
– MDSTk – Medical Data Segmentation Toolkit: A Brief Guide

September 2007

Michal Španěl
Image@FIT
Faculty of Information Technology
Brno University of Technology



Outline

- Introduction
 - Architecture and basic features.
 - Libraries and modules.
- Installation
 - How to get sources?
 - Compilation
 - Using command line.
- Source code
 - Libraries and modules.
 - Documentation and coding rules.
 - How can I write my own module?
- Sample modules
- Libraries in detail
 - Base, Math, Image and Module library.
 - VectorEntity

– MDSTk –

Medical Data Segmentation Toolkit: A Brief Guide

Introduction

What is MDSTk?

- Medical Data Segmentation Toolkit
- Collection of 2D/3D image processing tools aimed originally at medical image segmentation.

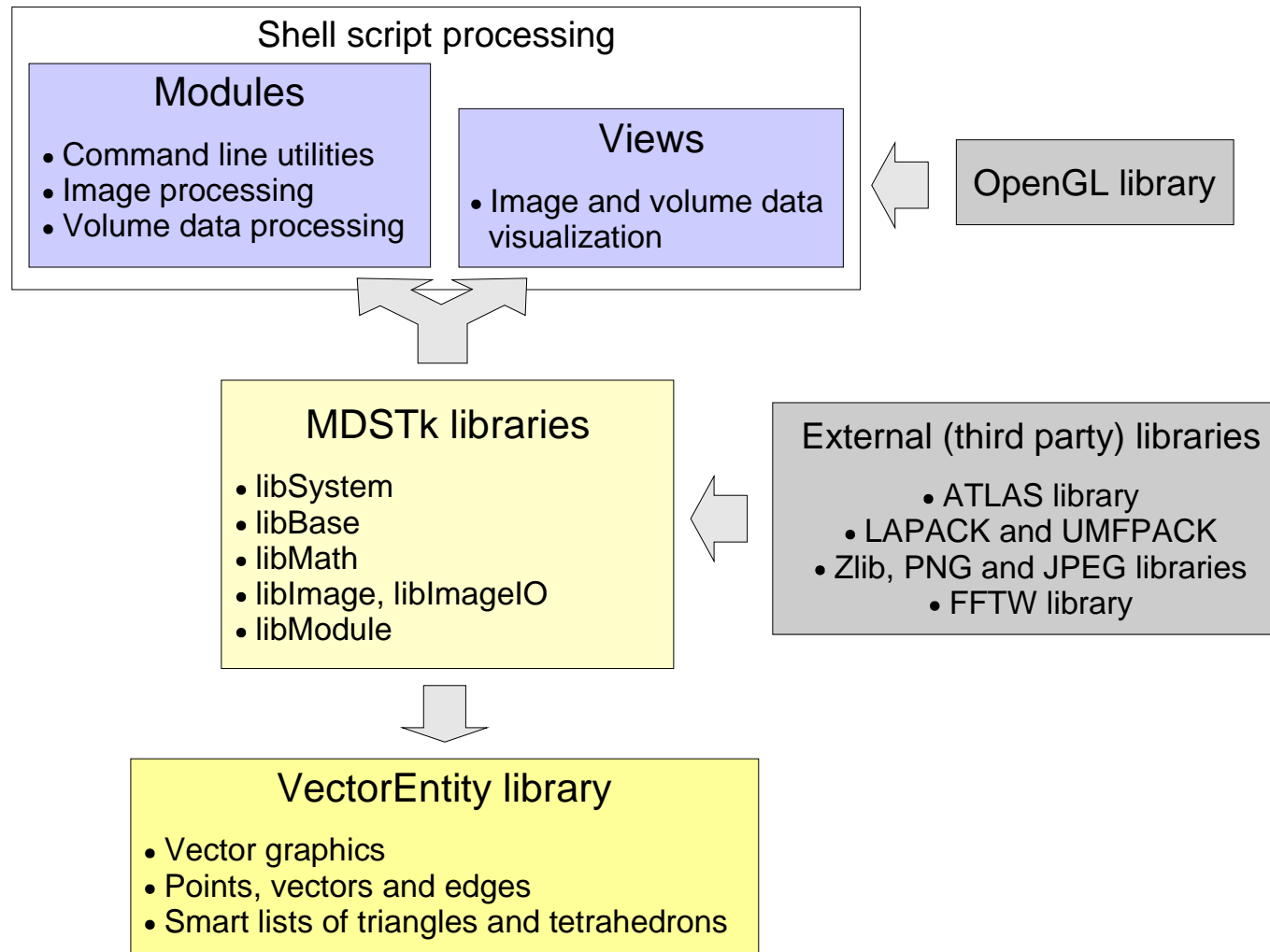
Authors and Major Contributors

- Španěl, M.:
 - Main author of the MDSTk and current maintainer.
- Kršek, P.:
 - Author of the VectorEntity library.
 - DICOM format parser.
- Švub, M.:
 - Image processing routines.
 - Compilation using CMake build system.
- and many others (see the documentation and source codes).
- All contributors in any way are welcome!

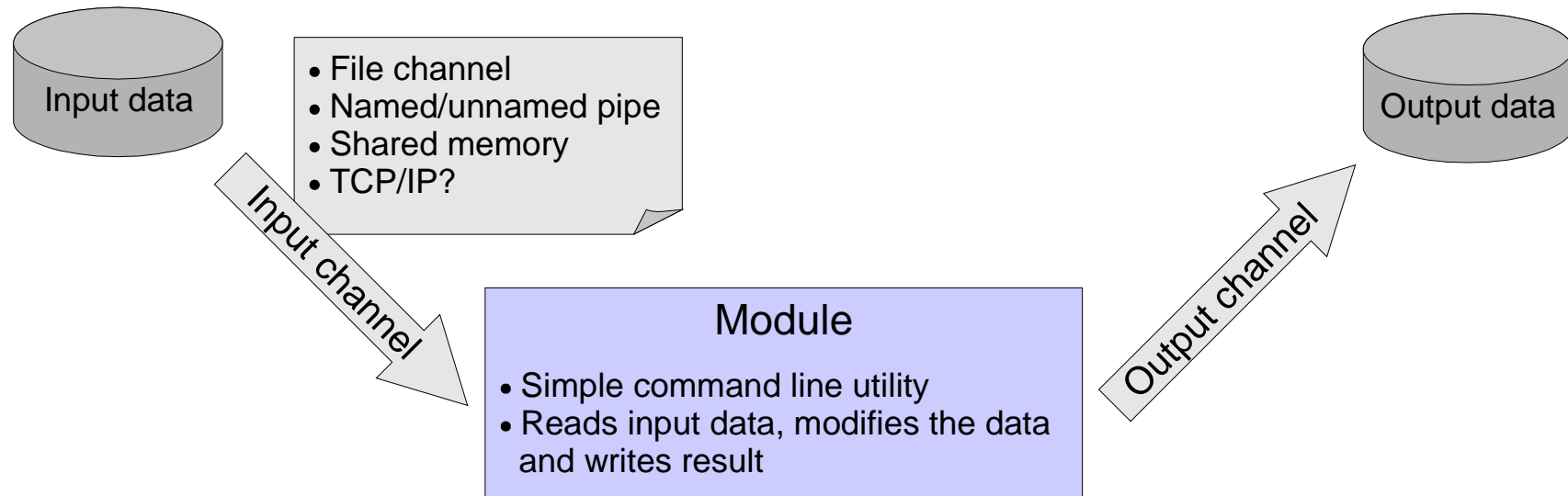
Basic features

- **Multiplatform** library (Windows/Linux), **CMake** build system.
- Simple **modular** architecture.
- **Modern** C++ design (templates, smart pointers, singletons, iterators, serialization, etc.).
- **Math library** (including precompiled ATLAS, LAPACK and UMFPACK libraries).
- **Image processing library** (support for 2D and 3D images).
- **DICOM 3.0** format parser.
- Effective low-level **vector graphics library**.
- Partially prepared for **multithreaded** applications.
- **Open source** under BSD like license.

Toolkit architecture



Modules



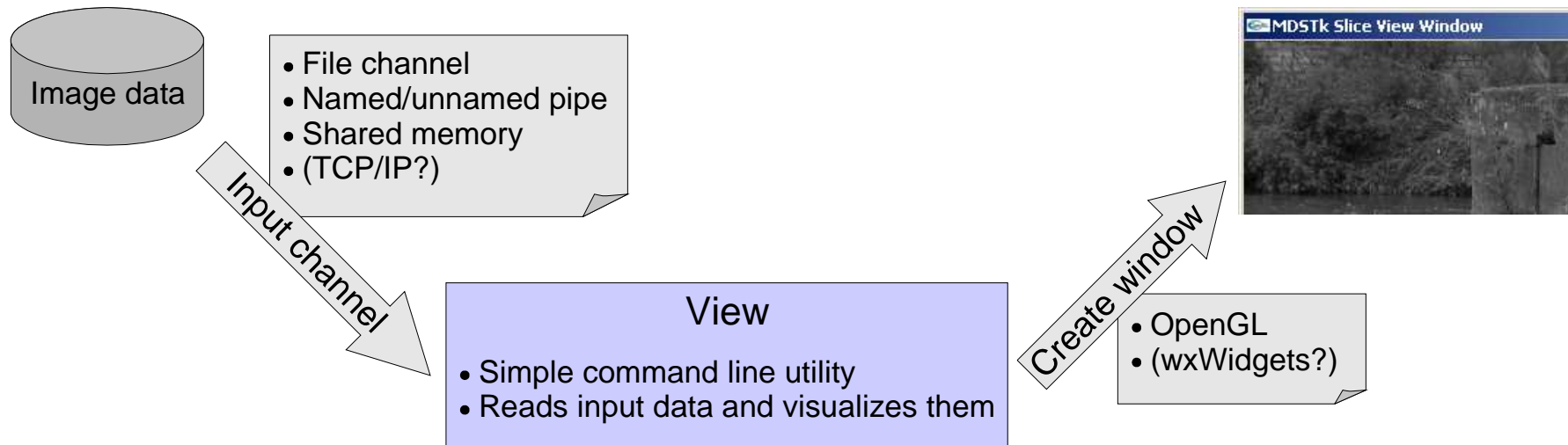
■ Sample

```
> mdsLoadJPEG <berounka.jpg |mdsSliceRange |mdsSliceView
```

Read input
from file

Redirect output (stdout) to the
following module through an
unnamed pipe

Views



■ Sample, cont.

```
> mdsLoadJPEG <berounka.jpg | mdsSliceRange | mdsSliceView
```

Read input JPEG image and convert it to the medical slice (12-bits grayscale image).

