Image Scaling

In digital image processing, image scaling is actually image resizing, meaning that the number of pixels in the digital image is changing. When scaling a bit mapped image, a new digital image with a higher or lower number of pixels must be generated. In the case of decreasing the number of pixels (scaling down) this usually results in a visible quality loss. The process of resizing can be treated as a process of resampling with the same effects in the frequency domain such as aliasing. The filtering should be done to prevent aliasing and image quality loss.

In MatDeck, there is a function called image scale() for effective image scaling which can be used for image size increase and reduction. The width and height of the scaled image can be selected freely. The interpolation method, for better image quality, can be selected. The function supports two interpolation methods: nearest neighbor and bilinear interpolation.

In order to illustrate the various functionalities of MatDeck's functions, we will use the standard text image Lena with a size of 512 by 512 pixels. First, we will illustrate image reduction to 256 by 300 pixels, using the nearest neighbor interpolation method. Original and scaled images are shown.

```
img := image read("lena.png")
img_reduced := image scale(img , 128 , 200 , "nearest")
pic1 := image widget(0 , img)
set size(pic1 , 512 , 512)
```

```
pic2 := image widget(0 , img_reduced)
set size(pic2 , 128 , 200)
```

We can also increase the size of the original image using the same function, and same interpolation method.

```
img_enlarge := image scale(img , 600 , 550 , "nearest")
pic3 := image widget(0 , img_enlarge)
set size(pic3 , 600 , 550)
```

We see that the main effect of image scaling is the loss of details as the scaled image is not as sharp as the original.

Image Shear

Image scaling can be done by using the function image shear, as well. Here, image resizing and shear factor is controlled by using the scaling factors per each dimension.

```
img_s := image shear(img, 0.25, 0.25, "nearest")
pics := image widget(0, img_s)
wi := image width(img_s)
he := image height(img_s)
set size(pics, wi, he)
```