

Image Plugin

The SOFA team

2011

Abstract

This document describes the SOFA image plugin.

1 Overview

2 Image Type

The `Image` data type is instantiated on the pixel types. The resulting `Image` types are:

- `ImageC` (char),
- `ImageUC` (unsigned char),
- `ImageI` (int),
- `ImageUI` (unsigned int),
- `ImageS` (short),
- `ImageUS` (unsigned short),
- `ImageL` (long),
- `ImageUL` (unsigned long),
- `ImageF` (float),
- `ImageD` (double),
- `ImageB` (bool).

All components of the plugin are instantiated on the image types (`ImageUC` by default).

Images have a total of five dimensions: the spatial dimensions $[x, y, z]$, the channels (for instance the three *rgb* color channels, the three values of a vectorial image, the six values of a tensor image, etc.) and the time.

Internally, SOFA uses the open-source `CImg` library (<http://cimg.sourceforge.net/>) to store and process images.

3 Image Transformations

Image transformation types have been implemented to describe the relation between 5d image integer coordinates and the real 3d spatial and temporal coordinate. They are instantiated on Reals (double or float precisions).

Currently, linear (scale+rotation+translation) and perspective transforms are implemented using a single structure. Therefore, components do not need to be instantiated on the transformation types (yet). A transformation vector is a concatenation of 12 parameters: spatial translation + rotation (3 Euler angles in degrees) + spatial scale (=pixel spacing) + temporal offset + temporal scale + a boolean to switch between linear and perspective transforms. For instance the translation [1, 2, 3] and orientation [10, 20, 30] of a 3D image of voxel size [4, 5, 6] can be described in a scene as : *transform* = "1 2 3 10 20 30 4 5 6 0 1 0".

Perspective transforms (only for 2D and 2D+t images) model pinhole cameras. Cameras are located in space at the position $[scale_x(dim_x - 1)/2, scale_y(dim_y - 1)/2, -scale_z/2]$. Note that the image origin (=value of translation vector) is kept at the image corner, and not displaced to the camera location as in the classic pinhole camera model (http://opencv.willowgarage.com/documentation/cpp/camera_calibration_and_3d_reconstruction.html). Changing coordinates, we get the classic camera intrinsics: $f_x = scale_z/(2 \times scale_x)$, $f_y = scale_z/(2 \times scale_y)$, $c_x = (dim_x - 1)/2$ and $c_y = (dim_y - 1)/2$.

Transformations are encapsulated into SOFA Data, and can therefore be easily transmitted from one component to another.

4 Input / Output

The following formats are supported by the ImageContainer (reader) and ImageExporter (writer) components :

- 2D:
 - ".bmp", ".jff", ".jif", ".ppm", ".pgm", ".pnm", ".pfm", ".exr", ".cr2", ".crw", ".dcr", ".mrw", ".nef", ".orf", ".pix", ".ptx", ".raf", ".srf"
 - compressed (requires imagemagick <http://www.imagemagick.org/Magick++>): ".png", ".jpg", ".jpeg", ".jpe", ".tiff", ".tif"
 - dicom (requires xmedcon (<http://xmedcon.sourceforge.net/>)): ".dcm", ".dicom"
- 2D + time:
 - videos (requires ffmpeg <http://www.ffmpeg.org>): ".avi", ".mov", ".asf", ".divx", ".flv", ".mpg", ".m1v", ".m2v", ".m4v", ".mjp", ".mkv", ".mpe", ".movie", ".ogm", ".ogg", ".qt", ".rm", ".vob", ".wmv", ".xvid", ".mpeg"
- 3D:
 - parrec: ".par", ".rec"
 - analyze: ".hdr", ".nii"
 - inria: ".inr"
- 3D + time :

- raw + meta image header file : ".raw" + ".mhd" (<http://www.vtk.org/Wiki/MetaIO/Documentation>)
- cimg: ".cimg"

Some formats (".par", ".rec" and ".inr") implement the pixel size. The spatial scale transformation parameters are retrieved automatically by the reader. For the metaimage format, a separate header file containing the transformation and the image dimensions is read/written. For the other formats, the transformation needs to be defined in the xml scene.

Only 2D dicoms are supported. To import a volume from dicom files, taking into account scale, orientation and position, the best way is to first convert dicoms series to metaimage using available software (such as <http://lcni.uoregon.edu/~jolinda/MRIConvert/>), and then load metaimages in sofa.

5 Engines

By default, images are shared memory images. So, there is no memory overhead if they are referenced into different components using **Data**. Engines can be implemented to process images. The classic image filters are implemented through the **ImageFilter** component (CImg binding).

6 GUI

7 Implementation