void SSETRegistration::Run()

{

amyVariable v1=this->GetStack()->Pop();

FETCH(amySeriesSet,set,v1);

vector<amyCube::Keeper> PreventFromReleasing;

amyVariable ret=amyVariable::New(VTYPE\_SERIES\_SET,"Ret");//没设定大小

FETCH(amySeriesSet,retsset,ret);

// retsset.SetSSET(set);//设定大小,有用吗?

/////////ARGS/////////////////////////////////////////////////

int FixedImageSmoothingFactor=0;

int MovingImageSmoothingFactor=0;

int HistogramBins=30; //////FFFFF

int SpatialSamples=10000.0;

int Iterations=200;

int TranslationScale=100.0 ; //////FFFFFF

string args="";

args+="Smoothing level for fixed image:%d\n";

args+="Smoothing level for Moving image:%d\n";

args+="Histogram Bins:%d\n";

args+="Spatial Samples:%d\n";

args+="Iterations:%d\n";

args+="Translation scalings:%d\n";

amyUtil::scanf(args.c\_str(),&FixedImageSmoothingFactor,&MovingImageSmoothingFactor,

&HistogramBins,&SpatialSamples,

&Iterations,&TranslationScale);

/////////ARGS/////////////////////////////////////////////////

for(int i=0;i<set.seriesarr.size();i++)//

{

amyCube::Keeper c1=amySeriesManager::GetInstance()->GetCube(set.seriesarr[0]);//0是固定图像

amyCube::Keeper c2=amySeriesManager::GetInstance()->GetCube(set.seriesarr[i]);

PreventFromReleasing.push\_back(c2);

amyCube::tCube::Pointer pcube=PreprocessCube(c2->obj);

amyCube::tCube::Pointer o1=c1->obj;

amyCube::tCube::Pointer o2=c2->obj;

typedef itk::Image<float,3> CubeType;

//typedef CubeType FixedImageType;

//typedef CubeType MovingImageType;

typedef itk::CastImageFilter<amyCube::tCube,CubeType> CastType;

CastType::Pointer Cast1=CastType::New();

CastType::Pointer Cast2=CastType::New();

Cast1->SetInput(o1);

Cast2->SetInput(o2);

Cast1->Update();

Cast2->Update();

CubeType::Pointer fo1=Cast1->GetOutput();

CubeType::Pointer fo2=Cast2->GetOutput();

CubeType::Pointer FImage=fo1;

CubeType::Pointer MImage=fo2;

/////////ARGS/////////////////////////////////////////////////

//移到上面

/////////ARGS/////////////////////////////////////////////////

const unsigned int ImageDimension = 3;

typedef float T1;

typedef T1 FixedPixelType; // ##

typedef itk::Image<FixedPixelType, ImageDimension> FixedImageType; // ##

typedef itk::OrientImageFilter<FixedImageType, FixedImageType> FixedOrientFilterType; // ##

typedef float T2;

typedef T2 MovingPixelType; // ##

typedef itk::Image<MovingPixelType, ImageDimension> MovingImageType; // ##

typedef itk::OrientImageFilter<MovingImageType, MovingImageType> MovingOrientFilterType; // ##

typedef itk::MattesMutualInformationImageToImageMetric<FixedImageType, MovingImageType> MetricType; // ##

typedef itk::RegularStepGradientDescentOptimizer OptimizerType;

typedef itk::LinearInterpolateImageFunction<MovingImageType, double> InterpolatorType; // ##

typedef itk::ImageRegistrationMethod<FixedImageType, MovingImageType> RegistrationType; // ##

typedef itk::AffineTransform<double> TransformType;

typedef OptimizerType::ScalesType OptimizerScalesType;

typedef itk::ResampleImageFilter<MovingImageType, MovingImageType> ResampleType; // ##

typedef itk::LinearInterpolateImageFunction<MovingImageType, double> ResampleInterpolatorType; // ## // ##

typedef itk::ContinuousIndex<double, 3> ContinuousIndexType;

FixedOrientFilterType::Pointer orientFixed = FixedOrientFilterType::New(); // ##

orientFixed->UseImageDirectionOn();

orientFixed->SetDesiredCoordinateOrientationToAxial();

if( FixedImageSmoothingFactor != 0 ) ////FFFFF1

{

typedef itk::BinomialBlurImageFilter<FixedImageType, FixedImageType> BinomialFixedType;

