**impixel -** Pixel color values

**Syntax**

P = impixel(I)

P = impixel(X,map)

P = impixel(RGB)

P = impixel(I,c,r)

P = impixel(X,map,c,r)

P = impixel(RGB,c,r)

[c,r,P] = impixel(...)

P = impixel(x,y,I,xi,yi)

P = impixel(x,y,X,map,xi,yi)

P = impixel(x,y,RGB,xi,yi)

[xi,yi,P] = impixel(x,y,...)

**Description**

impixel returns the red, green, and blue color values of specified image pixels. In the syntax below, impixel displays the input image and waits for you to specify the pixels with the mouse.

P = impixel(I)

P = impixel(X,map)

P = impixel(RGB)

If you omit the input arguments, impixel operates on the image in the current axes.

Use normal button clicks to select pixels. Press **Backspace** or **Delete** to remove the previously selected pixel. A shift-click, right-click, or double-click adds a final pixel and ends the selection; pressing **Return** finishes the selection without adding a pixel.

When you finish selecting pixels, impixel returns an m-by-3 matrix of RGB values in the supplied output argument. If you do not supply an output argument, impixel returns the matrix in ans.

You can also specify the pixels noninteractively, using these syntax.

P = impixel(I,c,r)

P = impixel(X,map,c,r)

P = impixel(RGB,c,r)

r and c are equal-length vectors specifying the coordinates of the pixels whose RGB values are returned in P. The kth row of P contains the RGB values for the pixel (r(k),c(k)).

If you supply three output arguments, impixel returns the coordinates of the selected pixels. For example,

[c,r,P] = impixel(...)

To specify a nondefault spatial coordinate system for the input image, use these syntax.

P = impixel(x,y,I,xi,yi)

P = impixel(x,y,X,map,xi,yi)

P = impixel(x,y,RGB,xi,yi)

x and y are two-element vectors specifying the image XData and YData. xi and yi are equal-length vectors specifying the spatial coordinates of the pixels whose RGB values are returned in P. If you supply three output arguments, impixel returns the coordinates of the selected pixels.

[xi,yi,P] = impixel(x,y,...)

**Class Support**

The input image can be of class uint8, uint16, int16, single, double, or logical. All other inputs are of class double.

If the input is double, the output P is double. For all other input classes the output is single. The rest of the outputs are double.

**Remarks**

impixel works with indexed, intensity, and RGB images. impixel always returns pixel values as RGB triplets, regardless of the image type:

* For an RGB image, impixel returns the actual data for the pixel. The values are either uint8 integers or double floating-point numbers, depending on the class of the image array.
* For an indexed image, impixel returns the RGB triplet stored in the row of the colormap that the pixel value points to. The values are double floating-point numbers.
* For an intensity image, impixel returns the intensity value as an RGB triplet, where R=G=B. The values are either uint8 integers or double floating-point numbers, depending on the class of the image array.

**Examples**

RGB = imread('peppers.png');

c = [12 146 410];

r = [104 156 129];

pixels = impixel(RGB,c,r)

pixels =

62 34 63

166 54 60

59 28 47

**improfile -** Pixel-value cross-sections along line segments

**Syntax**

c = improfile

c = improfile(n)

c = improfile(I,xi,yi)

c = improfile(I,xi,yi,n)

[cx,cy,c] = improfile(...)

[cx,cy,c,xi,yi] = improfile(...)

[...] = improfile(x,y,I,xi,yi)

[...] = improfile(x,y,I,xi,yi,n)

[...] = improfile(...,method)

**Description**

improfile computes the intensity values along a line or a multiline path in an image. improfile selects equally spaced points along the path you specify, and then uses interpolation to find the intensity value for each point. improfile works with grayscale images and RGB images.

If you call improfile with one of these syntax, it operates interactively on the image in the current axes.

c = improfile

c = improfile(n)

n specifies the number of points to compute the intensity value for. If you do not provide this argument, improfile chooses a value for n, roughly equal to the number of pixels the path traverses.

You specify the line or path using the mouse, by clicking points in the image. Press **Backspace** or **Delete** to remove the previously selected point. A shift-click, right-click, or double-click adds a final point and ends the selection; pressing **Return** finishes the selection without adding a point. When you finish selecting points, improfile returns the interpolated data values in c. c is an n-by-1 vector if the input is a grayscale intensity image, or an n-by-1-by-3 array if the input is an RGB image.

If you omit the output argument, improfile displays a plot of the computed intensity values. If the specified path consists of a single line segment, improfile creates a two-dimensional plot of intensity values versus the distance along the line segment; if the path consists of two or more line segments, improfile creates a three-dimensional plot of the intensity values versus their *x*- and *y*-coordinates.

You can also specify the path non-interactively, using these syntax.

c = improfile(I,xi,yi)

c = improfile(I,xi,yi,n)

xi and yi are equal-length vectors specifying the spatial coordinates of the endpoints of the line segments.

You can use these syntax to return additional information.

[cx,cy,c] = improfile(...)

[cx,cy,c,xi,yi] = improfile(...)

cx and cy are vectors of length n, containing the spatial coordinates of the points at which the intensity values are computed.

To specify a non-default spatial coordinate system for the input image, use these syntax.

[...] = improfile(x,y,I,xi,yi)

[...] = improfile(x,y,I,xi,yi,n)

x and y are two-element vectors specifying the image XData and YData.

[...] = improfile(...,method) uses the specified interpolation method. method is a string that can have one of these values. The default value is enclosed in braces ({}).

| **Value** | **Description** |
| --- | --- |
| {'nearest'} | Nearest-neighbor interpolation |
| 'bilinear' | Bilinear interpolation |
| 'bicubic' | Bicubic interpolation |

**Class Support**

The input image can be uint8, uint16, int16, single, double, or logical. All other inputs and outputs must be double.

**Examples**

I = imread('liftingbody.png');

x = [19 427 416 77];

y = [96 462 37 33];

improfile(I,x,y),grid on;