(rgb2gray(base), rgb2gray(squeeze(
patches(k, :, :, :))), 1);
                end
            end
            [min_value, min_index] = min(values);
            J(i: i + r - 1, j: j + r - 1, :) = patches(min_index, :, :, :);
            solution(ceil(i / r), ceil(j / r)) = min_index;
            imshow(J, []);
        end
    end
```

```
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);
temp = imread("images\Q3\Puzzle_2_160\Shuffled_Patches.tif");
row = size(temp, 1);
column = size(temp, 2);
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)
```

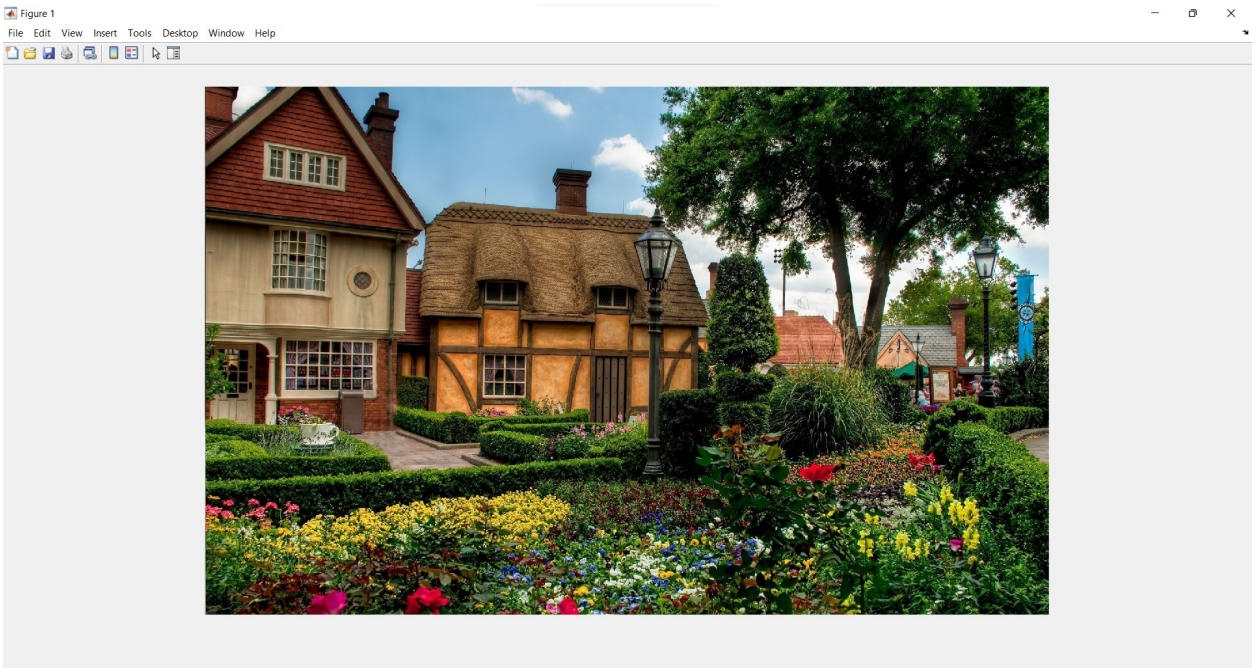
```
            base = J(i: i + r - 1, j - r: j - 1, :);
            values = uint32(zeros(1, number_of_pieces));
            for k = 1: number_of_pieces
                values(k) = borderDiff(rgb2gray(base), rgb2gray(squeeze(patches(k, :, :,
    :))), 1);
            end
            [min_value, min_index] = min(values);
            J(i: i + r - 1, j: j + r - 1, :) = patches(min_index, :, :, :);
            solution(ceil(i / r), ceil(j / r)) = min_index;
            imshow(J, []);
        else
            base = J(i - r: i - 1, j: j + r - 1, :);
            values = uint32(zeros(1, number_of_pieces));
            for k = 1: number_of_pieces
                values(k) = borderDiff(rgb2gray(base), rgb2gray(squeeze(patches(k, :, :,
    :))), 0);
            end
            if (j > 1)
                base = J(i: i + r - 1, j - r: j - 1, :);
                for k = 1: number_of_pieces
                    values(k) = values(k) + borderDiff(rgb2gray(base), rgb2gray(squeeze(
    patches(k, :, :, :))), 1);
                end
            end
            [min_value, min_index] = min(values);
            J(i: i + r - 1, j: j + r - 1, :) = patches(min_index, :, :, :);
            solution(ceil(i / r), ceil(j / r)) = min_index;
            imshow(J, []);
        end
    end
end
```

۴.۳ Results

۱.۴.۳ الف

شکل ۱۱

۲.۴.۳ ب



شکل ۱۲

منابع