Normalize intensity to the interval <0..255>.

Modules and views: command line usage

- > mdsModule **-h** ... shows help
- > mdsModule **-log** {null | stderr | **file** | both} ... logging
- Input channel specification
 - > mdsModule **<filename** ... input via stdin
 - > mdsModule **-i** ch1[::ch2[::ch3]...]
- Channel description **chN**
 - **medium[:attrib1[:attrib2[...]]]**
medium ... {**file** | **stdio** | **pipe** | **shm**}
attributes ... {**file:name** | **stdio** | **pipe:name** | **shm:name**}

Modules and views: command line usage

- Output channels (... analogically ...)

- > mdsModule >filename

- > mdsModule -o ch1[::ch2[::ch3]...]

- Sample

- > mdsLoadJPEG -i file:../temp/berounka.jpg \

- |mdsSliceRange -o stdio \

- |mdsSliceView -i stdio

More samples

> `mdsLoadJPEG -format rgb <berounka.jpg | mdsRGBImageView`

> `mdsLoadJPEG -format slice <berounka.jpg \
| mdsSliceRange \
| mdsSliceView`



More samples

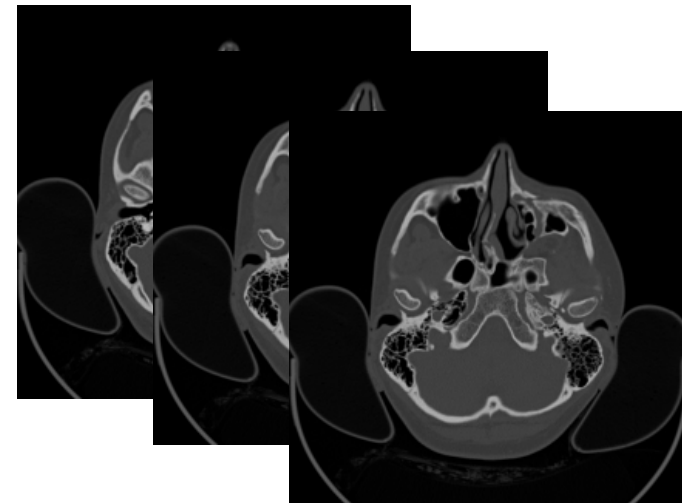
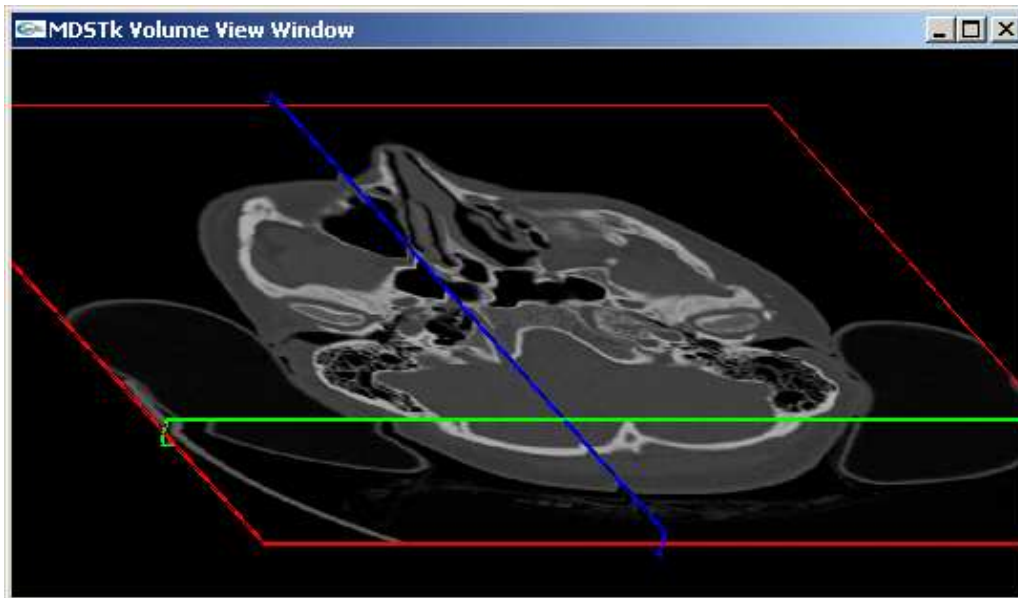
```
> mdsLoadJPEG <berounka.jpg \  
  | mdsSliceFilter -filter IF_SOBEL_Y \  
  | mdsSliceRange -max 1000 \  
  | mdsSliceView
```



berounka.jpg

More samples (Linux shell only)

```
> for i in `ls -d data/dicom/*.dcm`; do \  
    mdsLoadDicom <$i >>slices.slc; done  
> mdsMakeVolume <slices.slc \  
    | mdsVolumeRange \  
    | mdsVolumeView
```



75.dcm, 76.dcm, ... , 85.dcm

– MDSTk –

Medical Data Segmentation Toolkit: A Brief Guide

Installation

How to get sources?

- Latest release version is [0.7.0beta](#).
- Download from the web
 - Hosted by SourceForge.net
<http://sourceforge.net/projects/mdstk>
 - Home page
<http://mdstk.sourceforge.net/>
- Access via SVN
 - Private for now (developers only)
<svn+ssh://user@merlin.fit.vutbr.cz/svn/MediTools/MDSTk/tags/v0.7.0beta>

Access via SVN (developers only)

- Subversion archive structure
 - [.../svn/MediTools/MDSTk/trunk](#) – current developing version
 - [.../svn/MediTools/MDSTk/tags/vX.X.X](#) – released “stable” version ☺
- Sample
 - > `svn checkout svn+ssh://.../trunk ~/MediTools/MDSTk`
 - > `svn export svn+ssh://.../tags/v0.7.0beta ~/MediTools/MDSTk`

SVN archive

Destination directory
 - > `svn commit ~/MediTools/MDSTk`
- TortoiseSVN
 - “The coolest interface to (sub)version control.”
 - SVN client for Windows.
 - <http://tortoisesvn.tigris.org/>

Installation prerequisites – Linux

- GCC 3.2+ and GNU Make
- CMake build system
 - <http://www.cmake.org/>
- OpenGL and GLUT graphic libraries

Both libraries are already installed in all standard Linux distributions!

Installation prerequisites – Windows & MinGW

- MinGW (Minimalist GNU for Windows)
 - GCC 3.2+ and GNU Make
 - Windows API header files.
 - <http://www.mingw.org/>

- MSYS (Minimal SYStem)
 - Linux shell on windows.
 - <http://www.mingw.org/msys.shtml>

- GLUT library
 - Can be found in the 'MDSTk/misc/glut/mingw_glut-v3.7.6.zip' file.
 - Unzip archive to your MinGW directory and place 'glut32.dll' to 'Windows\system32' directory.

Installation prerequisites – MS Visual C++

- MS Visual C++ 7.1 (2003) or later
- GLUT library
 - Precompiled library can be found in the 'MDSTk/misc/glut/vc_glut-v3.7.6.zip' file.
 - Unzip archive to your MSVC directory and place 'glut32.dll' to your 'Windows\system32' directory.

Compilation – Linux

- > `cd MDSTk` ... change to the MDSTk root directory
- > `mkdir build` ... directory for an out-of-source build

- > `ccmake ..` ... run `cmake` or `ccmake` configuration utility which generates all Makefiles
- or
- > `ccmake .. -DCMAKE_BUILD_TYPE=Debug` ... debug version

- > `make` ... compile all targets specified in configuration step

- > `make doc` ... compile the documentation

Compilation – Windows & MinGW

- Check that windows system **PATH** variable is correctly set to '**MSYS\bin**' and '**MinGW\bin**' directories!
(Control Panel->System->Advanced->Environment Variables)
- Use the **CMakeSetup** utility to generate Makefiles for MSYS.
- Enter build directory different from the source one.
- Open the command line window **Start->Run cmd**

> cd MDSTk/build	... change to the build directory
> make	... run compilation
> make doc	... compile documentation

Compilation – MS Visual C++

- Use the same [CMakeSetup](#) utility to generate MSVC projects and solution.

- 1. Open solution [MDSTk.sln](#) in the MDSTk build directory.
- 2. Select [ALL_BUILD](#) project and compile all libraries and modules.
- 3. Optionally choose [ALL_DOC](#) and compile the documentation.

Setting shell variables

- Not necessary! But it helps you.
- Modify files/scripts
 - `lsetenv` ... Linux
 - `msetenv` ... Windows & MinGW & MSYS
 - `wsetenv` ... MS Visual C++
 - `cygsetenv` ... Cygwinin the MDSTk root directory.
- Execute appropriate script from the command line at the beginning of your work ...

