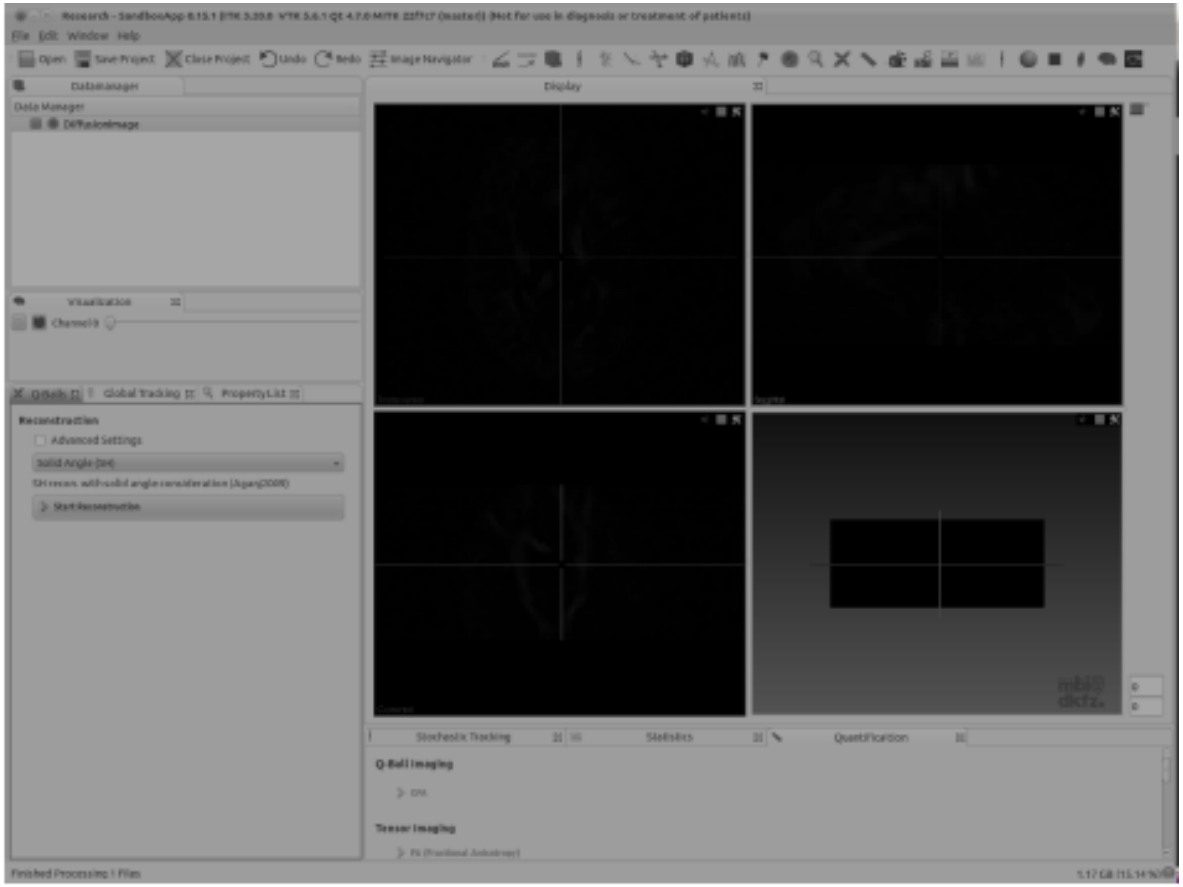
03/09/14

Using QThread in MITK (2)

Andreas Fetzer

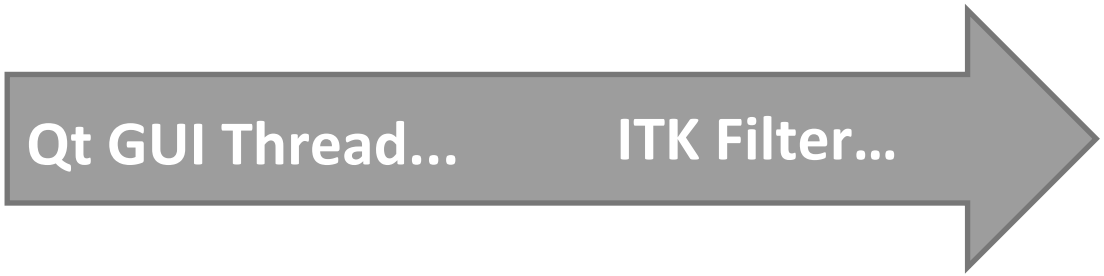
Motivation



The problem

Qt (Main) GUI Thread = the one in which your application runs

The problem



Now start intensive computation…

The problem

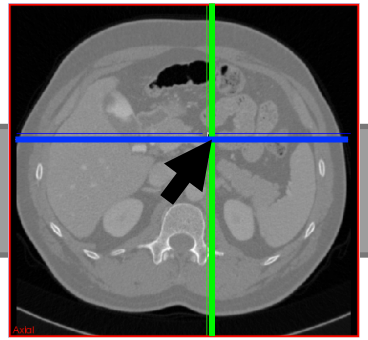
Thread is blocked during computation!

