This document summarizes some MATLAB software for analyzing intensity changes in 1D and 2D images. This software is available in M-Files stored in the following subdirectory on the CS file server: /home/cs332/download/edges. The following notes provide the name of each M-File, the format for calling each function, and brief comments describing the function. Keep in mind that the names of the input and output variables specified in actual function calls will change depending on the context.

Analyzing 1D intensity profiles

`gauss1D.m`

```
g = gauss1D(sigma);
gauss1D returns a 1D matrix of samples of a Gaussian function. The size of the matrix is 8*sigma+1 and the Gaussian function is computed for values of x from -4*sigma to +4*sigma.
```

`d2g.m`

```
dg = d2g(sigma);
d2g returns a 1D matrix of samples of the second derivative of a Gaussian function. The size of the matrix is 8*sigma+1 and the function is computed for values of x from -4*sigma to +4*sigma.
```

`conv1D.m`

```
conv = conv1D(profile, operator);
conv1D convolves the input 1D intensity profile with the input 1D convolution operator and returns a 1D matrix of the same size that contains the convolution result. No convolution values are computed for regions at the beginning and end of the profile that are half the operator size.
```

`zeros1D.m`

```
zc = zeros1D(conv);
zeros1D returns a 1D matrix of the zero-crossings of the input 1D matrix of convolution results. The derivative (dx) of the convolution is stored at each zero-crossing location.
```

`plot4.m`

```
plot4(profile, smooth, conv, zcs);
plot4 is a special-purpose plotting function that creates a single figure with graphs of the four 1D input matrices that are assumed to correspond to an intensity profile, the result of smoothing the profile with a Gaussian function, the result of convolution with a second derivative of a Gaussian, and the zero-crossings.
```

1D and 2D image processing

`getRange.m`

```
[minv maxv] = getRange(image, border);
getRange returns the minimum and maximum values in a central region of an input 1D or 2D image that excludes a border around the outer edge of the image.
```
Analyzing 2D images

**gauss2D.m**

```matlab
g = gauss2D(sigma);
```

gauss2D returns a 2D matrix of samples of the 2D Gaussian function. The size of the matrix is 
\((8*sigma+1) \times (8*sigma+1)\) and the Gaussian function is computed for values of \(x\) and \(y\) that range from \(-4*sigma\) to \(+4*sigma\).

**laplacian.m**

```matlab
lap = laplacian(w);
```

laplacian returns a 2D matrix of samples of a Laplacian-of-Gaussian function with a diameter of \(w\) for the central positive region. The matrix is of size \((4*w+1) \times (4*w+1)\) and the function is computed for values of \(x\) and \(y\) that range from \(-2*w\) to \(+2*w\).

**conv2D.m**

```matlab
conv = conv2D(image, operator);
```

conv2D convolves the input 2D image with the input 2D convolution operator and returns a 2D matrix of the same size as the image, containing the convolution result. No convolution values are computed for a border around the outer edge of the image that is half the operator size.

**zeros2D.m**

```matlab
zc = zeros2D(conv);
```

zeros2D returns a 2D matrix of the slopes of the zero-crossings of the input 2D matrix of convolution results. A value of 0 is placed at locations that do not correspond to zero-crossings.

**zcMap.m**

```matlab
zc = zcMap(zeros2D, threshold);
```

zcMap returns a 2D 8-bit matrix that contains the value 255 at the location of each zero-crossing in the input zeros2D matrix, and 0 elsewhere. The input threshold is a fraction between 0.0-1.0. Only zero-crossings whose slope is larger than this fraction of the maximum slope are preserved. If the threshold is 0.0, all zero-crossings are preserved.

**zcs.m**

```matlab
zc = zcs(conv);
```

zcs returns a 2D double-type matrix that contains the value 1.0 at the locations of zero-crossings in the input convolution result, and 0.0 elsewhere.

**overlayZC.m**

```matlab
newImage = overlayZC(image, zcMap);
```

overlayZC returns a 2D 8-bit matrix of the same size as the input 2D 8-bit image that contains white zero-crossing contours (stored in the input zcMap) superimposed on a low-contrast version of the original image.

**displayImage.m**

```matlab
displayImage(image, border);
```

displayImage is a short-cut function for invoking imtool on an image with the range of values that are contained within a central region that excludes a border around the outer edge of the image.