Comprehensive sample

1. Open command line window.

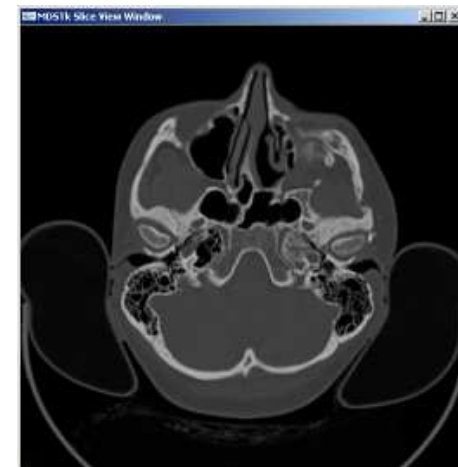
- > `cd ../MDSTk/build` ... change to the build directory
- > `ccmake ..` ... run CMake
- > `make` ... compile all libraries and modules

2. Modify the `./lsetenv` file according to your path.

- > `./lsetnev` ... set the PATH variable
- > `cd temp` ... working directory

3. Let's show one of the sample DICOM images.

- > `mdsLoadDicom <../data/dicom/80.dcm \`
 `| mdsSliceRange \`
 `| mdsSliceView`



Comprehensive sample, cont.

4. Show slice information ...

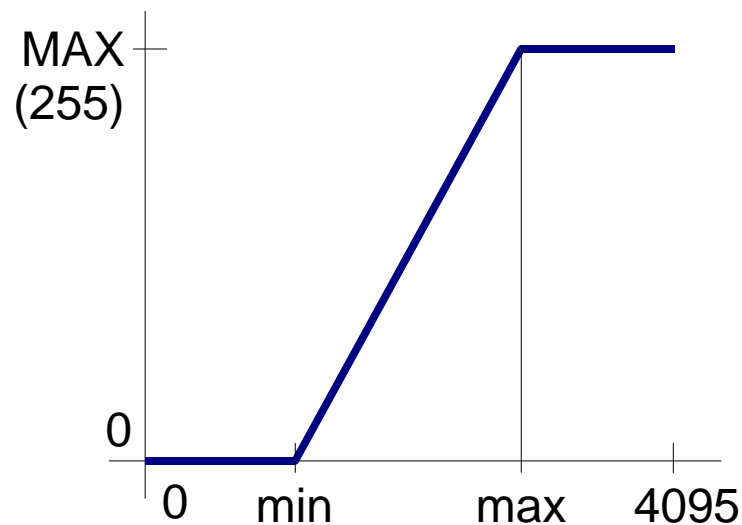
> `mdsLoadDicom <../data/dicom/80.dcm | mdsSliceInfo`

```
sh-2.04$  
sh-2.04$  
sh-2.04$  
sh-2.04$  
sh-2.04$  
sh-2.04$  
sh-2.04$  
sh-2.04$  
sh-2.04$  
sh-2.04$  
sh-2.04$  
sh-2.04$  
sh-2.04$ mdsLoadDicom <../data/dicom/80.dcm | mdsSliceInfo  
-- Information on slice image data --  
Width:      512  
Height:     512  
Index:      80  
Position:   -594.60  
Pixel:      0.45, 0.45  
Thickness:  1.30  
-- Pixel values statistic --  
Minimum:    0  
Maximum:    3103  
Mean value: 486.02  
Variance:   639.90  
sh-2.04$  
sh-2.04$ █
```

Comprehensive sample, cont.

5. Adjust intensity of the image ...

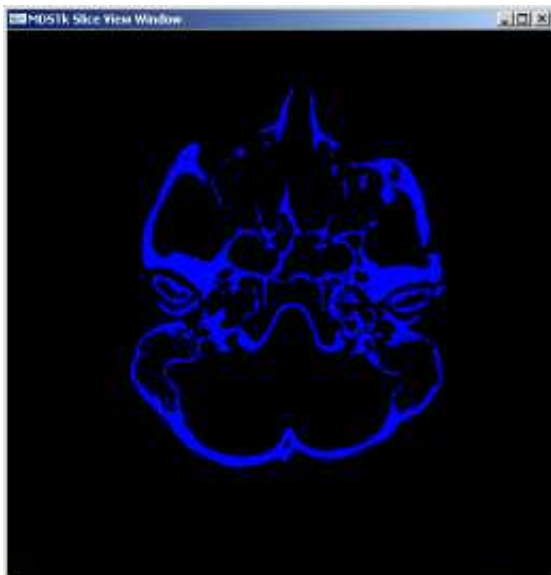
```
> mdsLoadDicom <../data/dicom/80.dcm \  
    | mdsSliceRange -min 1000 -max 2500 \  
    | mdsSliceView
```



Comprehensive sample, contd.

6. Simple thresholding ...

```
> mdsLoadDicom <../data/dicom/80.dcm \  
> | mdsSliceThresholding -min 1500 -max 4095 \  
| mdsSliceView -coloring segmented
```

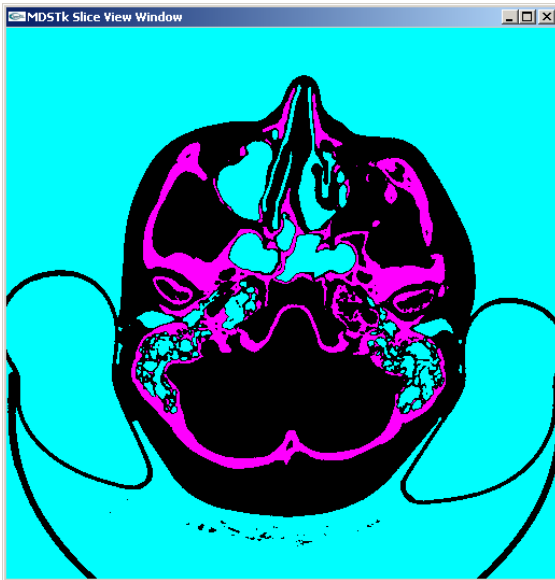


Input slice is a **labeled (segmented) image**. Pixel value represents index of an image region.

Comprehensive sample, contd.

7. Let's try some “more sophisticated” segmentation technique ...

```
> mdsLoadDicom <../data/dicom/80.dcm \  
> | mdsSliceSegFCM -clusters 3 \  
> | mdsSliceView -coloring segmented
```



Result of the Fuzzy C-Means (FCM) segmentation algorithm.

Comprehensive sample, contd.

8. Compute FFT of an image ...

```
> mdsLoadDicom <../data/dicom/75.dcm \
```

```
> | mdsSliceFFT -result abs -shift -logspect \
```

```
| mdsSliceRange \
```

```
| mdsSliceView
```

Use logarithmic scale.

Return absolute value (magnitude) of the spectrum.

Shift zero-frequency component to the center.























Magnitude spectrum of the input image on logarithmic scale.

– MDSTk –

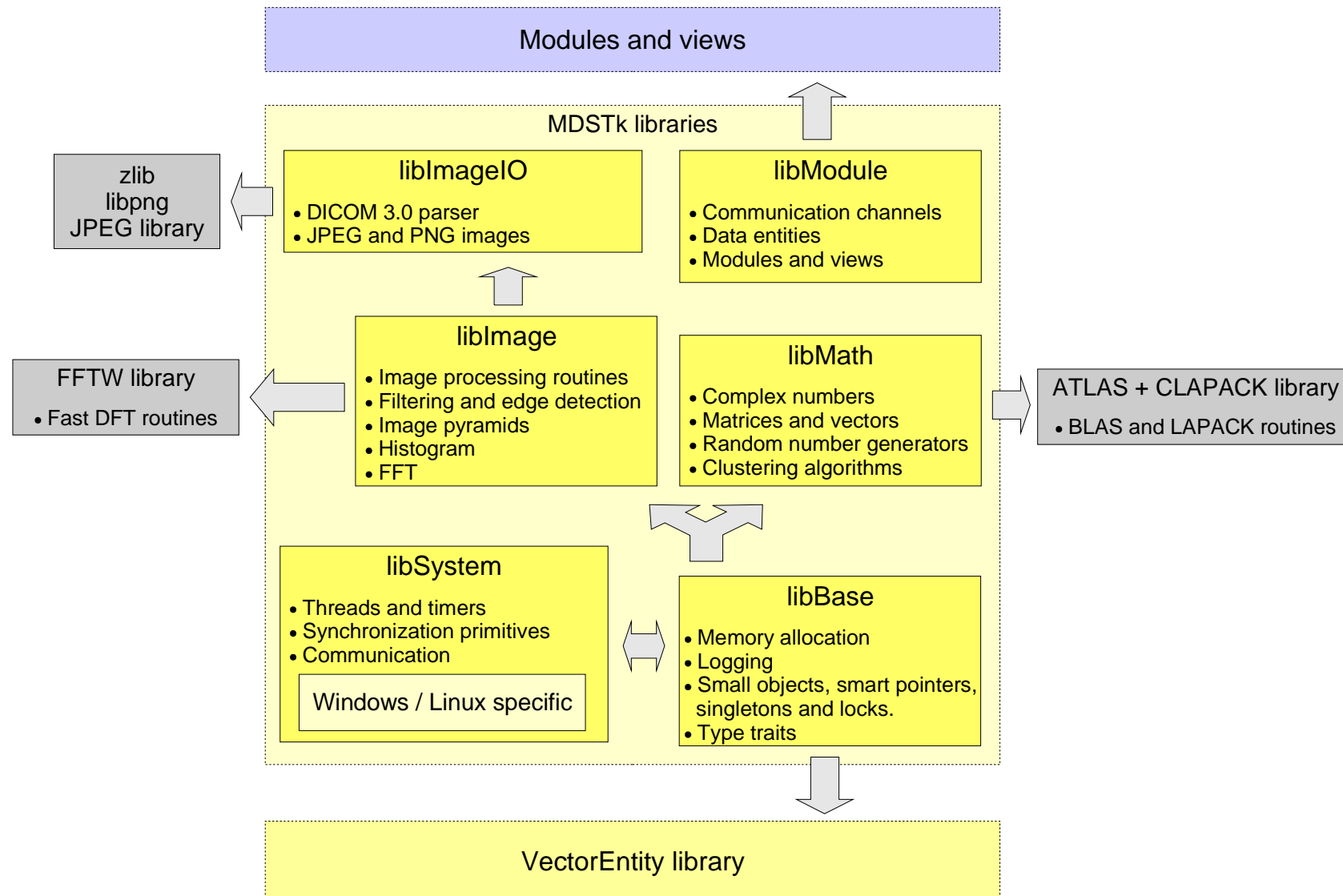
Medical Data Segmentation Toolkit: A Brief Guide