The problem

Now start intensive computation… = no GUI interaction!

The problem

Qt GUI Thread... ITK Filter...

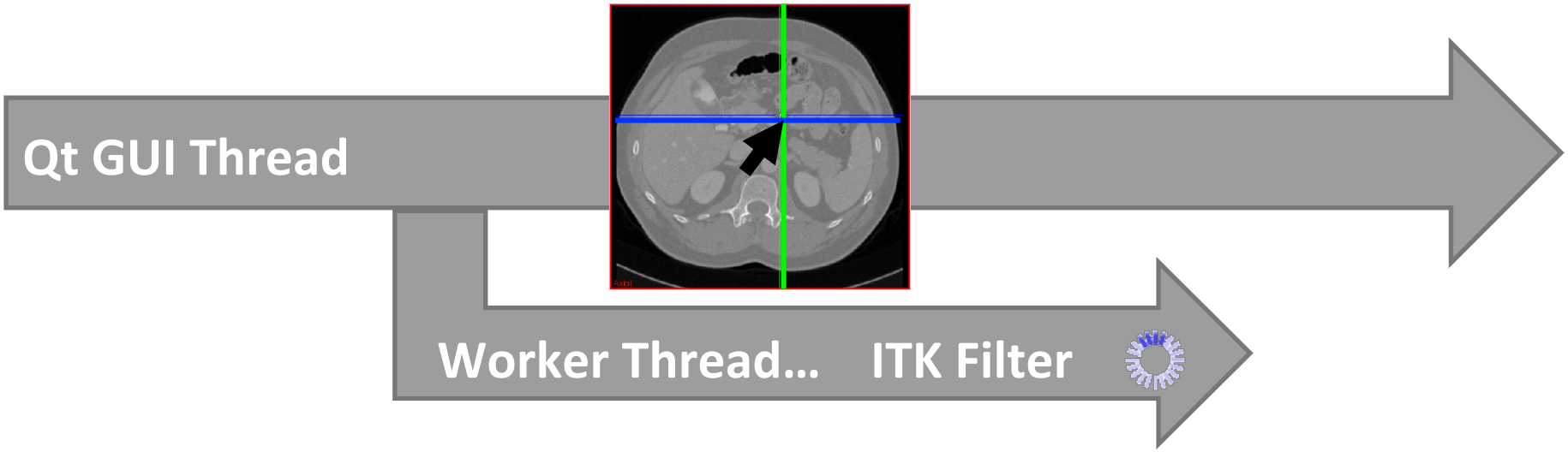


Until computation is finished!