BinomialFixedType::Pointer BinomialFixed = BinomialFixedType::New();

BinomialFixed->SetInput( FImage );

BinomialFixed->SetRepetitions( FixedImageSmoothingFactor \* 2);

//itk::PluginFilterWatcher watchfilter(BinomialFixed, "Binomial Filter Fixed", CLPProcessInformation, 1.0 / 5.0,

// 1.0 / 5.0);

BinomialFixed->Update();

orientFixed->SetInput(BinomialFixed->GetOutput() );

}

else

{

orientFixed->SetInput(FImage );

}

orientFixed->Update();

MovingOrientFilterType::Pointer orientMoving = MovingOrientFilterType::New(); // ##

orientMoving->UseImageDirectionOn();

orientMoving->SetDesiredCoordinateOrientationToAxial();

///FFFFFF

if( MovingImageSmoothingFactor != 0 )

{

typedef itk::BinomialBlurImageFilter<MovingImageType, MovingImageType> BinomialMovingType;

BinomialMovingType::Pointer BinomialMoving = BinomialMovingType::New();

BinomialMoving->SetInput( MImage );

BinomialMoving->SetRepetitions( MovingImageSmoothingFactor \* 2);

//itk::PluginFilterWatcher watchfilter(BinomialMoving, "Binomial Filter Moving", CLPProcessInformation, 1.0 / 5.0,

// 3.0 / 5.0);

BinomialMoving->Update();

orientMoving->SetInput(BinomialMoving->GetOutput() );

}

else

{

orientMoving->SetInput(MImage );

}

orientMoving->Update();

OptimizerType::Pointer optimizer = OptimizerType::New();

optimizer->SetNumberOfIterations( Iterations );

optimizer->SetMinimumStepLength( .0005 );

optimizer->SetMaximumStepLength( 10.0 );

optimizer->SetMinimize(true);

TransformType::Pointer transform = TransformType::New();

OptimizerScalesType scales( transform->GetNumberOfParameters() );

scales.Fill( 1.0 );

for( unsigned j = 9; j < 12; j++ )

{

scales[j] = 1.0 / vnl\_math\_sqr(TranslationScale);

}

optimizer->SetScales( scales );

TransformType::InputPointType centerFixed;

FixedImageType::RegionType::SizeType sizeFixed =

orientFixed->GetOutput()->GetLargestPossibleRegion().GetSize();

ContinuousIndexType indexFixed;

for( unsigned j = 0; j < 3; j++ )

{

indexFixed[j] = (sizeFixed[j] - 1) / 2.0;

}

orientFixed->GetOutput()->TransformContinuousIndexToPhysicalPoint( indexFixed, centerFixed );

TransformType::InputPointType centerMoving;

MovingImageType::RegionType::SizeType sizeMoving =

orientMoving->GetOutput()->GetLargestPossibleRegion().GetSize();

// Find the center

ContinuousIndexType indexMoving;

for( unsigned j = 0; j < 3; j++ )

{

indexMoving[j] = (sizeMoving[j] - 1) / 2.0;

}

orientMoving->GetOutput()->TransformContinuousIndexToPhysicalPoint( indexMoving, centerMoving );

transform->SetCenter( centerFixed );

transform->Translate(centerMoving - centerFixed);

//////读文件

//if( InitialTransform != ""

// && initialTransform->GetTransformList()->size() != 0 )

//{

// TransformReaderType::TransformType::Pointer initial

// = \*(initialTransform->GetTransformList()->begin() );

// // most likely, the transform coming in is a subclass of

// // MatrixOffsetTransformBase

// typedef itk::MatrixOffsetTransformBase<double, 3, 3> DoubleMatrixOffsetType;

// typedef itk::MatrixOffsetTransformBase<float, 3, 3> FloatMatrixOffsetType;

// DoubleMatrixOffsetType::Pointer da

// = dynamic\_cast<DoubleMatrixOffsetType \*>(initial.GetPointer() );

// FloatMatrixOffsetType::Pointer fa

// = dynamic\_cast<FloatMatrixOffsetType \*>(initial.GetPointer() );

// if( da )

// {

// transform->SetMatrix( da->GetMatrix() );

// transform->SetOffset( da->GetOffset() );

// }

// else if( fa )

// {

// vnl\_matrix<double> t(3, 3);

// for( int i = 0; i < 3; ++i )

// {

// for( int j = 0; j < 3; ++j )