Source Code

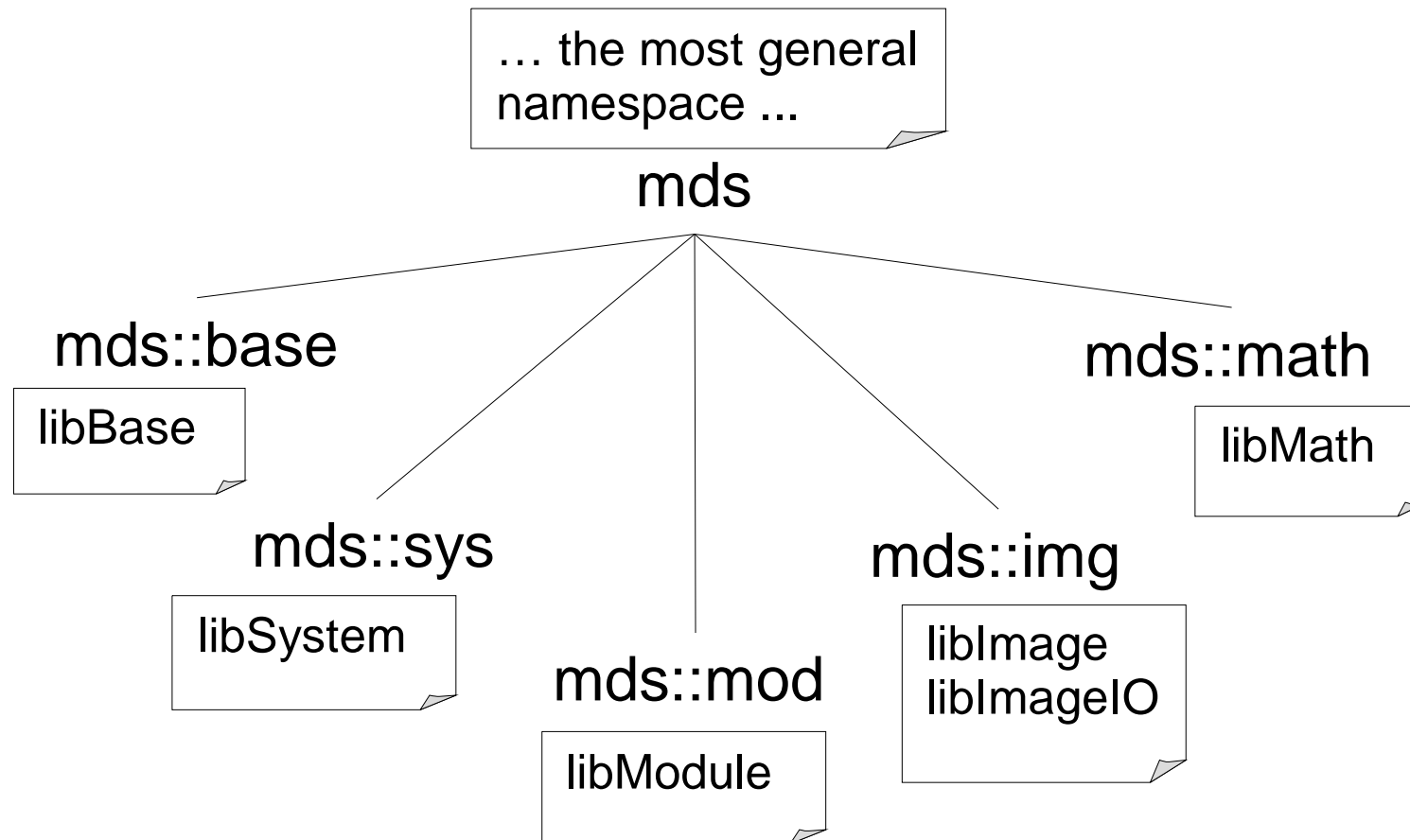
Directory structure

- [-]  MDSTk
 -  build ... default build directory
 - [-]  data ... **sample medical image data** in the DICOM format.
 -  dicom
 - [+]  doc
 - [-]  include
 - [+]  MDSTk ... **header files** of the MDSTk libraries.
 -  VectorEntity ... VectorEntity header files.
 - [-]  lib
 -  dll ... all required **dll libraries** (Windows only)
 - [+]  linux ... third party **precompiled libraries**
 - [+]  vc
 - [+]  win
 - [+]  misc
 -  scripts
 - [-]  src ... **source files** of the MDSTk libraries.
 - [+]  lib ... source files of all modules.
 - [+]  modules ... various **testing utilities**.
 - [+]  test ... my helper working directory.
 -  temp

MDSTk libraries



Namespaces



Using MDSTk libraries

- Remember to tell compiler where the header files (and compiled libraries) can be found, e.g.
 - GCC ... `-D_LINUX -I.../MDSTk/include -I.../MDSTk/build/include -L.../MDSTk/build/lib`
_LINUX ... compilation on Linux.
 - MS Visual C++ ... set project properties
_WIN32 ... defined by the compiler.
- Include pattern
`#include <MDSTk/Base/mdsSetup.h>` ... header file which must be included foremost.
`#include <MDSTk/Image/mdsImage.h>`
`#include <MDSTk/Math/mdsMatrix.h>`
`#include <VectorEntity/mctris.h>`

Sample “ASCII table” application

- Just an [illustration of the compilation](#).
- ASCII table represented as a matrix of characters.

```
majkl@NBSPANEL /d/majkl/skola-ds/prezentace/2006/MDSTk/samples/ASCIITable
$ ./ASCIITable
Table of ASCII characters starting with space:
! " # $ % & ' ( ) * + , - . / 0 1 2 3 4 5 6 7 8
9 : ; < = > ? @ A B C D E F G H I J K L M N O P Q
R S T U V W X Y Z [ \ ] ^ _ ` a b c d e f g h i j
k l m n o p q r s t u v w x y z { | } ~ ¡ ¢ £ ¤ ¥ ¦ § ¨
                  ¡ ¢ ¤ ¥ ¦ § ¨ ¨ ª « ¬ ® ¯
° ± ² ³ ´ µ ¶ · ¸ ¹ º » ¼ ½ ¾ ¿
majkl@NBSPANEL /d/majkl/skola-ds/prezentace/2006/MDSTk/samples/ASCIITable
$
```

... see the ‘[samples/ASCIITable](#)’ directory.

Sample “ASCII table” application, contd.

- Source code

```
#include <MDSTk/Base/mdsSetup.h>
#include <MDSTk/Math/mdsMatrix.h>
#include <MDSTk/Math/mdsMatrixFunctions.h>
#include <iostream>

int main(int argc, char *argv[])
{
    typedef mds::math::CMatrix<unsigned char> tTable;

    tTable Table(10, 25, ' ');
    tTable::tIterator itEnd = Table.getEnd(), it = Table.getBegin();
    for( unsigned char ucValue = 32; it != itEnd; ++it, ++ucValue )
    {
        *it = ucValue;
        if( ucValue == 255 ) break;
    }

    std::cout << "Table of ASCII characters starting with space:" << std::endl;
    std::cout << Table << std::endl;

    return 0;
}
```

Sample “ASCII table” application, contd.

- Makefile for Linux

```
GCC = g++
TARGET = ASCIIITable
MDSTK = /home/majkl/MediTools/MDSTk
MDSTK_BUILD = $(MDSTK)/build

INCLUDES = -I$(MDSTK)/include -I$(MDSTK_BUILD)/include
LIBS = -L$(MDSTK_BUILD)/lib -lImageIO -lImage -lModule -lMath
        -lSystem -lBase -lpthread -lrt
DEFINES = -D_LINUX

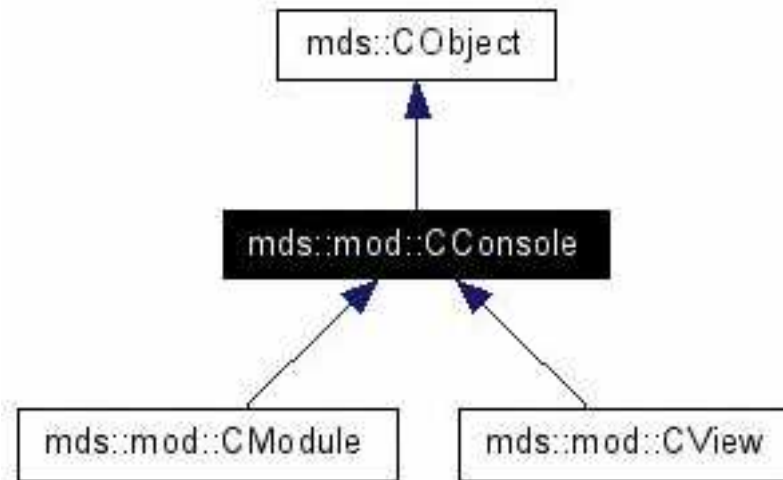
all: $(TARGET)

ASCIIITable.o: ASCIIITable.cpp
    $(GCC) -c -o $@ $< $(DEFINES) $(INCLUDES)

$(TARGET): ASCIIITable.o
    $(GCC) -o $@ $< $(LIBS)
```

MDSTk modules

```
#include <MDSTk/Module/mdsModule.h>
```



- Command line arguments parsing.
- Creation and initialization of I/O channels.
- Initialization of the logging interface.
- etc.

