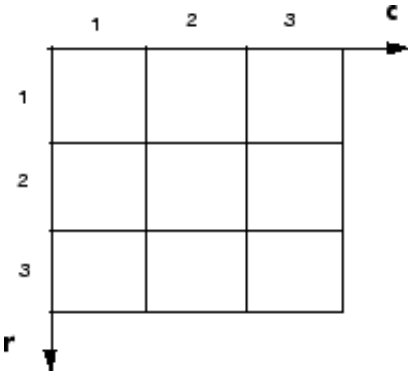
Expressing Image Locations

Pixel Indices

Often, the most convenient method for expressing locations in an image is to use pixel indices. The image is treated as a grid of discrete elements, ordered from top to bottom and left to right, as illustrated by the following figure.

Pixel Indices



For pixel indices, the row increases downward, while the column increases to the right. Pixel indices are integer values, and range from 1 to the length of the row or column.

There is a one-to-one correspondence between pixel indices and subscripts for the first two matrix dimensions in MATLAB. For example, the data for the pixel in the fifth row, second column is stored in the matrix element (5,2). You use normal MATLAB matrix subscripting to access values of individual pixels. For example, the MATLAB code

```
I(2,15)
```

returns the value of the pixel at row 2, column 15 of the image `I`. Similarly, the MATLAB code

```
RGB(2,15,:)
```

returns the `R`, `G`, `B` values of the pixel at row 2, column 15 of the image `RGB`.

The correspondence between pixel indices and subscripts for the first two matrix dimensions in MATLAB makes the relationship between an image's data matrix and the way the image is displayed easy to understand.

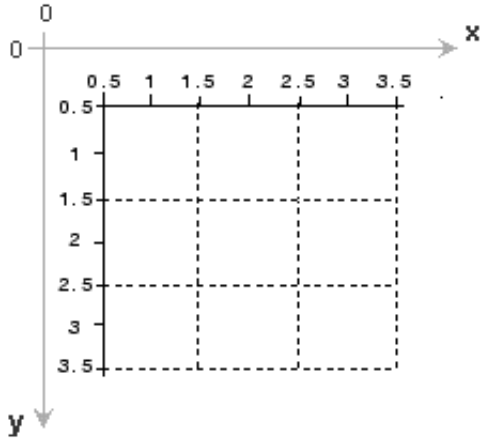
Spatial Coordinates

Another method for expressing locations in an image is to use a system of continuously varying coordinates rather than discrete indices. This lets you consider an image as covering a square patch, for example. In a *spatial coordinate system* like this, locations in an image are positions on a plane, and they are described in terms of *x* and *y* (not row and column as in the pixel indexing system). From this Cartesian perspective, an (*x*,*y*) location such as (3.2,5.3) is meaningful, and is distinct from pixel (5,3).

Intrinsic Coordinates

By default, the toolbox uses a spatial coordinate system for an image that corresponds to the image's pixel indices. It's called the intrinsic coordinate system and is illustrated in the following figure. Notice that *y* increases downward, because this orientation is consistent with the way in which digital images are typically viewed.

Intrinsic Coordinate System



The intrinsic coordinates (*x*,*y*) of the center point of any pixel are identical to the column and row indices for that pixel. For example, the center point of the pixel in row 5, column 3 has spatial coordinates *x* = 3.0, *y* = 5.0. This correspondence simplifies many toolbox functions considerably. Be aware, however, that the order of coordinate specification (3.0,5.0) is reversed in intrinsic coordinates relative to pixel indices (5,3).

Several functions primarily work with spatial coordinates rather than pixel indices, but as long as you are using the default spatial coordinate system (intrinsic coordinates), you can specify locations in terms of their columns (*x*) and rows (*y*).

When looking at the intrinsic coordinate system, note that the upper left corner of the image is located at (0.5,0.5), not at (0,0), and the lower right corner of the image is located at (`numCols` + 0.5, `numRows` + 0.5), where `numCols` and `numRows` are the number of rows and columns in the image. In contrast, the upper left pixel is pixel (1,1) and the lower right pixel is pixel (`numRows`, `numCols`). The center of the upper left pixel is (1.0, 1.0) and the center of the lower right pixel is (`numCols`, `numRows`). In fact, the center coordinates of every pixel are integer valued. The center of the pixel with indices (*r*, *c*) — where *r* and *c* are integers by definition — falls at the point *x* = *c*, *y* = *r* in the intrinsic coordinate system.