The solution

Again our GUI thread

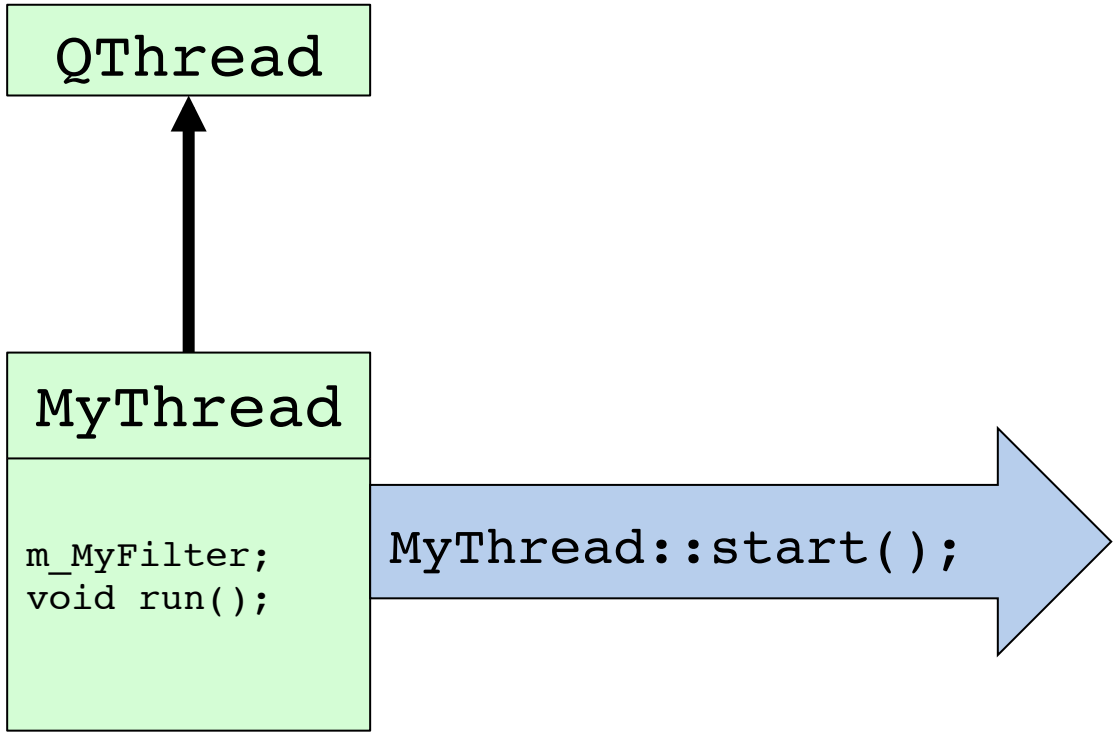
The solution



Simply run computation in a so called “worker thread”

Short flashback (2011-03-09 by Peter Neher)

- Using a QThread to execute filter or operation



Another method: QtConcurrent

- QtConcurrent is a namespace that provides a high-level API for writing multithreaded programs
- Starting a thread:
  `QFuture QtConcurrent::run(Function` *`function, …`*`)`
- If data for processing can be made available within a container, `QtConcurrent` can distribute the processing on all available cores
- Returns `QFuture`

QFuture

- Represents the result of an asynchronous computation
- Helps to synchronize threads and the respective results
- Offers ways to interact with running threads
  - `cancel(), setPaused()`
  - `resume()`
- Retrieve progress information
  - `progressValue()`
  - `waitForFinished()`

Blocks the calling thread!

- Retrieve state information
  - `isRunning(), isCanceled(), …`

QFutureWatcher

- If you don't want to wait for finished ;)
- For monitoring a QFuture using signals and slots
- Convenience methods for accessing the one's of `QFuture` (`isRunning()`, `waitForFinished()`,...)
- `cancel()`, `pause()` are available as slots

Example


```
void myFuncForThread()
{
  // Do something
}

void anotherFunction ()
{
  QFuture<void> future =
                QtConcurrent::run(myFuncForThread());
  future.waitForFinished();
}
```

Example – don't wait for finished


```
void myFuncForThread()
{
  // Do something
}

void OnFinished()
{
}

void anotherFunction ()
{
  QFutureWatcher<void> myWatcher;
  connect(&myWatcher, SIGNAL(finished()), this,
          SLOT(onFinished()));

  QFuture<void> future =
                QtConcurrent::run(myFuncForThread());

  myWatcher.setFuture(future);
}
```

Example – now with parameter


```
void myFuncForThread(int arg1)
{
  // Do something
}

void OnFinished()
{
}

void anotherFunction ()
{
  QFutureWatcher<void> myWatcher;
  connect(&myWatcher, SIGNAL(finished()), this,
          SLOT(onFinished()));

  int value(20);
  QFuture<void> future =
                QtConcurrent::run(myFuncForThread(), value);

  myWatcher.setFuture(future);
}
```

Example – now with return value


```
double myFuncForThread(int arg1)
{
  // Do something
}

void OnFinished()
{
  QFutureWatcher<double>* watcher =
       qobject_cast<QFutureWatcher<double>*>(QObject::sender());
  double result = watcher.result(); //also future.result() possible
}


void anotherFunction ()
{
  QFutureWatcher<double> myWatcher;
  connect(&myWatcher, SIGNAL(finished()), this,
          SLOT(onFinished()));

  int value(20);
  QFuture<double> future =
                QtConcurrent::run(myFuncForThread(), value);

  myWatcher.setFuture(future);
}
```

CAVE!

- **Thread safety**
  - ✓Qt signal/slots
  - – ITK events
- **Handle exceptions**
  - Take care a exception is actually thrown!
  - Then throwing it from one thread to another is possible using Qt means (`QtConcurrent::Exceptions`)
  - isCanceled can help to check whether a thead was aborted
- **::run() does not necessary start the thread**
  → `QtConcurrent` uses global thread pool!
  → If no resource is available start will be delayed
- **Not all QFutures can be cancelled**

References

- http://qt-project.org/doc/qt-4.8/thread-basics.html
- http://qt-project.org/doc/qt-4.8/qtconcurrent.html
- http://qt-project.org/doc/qt-4.8/qfuture.html
- http://qt-project.org/doc/qt-4.8/qfuturewatcher.html
- http://qt-project.org/doc/qt-4.8/qtconcurrent-exception.html

- QtThread (1), Peter Neher: http://www.mitk.org/BugSquashingSeminars?action=AttachFile&do=view&target=Using_QThread.pdf

dkfz. GERMAN CANCER RESEARCH CENTER IN THE HELMHOLTZ ASSOCIATION

50 Years – Research for
A Life Without Cancer