MDSTk modules, contd.

- Following **virtual methods must be implemented!**

virtual bool `startup()` ... called once on module startup
(returns *false* on failure).

virtual bool `main()` ... repeatedly called by the processing
thread until *false* is returned.

virtual void `shutdown()` ... called once on module
shutdown.

virtual void `writeExtendedUsage(std::ostream& Stream)`
... called on writing usage to the *std::cerr* (*-h* argument).

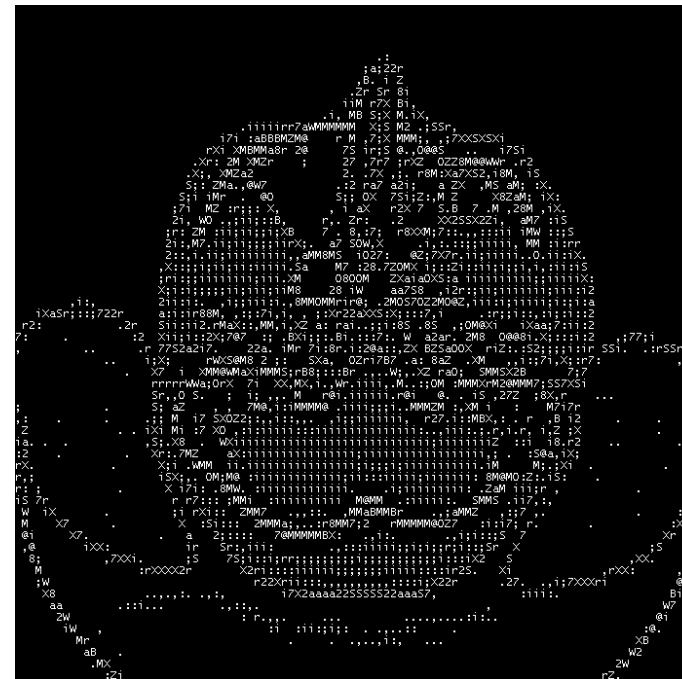
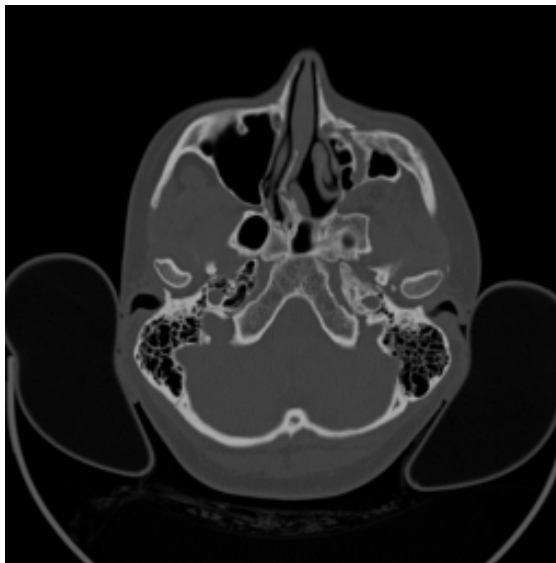
– MDSTk –

Medical Data Segmentation Toolkit: A Brief Guide

Sample modules

Sample 1: ASCII art

- Convert an input grayscale image (slice) to text (**ASCII art**).
- **Input**: image/slice.
- **Output**: text

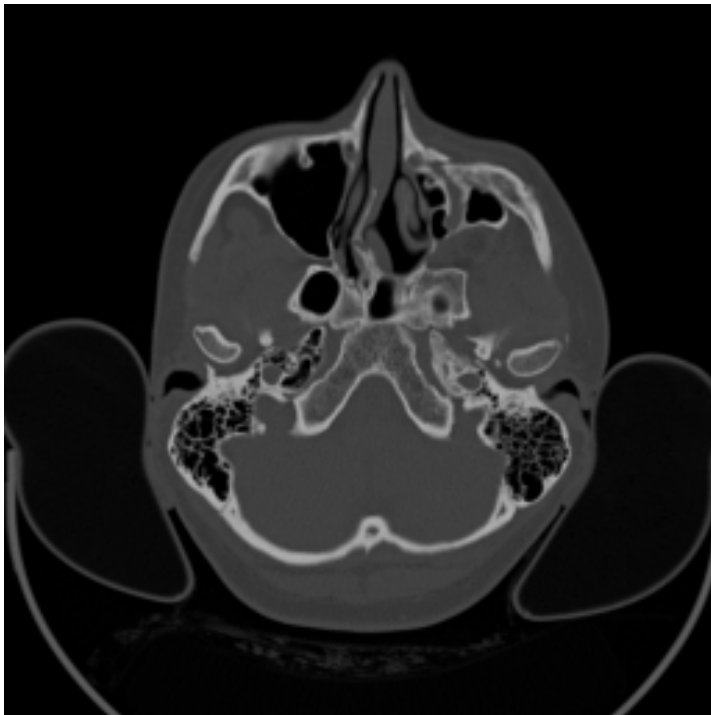


[<http://ascgen.jmsoftware.co.uk/>]

... see the '[samples/Slice2Text](#)' directory.

Sample 2: Image segmentation

- **Input:** image/slice.
- **Output:** labeled slice (pixel = index of an image region).



... see the '[MDSTk/src/modules/mdsSliceSeg](#)' directory.

– MDSTk –

Medical Data Segmentation Toolkit: A Brief Guide

Libraries in detail

Type traits

```
#include <MDSTk/Base/mdsTypeTraits.h>
```

- Provides information on a given type.
- Template `mds::CTypeTraits` specialized for concrete types.
- Static methods `getMin()`, `getMax()`, etc.

- Usage

```
int iMaxAllowedValue = mds::CTypeTraits<int>::getMax();
```

```
bool bIsFloatNum = mds::CTypeTraits<T>::isFloat;
```

```
mds::CTypeTraits<T>::tPointee           // Pointee type if T is pointer
```

Smart pointers

```
#include <MDSTk/Base/mdsSharedPtr.h>
```

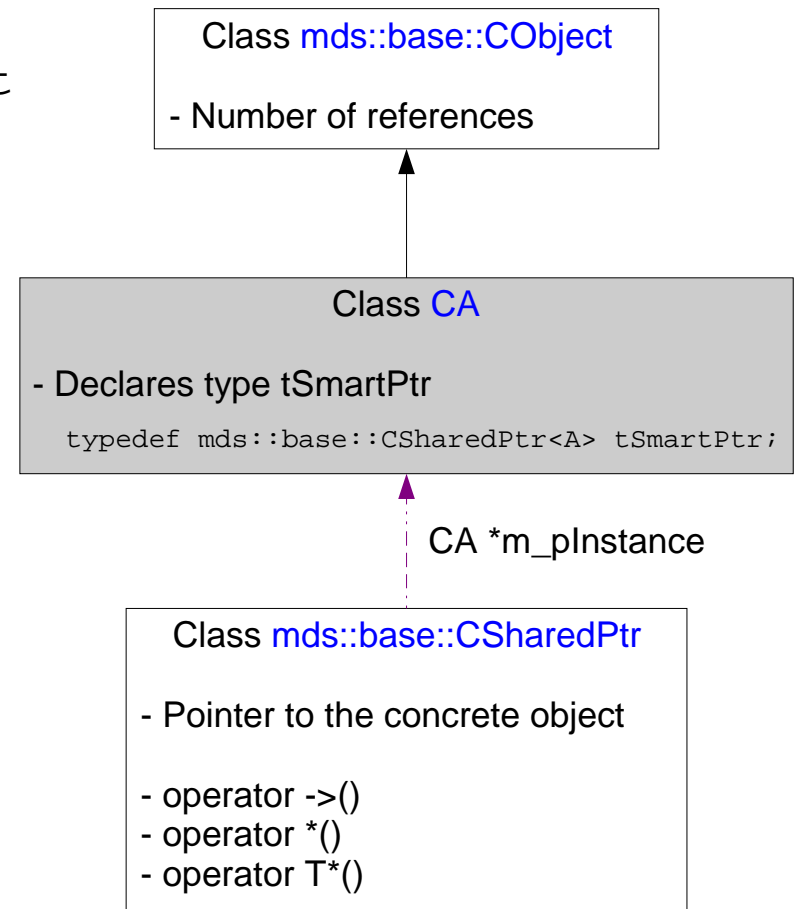
- Intrusive reference counting model → each object (derived from `mds::base::CObject` class) handles number of references to itself.
- Behaviour similar to the common pointers.
- The object is destroyed when the number of references decreases to zero.

Smart pointers, contd.

```
#include <MDSTk/Base/mdsSharedPtr.h>
```

```
class CA : public mds::base::CObject
{
public:
    // Macro declares type tSmartPtr
    MDS_SHARED_PTR(CA);
}
```

```
typedef CA::tSmartPtr CAPtr;
```



Smart pointers, contd.