World Coordinates

In some situations, you might want to use a world coordinate system (also called a nondefault spatial coordinate system). For example, you could shift the origin by specifying that the upper left corner of an image is the point (19.0,7.5), rather than (0.5,0.5). Or, you might want to specify a coordinate system in which every pixel covers a 5-by-5 meter patch on the ground.

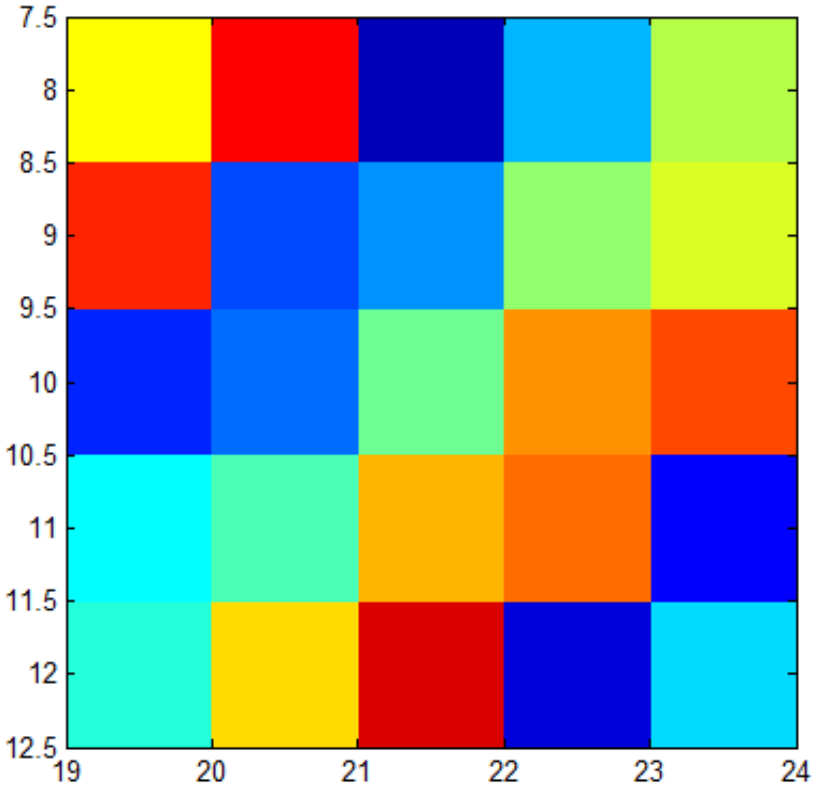
One way to define a world coordinate system for an image is to specify the `XData` and `YData` image properties for the image. The `XData` and `YData` image properties are two-element vectors that control the range of coordinates spanned by the image. When you do this, the MATLAB axes coordinates become identical to the world (nondefault) coordinates. If you do not specify `XData` and `YData`, the axes coordinates are identical to the intrinsic coordinates of the image. By default, for an image `A`, `XData` is `[1 size(A,2)]`, and `YData` is `[1 size(A,1)]`. With this default, the world coordinate system and intrinsic coordinate system coincide perfectly.

For example, if `A` is a 100 row by 200 column image, the default `XData` is [1 200], and the default `YData` is [1 100]. The values in these vectors are actually the coordinates for the center points of the first and last pixels (not the pixel edges), so the actual coordinate range spanned is slightly larger. For instance, if `XData` is [1 200], the interval in `X` spanned by the image is [0.5 200.5].

It's also possible to set `XData` or `YData` such that the *x*-axis or *y*-axis is reversed. You'd do this by placing the larger value first. (For example, set the `YData` to [1000 1].) This is a common technique to use with geospatial data.

These commands display an image using nondefault `XData` and `YData`.

```
A = magic(5);
x = [19.5 23.5];
y = [8.0 12.0];
image(A,'XData',x,'YData',y), axis image, colormap(jet(25))
```



Specifying Coordinate Information

To specify a world (nondefault spatial) coordinate system for an image, use the spatial referencing objects `imref2d` and `imref3d`. Spatial referencing objects let you define the location of the image in a world coordinate system and specify the image resolution, including nonsquare pixel shapes. These objects also support methods for converting between the world, intrinsic, and subscript coordinate systems. Several toolbox functions accept or return spatial referencing objects: `imwarp`, `imshow`, `imshowpair`, `imfuse`, `imregtform`, and `imregister`.

This example creates a spatial referencing object associated with a 2-by-2 image where the world extent is 4 units/pixel in the **x** direction and 2 units/pixel in the y direction. The example creates the object, specifying the pixels dimensions as arguments but does not specify world limits in the **x** and **y** directions. You could specify other information when creating an object, see `imref2d` for more information.

