Saving and Loading

This sections includes functions for saving and loading different types of images to and from disk.

[res] image.load(filename, [depth, tensortype])

Loads an image located at path `filename` having `depth` channels (1 or 3) into a Tensor of type `tensortype` (*float*, *double* or *byte*). The last two arguments are optional.

The image format is determined from the `filename`'s extension suffix. Supported formats are JPEG, PNG, PPM and PGM.

The returned `res` Tensor has size `nChannel x height x width` where `nChannel` is 1 (greyscale) or 3 (usually RGB or YUV.

Usage:

```
--To load as byte tensor for rgb imagefile
local img = image.load(imagefile,3,'byte')

--To load as byte tensor for gray imagefile
local img = image.load(imagefile,1,'byte')
```

[res] image.getSize(filename)

Return the size of an image located at path `filename` into a LongTensor.

The image format is determined from the `filename`'s extension suffix. Supported formats are JPEG,

PNG,
PPM and PGM.

The returned `res` Tensor has size `3` (nChannel, height, width).

image.save(filename, tensor)

Saves Tensor `tensor` to disk at path `filename`. The format to which
the image is saved is extrapolated from the `filename`'s extension suffix.
The `tensor` should be of size `nChannel x height x width`.
To save with a minimal loss, the tensor values should lie in the range [0, 1] since the tensor is clamped between 0 and 1 before being saved to the disk.

[res] image.decompressJPG(tensor, [depth, tensortype])

Decompresses an image from a ByteTensor in memory having `depth` channels (1 or 3)
into a Tensor
of type `tensortype` (*float*, *double* or *byte*). The last two arguments
are optional.

Usage:

```lua
local fin = torch.DiskFile(imfile, 'r')
fin:binary()
fin:seekEnd()
local file_size_bytes = fin:position() - 1
fin:seek(1)
local img_binary = torch.ByteTensor(file_size_bytes)
fin:readByte(img_binary:storage())
fin:close()
-- Then when you're ready to decompress the ByteTensor:
im = image.decompressJPG(img_binary)
```

[res] image.compressJPG(tensor, [quality])

Compresses an image to a ByteTensor in memory. Optional quality is between 1 and 100 and adjusts compression quality.