- Initialization

```
CA::tSmartPtr spA1(new CA);    // Init number of references (= 1)
CA::tSmartPtr spA2(spA1);      // Number of refs. = 2
```

- Usage

```
spA1->method( )
spA1->member
*spA1
spA1 = spA3;    // Decreases the number of references to A1 object and
                // increases num. of refs. to A3 object.
```


Iterators

```
#include <MDSTk/Base/mdsIterator.h>
```

- Generalization of pointers.
- Used to traverse items stored in a container.
- Every container provides types
 - `Iterator`
 - `ConstIterator`
- and typical methods
 - `getBegin()`
 - `getEnd()`

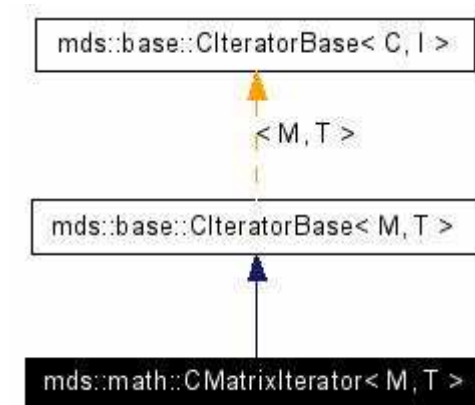
Iterators, contd.

```
#include <MDSTk/Base/mdsSetup.h>
#include <MDSTk/Math/mdsMatrix.h>

// Create a new matrix 10x10
typedef mds::math::CMatrix<double> tMyMatrix;
tMyMatrix M(10,10);

// Zero the matrix
tMyMatrix::tIterator itEnd = M.getEnd();
for(tMyMatrix::tIterator it = M.getBegin(); it != itEnd; ++it )
{
    *it = 0.0;
}

// or simply ☺
M.zeros();
```



Iterators, contd.

```
// Count the number of items.  
// More accurately, the number of increments (++) between two given  
// iterators.  
tMyMatrix::tIterator itBegin = M.getBegin();  
mds::tSize Count = itBegin.getDistance(itEnd);
```

Matrices and vectors

```
#include <MDSTk/Math/mdsMatrix.h>
```

- Template `mds::math::CMatrix<T>`, parameter T is type of the matrix element.

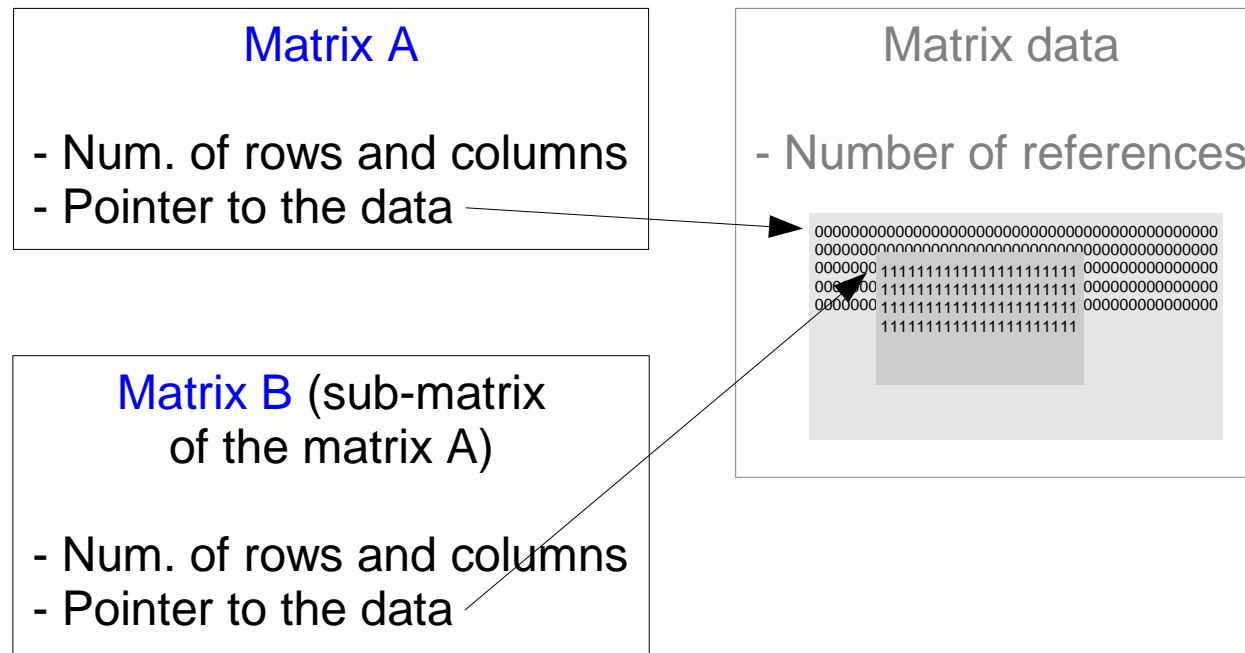
- Predefined types

```
typedef mds::math::CMatrix<int> CIMatrix;  
typedef mds::math::CMatrix<double> CDMatrix;  
...
```

- Stored in **row-major format**.
- Allowed to initialize matrix as reference to an existing one (**sub-matrix**).

Matrices and vectors, contd.

```
#include <MDSTk/Math/Matrix.h>
```



```
mds::math::CIMatrix A(100, 50);
```

```
A.fill(0);
```

```
mds::math::CDMatrix B(A,   
                        position size make reference  
                        20, 10, 50, 20, mds::REFERENCE);  
B.fill(1);
```

Matrices and vectors, contd.

```
#include <MDSTk/Math/mdsMatrix.h>

// Create and initialize a new matrix
mds::math::CDMatrix M(10, 10);
M.fill(0.0);

// Another way of the initialization
for( mds::tSize j = 0; j < M.getNumOfRows(); ++j )
    for( mds::tSize i = 0; i < M.getNumOfCols(); ++i )
    {
        M(i,j) = 0.0; // Method get(i,j) also returns reference!
//        M.set(i,j 0.0);
    }

// Iterators
mds::math::CDMatrix::tIterator it, itEnd = M.getEnd();
for( it = M.getBegin(); it != itEnd; ++it ) *it = 0.0;
```

Matrices and vectors, contd.

```
#include <MDSTk/Math/mdsVector.h>
```

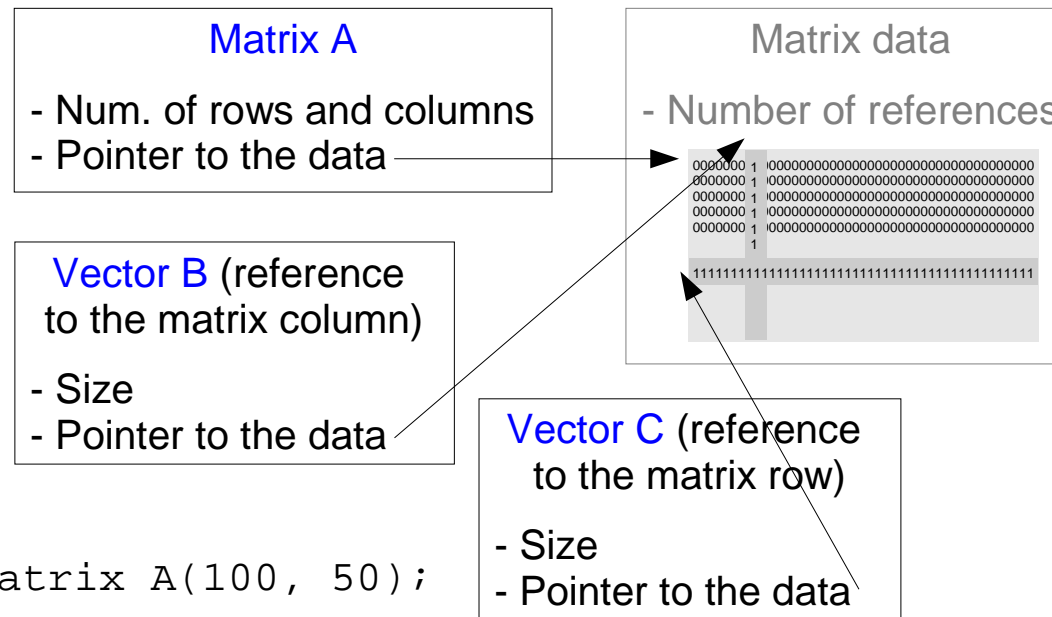
- Template `mds::math::CVector<T>`
- Predefined types

```
typedef mds::math::CVector<double> CDVector;  
...
```
- Reference to an existing vector (`sub-vector`).
- Reference to a matrix row, or column.
- Iterators

Matrices and vectors, contd.

```
#include <MDSTk/Math/Matrix.h>
```

```
#include <MDSTk/Math/Vector.h>
```



```
mds::math::CIMatrix A(100, 50);  
A.fill(0);
```

```
mds::math::CDVector B(A, 20, mds::math::COL_REFERENCE);  
B.fill(1);  
mds::math::CDVector C(A, 35, mds::math::ROW_REFERENCE);  
C.fill(1);
```

Diagram illustrating the structure of Matrix A, Vector B, and Vector C, and their relationship to Matrix data.

Matrix A: Num. of rows and columns, Pointer to the data

Matrix data: Number of references

Vector B (reference to the matrix column): Size, Pointer to the data

Vector C (reference to the matrix row): Size, Pointer to the data

Diagram illustrating the structure of Matrix A, Vector B, and Vector C, and their relationship to Matrix data.

Matrix A: Num. of rows and columns, Pointer to the data

Matrix data: Number of references

Vector B (reference to the matrix column): Size, Pointer to the data

Vector C (reference to the matrix row): Size, Pointer to the data

Diagram illustrating the structure of Matrix A, Vector B, and Vector C, and their relationship to Matrix data.