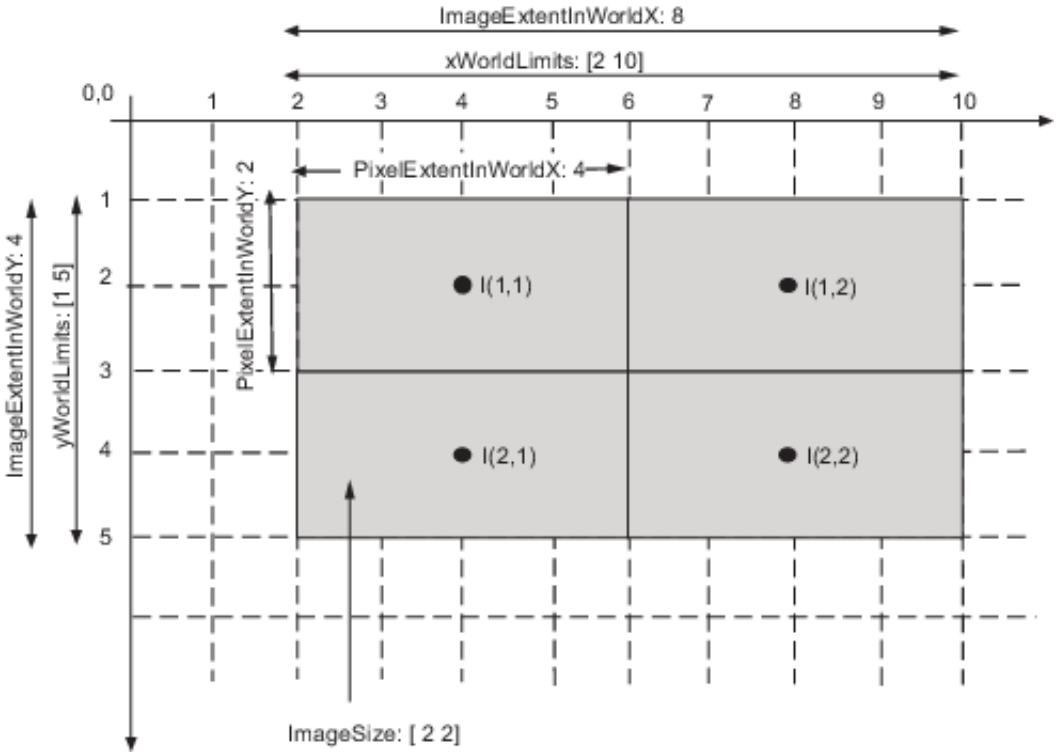
```
I = [1  2; 3 4]
R = imref2d(size(I),4,2)
```

```
R =

  imref2d with properties:

          XWorldLimits: [2 10]
          YWorldLimits: [1 5]
             ImageSize: [2 2]
    PixelExtentInWorldX: 4
    PixelExtentInWorldY: 2
    ImageExtentInWorldX: 8
    ImageExtentInWorldY: 4
       XIntrinsicLimits: [0.5000 2.5000]
       YIntrinsicLimits: [0.5000 2.5000]
```

The `imref2d` object contains information about the image, some of it provided by you and some of it derived by the object. The following table provides descriptions of spatial referencing object fields.

Field	**Description**
`XWorldLimits`	Upper and lower bounds along the *X* dimension in world coordinates (nondefault spatial coordinates)
`YWorldLimits`	Upper and lower bounds along the *Y* dimension in world coordinates (nondefault spatial coordinates)
`ImageSize`	Size of the image, returned by the `size` function.
`PixelExtentInWorldX`	Size of pixel along the *X* dimension
`PixelExtentInWorldY`	Size of pixel along the *Y* dimension
`ImageExtentInWorldX`	Size of image along the *X* dimension
`ImageExtentInWorldY`	Size of image along the *Y* dimension
`XIntrinsicLimits`	Upper and lower bounds along *X* dimension in intrinsic coordinates (default spatial coordinates)
`YIntrinsicLimits`	Upper and lower bounds along *Y* dimension in intrinsic coordinates (default spatial coordinates).

The following figure illustrates how these properties map to elements of an image.



You can also use the `XData` and `YData` properties to define a world (nondefault spatial) coordinate system. Several toolbox functions accept this data as arguments and return coordinates in the world coordinate system: `bwselect`, `imcrop`, `impixel`, `roipoly`, and `imtransform`.