Graphical User Interfaces

The following functions, except for image.toDisplayTensor,
require package qtlua and can only be
accessed via the `qlua` Lua interpreter (as opposed to the
th or luajit interpreter).

[res] image.toDisplayTensor(input, [...])

Optional arguments `[...]` expand to `padding`, `nrow`, `scaleeach`, `min`, `max`, `symmetric`, `saturate`.
Returns a single `res` Tensor that contains a grid of all in the images in `input`.
The latter can either be a table of image Tensors of size `height x width` (greyscale) or `nChannel x height x width` (color),
or a single Tensor of size `batchSize x nChannel x height x width` or `nChannel x height x width`
where `nChannel=[3,1]`, `batchSize x height x width` or `height x width`.

When `scaleeach=false` (the default), all detected images
are compressed with successive calls to image.minmax:

```
image.minmax{tensor=input[i], min=min, max=max, symm=symmetric,
saturate=saturate}
```

`padding` specifies the number of padding pixels between images. The default is 0.
`nrow` specifies the number of images per row. The default is 6.

Note that arguments can also be specified as key-value arguments (in a table).

[res] image.display(input, [...])

Optional arguments `[...]` expand to `zoom`, `min`, `max`, `legend`, `win`,
`x`, `y`, `scaleeach`, `gui`, `offscreen`, `padding`, `symm`, `nrow`.
Displays `input` image(s) with optional saturation and zooming.
The `input`, which is either a Tensor of size `HxW`, `KxHxW` or `Kx3xHxW`, or list,
is first prepared for display by passing it through image.toDisplayTensor:

```
input = image.toDisplayTensor{
   input=input, padding=padding, nrow=nrow, saturate=saturate,
   scaleeach=scaleeach, min=min, max=max, symmetric=symm
}
```

The resulting `input` will be displayed using qtlua.
The displayed image will be zoomed by a factor of `zoom`. The default is 1.
If `gui=true` (the default), the graphical user inteface (GUI)
is an interactive window that provides the user with the ability to zoom in or out.
This can be turned off for a faster display. `legend` is a legend to be displayed,
which has a default value of `image.display`. `win` is an optional qt window descriptor.
If `x` and `y` are given, they are used to offset the image. Both default to 0.
When `offscreen=true`, rendering (to generate images) is performed offscreen.

[window, painter] image.window([...])

Creates a window context for images.
Optional arguments `[...]` expand to `hook_resize`, `hook_mousepress`,
`hook_mousedoublepress`.
These have a default value of `nil`, but may correspond to commensurate qt objects.