Itk4: Functors and FunctorImageFilters

Tobias Norajitra

dkfz.
GERMAN
CANCER RESEARCH CENTER
IN THE HELMHOLTZ ASSOCIATION

Functors

- Functors = Function Objects

```
class MyOwnFunctor {
  public:
  MyOwnFunctor(int someAdditionalFunctionParameter) : m_SomeAdditionalFunctionParameter(n) {}
  int operator()(int inputParam1, int inputParam2) { //do some logic and return some result; }
private:
  int m_SomeAdditionalFunctionParameter;
};

int ApplySomeLogicOnIntegers(int param1, int param2, MyOwnFunctor functor)
{
  return functor(param1, param2);
}

template <typename FuncType>
int ApplySomeLogicOnIntegers(int param1, int param2, FuncType functor)
{
  return functor(param1, param2);
}
```

- Handled like function call thanks to overloaded operator()

- Provides flexible functionality at runtime

ITKv4 Functors I

- Part of Namespace itk::functor

```
namespace itk {

  typedef double valueType;
  typedef unsigned char outputType;
  typedef Functor::BinaryThreshold< valueType, outputType > BinaryThresholdFunctorType;

}
```

- Various input cardinalities

```
namespace itk {

  typedef unsigned short valueType;
  typedef Functor::Add2< valueType, valueType, valueType > AddTwoValuesFunctorType;

}
```

ITKv4 Functors II

- Many different functors available in itk:
  - Simple Arithmetics: Add, AddConstant, DivideByConstant, Acos, AND, …
  - Complex functions: EdgePotential, GradientMagnitude, IntensityWindowingTransform, …
  - Type Conversions: Cast, VectorCast, JoinFunctor, …
  - Accessors: AccessorFunctor (wrap itk::Accessor classes), LabelObjectAccessors

itkFunctorImageFilters I

- Voxelwise functor invocation on images

```
namespace itk {

  ...

  typedef unsigned short constantType;
  typedef itk::Image< unsigned short, 3 > ImageType;
  typedef itk::Functor::AddConstantTo< constantType > FunctorType;
  typedef itk::UnaryFunctorImageFilter< ImageType, ImageType, FunctorType > FilterType;

  FunctorType functor;
  functor.SetConstant( 5 );

  FilterType::Pointer filter = FilterType::New();
  filter->SetInput( input1 );
  filter->SetFunctor( functor );
  filter->Update();

  ImageType::Pointer output = filter->GetOutput();

  ...

}
```

- Note: FunctorImageFilters derive from itk::InPlaceImageFilter (input overwriting)

itkFunctorImageFilters II

- Different itkFunctorImageFilters for different functor cardinalities
  - itk(GPU)UnaryFunctorImageFilter
  - itkBinaryFunctorImageFilter
  - itkTernaryFunctorImageFilter
  - itkNaryFunctorImageFilter

itkFunctorImageFilters: Example

- BinaryFunctorImageFilter Example:

```
namespace itk {

  ...

  typedef unsigned short MultiplierType;
  typedef itk::Image< unsigned short, 3 > ImageType;
  typedef itk::Functor::AddMultiplied< MultiplierType > AddMultipliedFunctorType;
  typedef itk::BinaryFunctorImageFilter< ImageType, ImageType, ImageType, AddMultipliedFunctorType > FilterType;

  AddMultipliedFunctorType addMultFunctor;
  addMultFunctor.SetMultiplier( 3 );
  FilterType::Pointer filter = FilterType::New();
  filter->SetInput( 0, input1 );
  filter->SetInput( 1, input2 );
  filter->SetFunctor( addMultFunctor );
  filter->Update();

  ImageType::Pointer output = filter->GetOutput();

  ...

}
```

Thanks for the attention.