// {

// t.put(i, j, fa->GetMatrix().GetVnlMatrix().get(i, j) );

// }

// }

// transform->SetMatrix( t );

// transform->SetOffset( fa->GetOffset() );

// }

// else

// {

// std::cout << "Initial transform is an unsupported type." << std::endl;

// }

// std::cout << "Initial transform: "; transform->Print( std::cout );

//}

MetricType::Pointer metric = MetricType::New();

metric->SetNumberOfHistogramBins( HistogramBins );

metric->SetNumberOfSpatialSamples( SpatialSamples );

metric->ReinitializeSeed(123);

InterpolatorType::Pointer interpolator = InterpolatorType::New();

// Set up the registration

//

RegistrationType::Pointer registration = RegistrationType::New();

registration->SetTransform( transform );

registration->SetInitialTransformParameters( transform->GetParameters() );

registration->SetMetric( metric );

registration->SetOptimizer( optimizer );

registration->SetInterpolator( interpolator );

registration->SetFixedImage( orientFixed->GetOutput() );

registration->SetMovingImage( orientMoving->GetOutput() );

try

{

registration->Update();

std::cout << "Optimizer stop condition: "

<< registration->GetOptimizer()->GetStopConditionDescription()

<< std::endl;

}

catch( itk::ExceptionObject & err )

{

std::cerr << "ExceptionObject caught !" << std::endl;

std::cerr << err << std::endl;

return;

}

transform->SetParameters( registration->GetLastTransformParameters() );

ResampleType::Pointer resample = ResampleType::New();

ResampleInterpolatorType::Pointer Interpolator = ResampleInterpolatorType::New();

//itk::PluginFilterWatcher watchResample(resample,

// "Resample",

// CLPProcessInformation,

// 1.0 / 5.0, 4.0 / 5.0);

resample->SetInput( MImage );

resample->SetTransform( transform );

resample->SetInterpolator( Interpolator );

MovingImageType::Pointer fixedInformation = MovingImageType::New();

fixedInformation->CopyInformation( FImage);

resample->SetOutputParametersFromImage( fixedInformation );

resample->Update();

typedef itk::CastImageFilter< FixedImageType, amyCube::tCube > CastFilterType;

CastFilterType::Pointer caster = CastFilterType::New();

caster->SetInput( resample->GetOutput() );

caster->Update();

//自己添加

amyVariable z1=amyVariable::New(VTYPE\_CUBE,"Ret");//消耗内存么？？

amyCube::Keeper retcube=z1.Get<amyCube>();

retcube->SetObj(caster->GetOutput());

//amyCube::tCube::Pointer pointer=retcube->obj;

//pointer.GetPointer()->

//retcube->SetSeries(set.seriesarr[i]);

//retsset.cubearr.push\_back(retcube);

//amySeries::Keeper retseries;

//retsset.seriesarr.push\_back(retcube->series);

//

vector<string> names;

vector<QString> qnames;

QString savepath="C:\\Users\\xuxieping\\Desktop\\ret";

QString nameformat="ret";

nameformat+="\_";

nameformat+="%03d";

nameformat+="\_";

nameformat+="%03d";

QString str;

for(int j=0;j<set.seriesarr[i]->slicearr.size();j++)//???

{

str.sprintf(nameformat.toLocal8Bit().data(),i,j);

qnames.push\_back(savepath+"/"+str);

names.push\_back(qnames[j].toLocal8Bit().data());

}

typedef itk::Image<short,2> OutputImageType;

typedef itk::ImageSeriesWriter<amyCube::tCube,OutputImageType> WriterType;

WriterType::Pointer writer = WriterType::New();

typedef itk::GDCMImageIO ImageIOType;

writer->SetImageIO(ImageIOType::New());

writer->SetInput(retcube->obj );

itk::GDCMImageIO::Pointer io=itk::GDCMImageIO::New();

writer->SetMetaDataDictionary(retcube->obj->GetMetaDataDictionary());//错误

writer->SetFileNames(names);//重要，一定要写？？

try

{

writer->Update();

}

catch( itk::ExceptionObject & excp )

{

std::cerr << "ExceptionObject caught !" << std::endl;

// std::cerr << err << std::endl;

return;//return 0;

}

//writer->Update();//有问题！！！

//writer->

//retsset.seriesarr.push\_back(z1);

}

this->GetStack()->Push(ret);

}