Matrix A: Num. of rows and columns, Pointer to the data

Matrix data: Number of references

Vector B (reference to the matrix column): Size, Pointer to the data

Vector C (reference to the matrix row): Size, Pointer to the data

Matrices and vectors, contd.

```
#include <MDSTk/Math/MatrixFunctions.h>
```

```
#include <MDSTk/Math/VectorFunctions.h>
```

- Global functions

- `mds::math::getMin(...)` ... Returns minimal value.
- `mds::math::getSum(...)` ... Sum of all values.
- `mds::math::getMean(...)` ... Mean value of all matrix/vector elements.
- etc.

Matrices and vectors, contd.

```
#include <MDSTk/Math/Vector.h>
#include <MDSTk/Math/VectorFunctions.h>

// Initialize the vector
mds::math::CIVector V(50);
for( mds::tSize i = 0; i < V.getSize(); ++i )
{
    V(i) = 2 * i;
}

// Get minimal value
int iMin = mds::math::getMin<int>(V);

// Compute mean value
double dMean = mds::math::getMean<double>(V);
```

Pixel types

```
#include <MDSTk/Image/mdsPixelTypes.h>
```

■ Pixel types

<code>tPixel8</code>	... 8-bit grayscale image, <0..255>
<code>tPixel16</code>	... 16-bit grayscale/intensity image
<code>tFloatPixel</code>	... float image, <-1,0..1,0>
<code>tDensityPixel</code>	... 12-bit medical image <0..4095>
<code>tComplexPixel</code>	... complex image, float RE/IM components
<code>tRGBPixel</code>	... RGB image, 8-bits R/G/B components

Pixel traits

```
#include <MDSTk/Image/mdsPixelTraits.h>
```

- Information on the pixel type.
- Template `mds::img::CPixelTraits` specialized for concrete types.
- Static methods `getPixelMin()`, `getPixelMax()`, etc.
- `getGray()` ... 50% gray (e.g. 128 in case of 8-bit images).
- Usage

```
mds::img::tDensityPixel Min, Gray;  
Min = mds::img::CPixelTraits<mds::img::tDensityPixel>::getPixelMax();  
Gray = mds::img::CPixelTraits<mds::img::tDensityPixel>::getGray();
```

Images

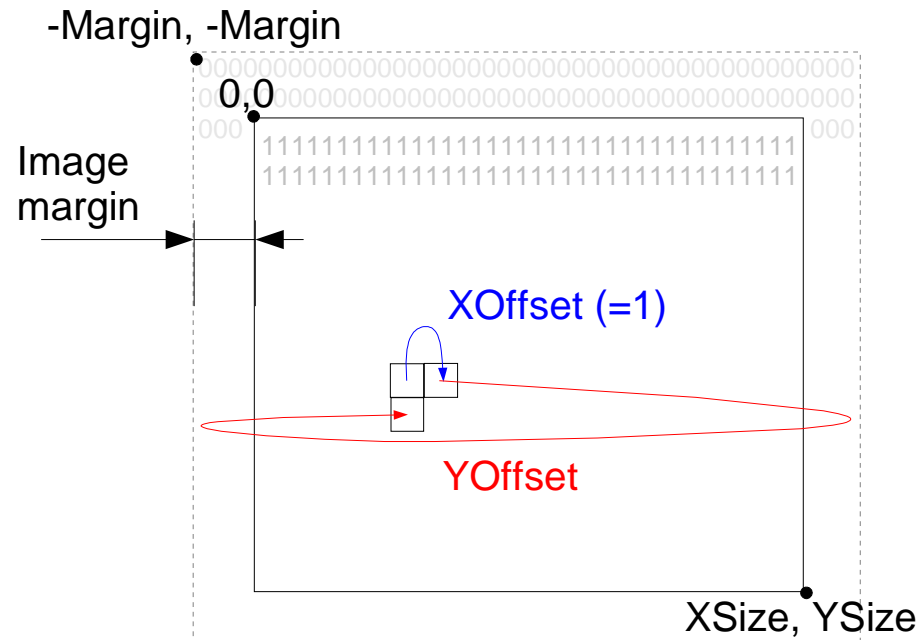
```
#include <MDSTk/Image/mdsImage.h>
```

- Template `mds::img::CImage<T>`, parameter T is type of the image pixel.
- Predefined images

```
typedef mds::img::CImage<mds::img::tPixel8> CImage8;  
typedef mds::img::CImage<mds::img::tDensityPixel> CDImage;  
...
```
- References to existing images (sub-images).
- Image is a data entity → it can be written to (read from) an I/O channel.
- Iterators.
- Conversion functions (`mds::img::CImage8` ↔ `mds::img::CFloatImage`, ...).

Images, contd.

```
#include <MDSTk/Image/mdsImage.h>
```



```
size margin
mds::img::CDImage Image(512, 512, 8);
Image.fill(1);
Image.fillMargin(0);
//Image.mirrorMargin();           // Pads image margin using mirroring
```

Images, contd.

```
#include <MDSTk/ Image/mdsImage.h>
```

Image I

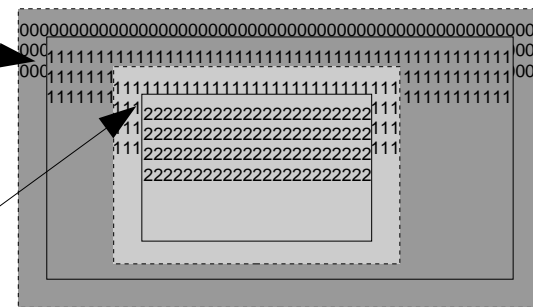
- Image size and margin
- Pointer to the data

Image G (sub-image of the image I)

- Image size
- Margin derived from I
- Pointer to the data

Image data

- Number of references



```
mds::img::CDImage I(500, 350, 8);  
I.fillMargin(0);  
I.fill(1);  
mds::img::CDImage G(I, 100, 80, 300, 150, mds::REFERENCE);  
G.fill(2);
```

Images, contd.

```
#include <MDSTk/Image/mdsImage.h>
```

```
// Create and initialize a new image (margin = 0)
```

```
mds::img::CFImage Image(10, 10);
```

```
Image.fill(0.0f);
```

```
// Another way of the initialization
```

```
for( mds::tSize j = 0; j < Image.getYSize(); ++j )
```

```
    for( mds::tSize i = 0; i < Image.getXSize(); ++i )
```

```
    {
```

```
        Image(i,j) = 0.0f; // Method get(i,j) also returns reference!
```

```
//        Image.set(i,j 0.0f);
```

```
    }
```

```
// Iterators
```

```
mds::img::CFImage::tIterator it, itEnd = Image.getEnd();
```

```
for( it = Image.getBegin(); it != itEnd; ++it ) *it = 0.0f;
```


Images, contd.

```
#include <MDSTk/Image/ImageFunctions.h>
```

- Functions
 - `mds::img::getMin(...)`, `mds::img::getMean(...)`, ...
- Conversion functions
 - `mds::img::convert(...)`, `mds::img::imag(...)`, etc.
- Others
 - `mds::img::fft(...)`, `mds::img::ifft(...)`

Images, contd.

```
#include <MDSTk/Image/mdsImage.h>
#include <MDSTk/Image/mdsImageFunctions.h>
#include <MDSTk/Image/mdsImageFFT.h>

// Initialize the image
mds::img::CImage8 Image(256, 256);
Image.fill(0);
mds::img::CImage8 Rect(Image, 64, 64, 128, 128, mds::REFERENCE);
Rect.fill(255);

// Compute its FFT
mds::img::CImageComplex Result(256, 256);
if( !mds::img::fft(Image, Result) ) return false;
mds::img::fftShift(Result);

// Get the abs image
mds::img::CImage8 Image8(256, 256);
mds::img::abs(Result, Image8);
```

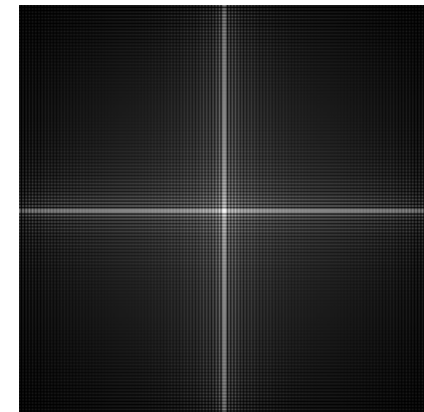
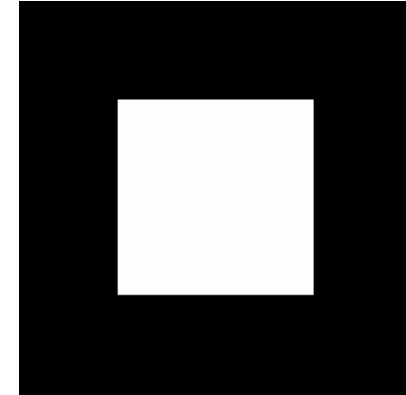


Image filtering

```
#include <MDSTk/Image/mdsImageFilter.h>
```

- Class template `mds::img::CImageFilter<I,Id,N>` specialized for concrete filter types.
 - `I` ... image type.
 - `Id` ... filter type (`IF_SOBEL_X`, `IF_GAUSSIAN`, `IF_MEDIAN`, `IF_CONVOLUTION`, etc.).
 - `N` ... filter response normalization (`IFN_ABS`, `IFN_MEAN`, `IFN_CONV`, etc.).
- Most of the filters are function objects (`functors`).

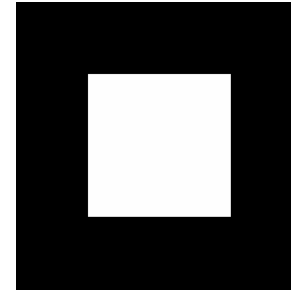
```
bool operator()(const I& SrcImage, I& DstImage);
```

Image filtering, contd.

```
#include <MDSTk/Image/mdsImage.h>
#include <MDSTk/Image/mdsImageFilters.h>
```

```
// Initialize the image
```

```
typedef mds::img::CImage8 tMyImage;
tMyImage Image(256, 256, 3);
Image.fill(0);
tMyImage Rect(Image, 64, 64, 128, 128, mds::REFERENCE);
Rect.fill(255);
```



```
// Sobel operator, version 1
```

```
tMyImage Result1(256, 256, 3);
mds::img::CImageFilter<tMyImage, IF_SOBEL_X, IFN_ABS> Sobel1;
if( !Sobel1(Image, Result1) ) { ... }
```

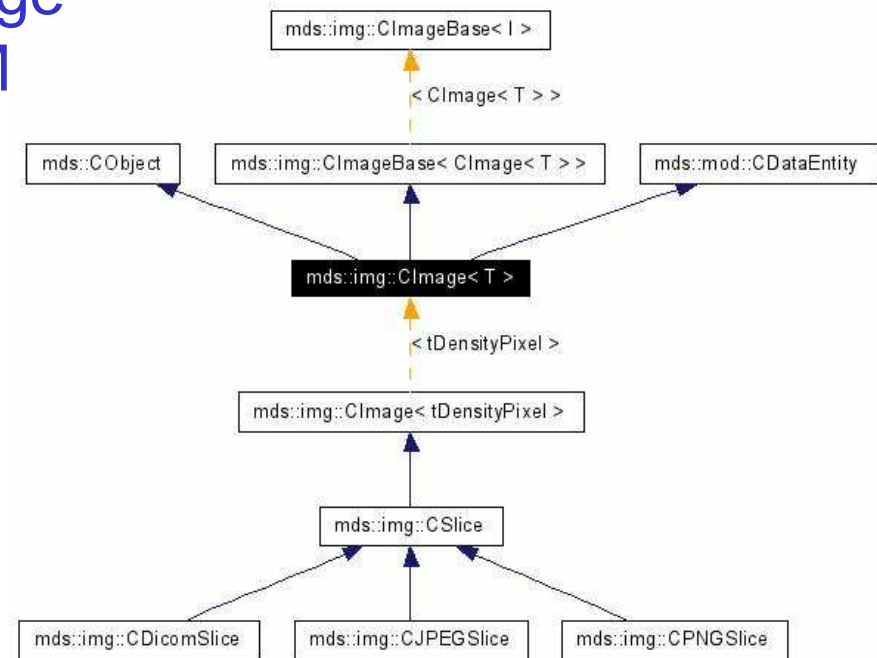


```
// Sobel operator, version 1
```

```
tMyImage Result2(256, 256, 3);
mds::img::CImageFilter<tMyImage, IF_SOBEL_X, IFN_MEAN> Sobel2;
if( !Sobel2(Image, Result2) ) { ... }
```

Slices

- Slice is a medical/density image originally stored in the DICOM format.
- Pixel values in the range $\langle 0..4095 \rangle$.
- Not only the image data.
- Slice contains additional information on the imaging process (CT/MRI):
 - Axis orientation, resolution, etc.
- and patient info (name, ...).



Slices, contd.

```
#include <MDSTk/Module/mdsChannel.h>
#include <MDSTk/Image/mdsSlice.h>
#include <MDSTk/ImageIO/mdsDicomSlice.h>
#include <MDSTk/ImageIO/mdsJPEGSlice.h>

// Open file
mds::mod::CFileChannel File(mds::mod::CH_IN, "filename");
if( !File.connect() ) return false;

// Load DICOM slice from the file/channel
mds::img::CDicomSlice Slice;
Slice.loadDicom(&File);

// Load JPEG image (implicit conversion of any JPEG image to the
// grayscale density image.
mds::img::CJPEGSlice Slice;
Slice.loadJPEG(&File);
```

Logging

```
#include <MDSTk/Base/mdsGlobalLog.h>
```

- Class `mds::base::CLog` and `mds::base::CLogChannel`
- Types of logging channels:
 - `stderr` ... usually writes to screen.
 - `file`
- **Global log** accessible via predefined macros.
- Initialization macros (creation and registration of channels for the global log).

```
MDS_LOG_INIT_STDERR;
```

```
MDS_LOG_INIT_FILE(Filename);
```

- **Logging macros**

```
MDS_LOG("Stream" << "like" << "expression" << std::endl);
```

```
MDS_LOG_CERR("Stream" << "like" << "expression" << std::endl);
```

```
MDS_LOG_TIME("Description");
```

Logging, contd.

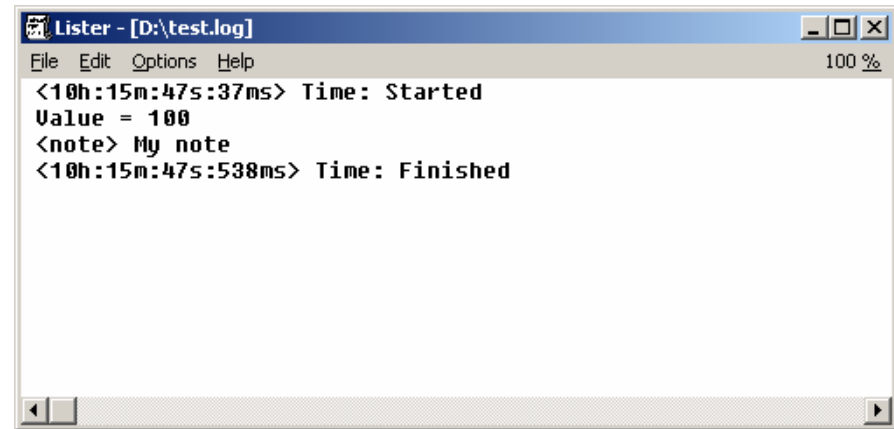
```
#include <MDSTk/Base/mdsSetup.h>
#include <MDSTk/Base/mdsGlobalLog.h>

int main(int argc, char *argv[])
{
    // Init global log
    MDS_LOG_INIT_FILE("test.log");

    int iValue = 100;

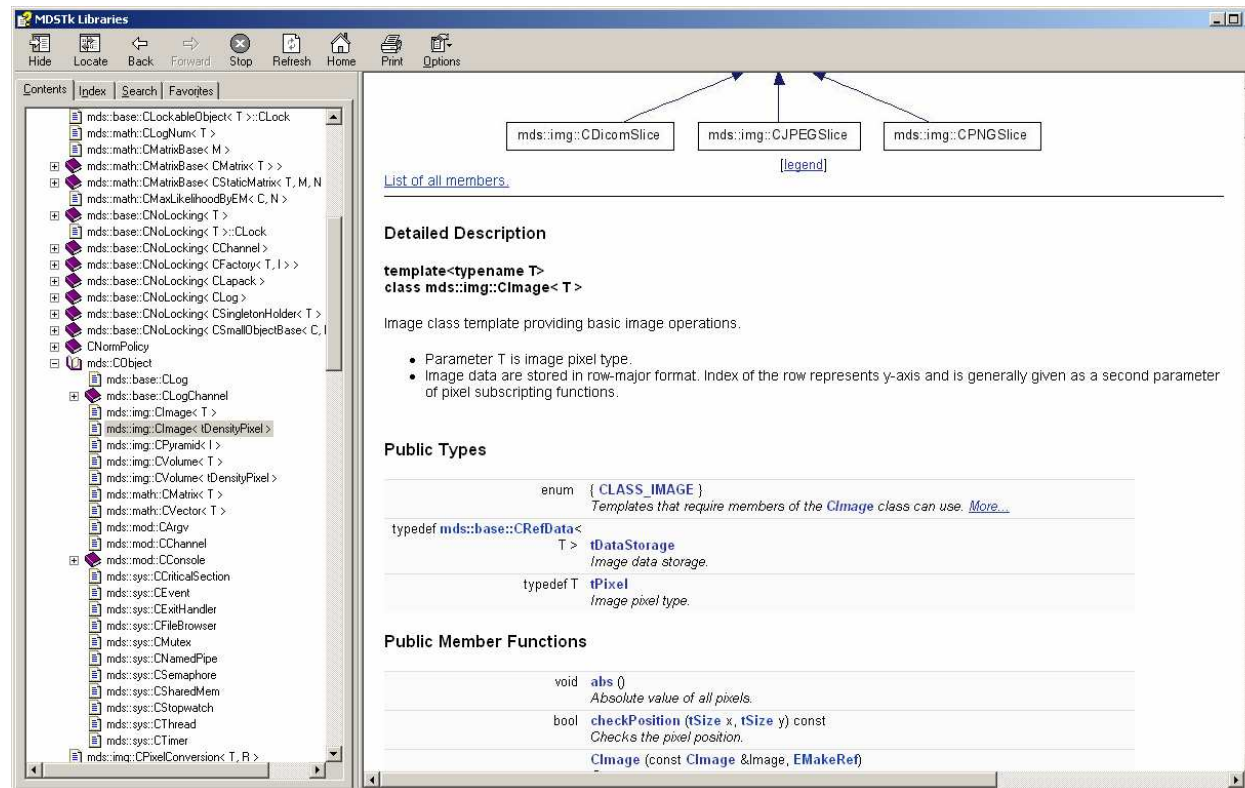
    // Logging
    MDS_LOG_TIME("Started");
    MDS_LOG("Value = " << iValue << std::endl);
    MDS_LOG_NOTE("My note");
    MDS_LOG_TIME("Finished");

    return 0;
}
```



Documentation

- Source code documentation generated using the **Doxygen** system.



- see the '**MDSTk/doc/doxygen**' directory ...