Teknisk dokumentation MCIV

Anders Eklund

Version 1.0

Status

Granskad		
Godkänd		

PROJEKTIDENTITET

MCIV 2006 VT

Linköpings Tekniska Högskola, CVL

Namn	Ansvar	Telefon	E-post
Fredrik Larsson	Projektledare (PL)		frela070@student.liu.se
Anders Eklund	Dokumentansvarig (DOK)		andek034@student.liu.se
Lars Arvidsson	Designansvarig (DES)		larar125@student.liu.se
Nils Ingemars	Testansvarig (TST)		nilin308@student.liu.se
Linus Gustafsson	Kvalitetsansvarig (KVA)		lingu023@student.liu.se

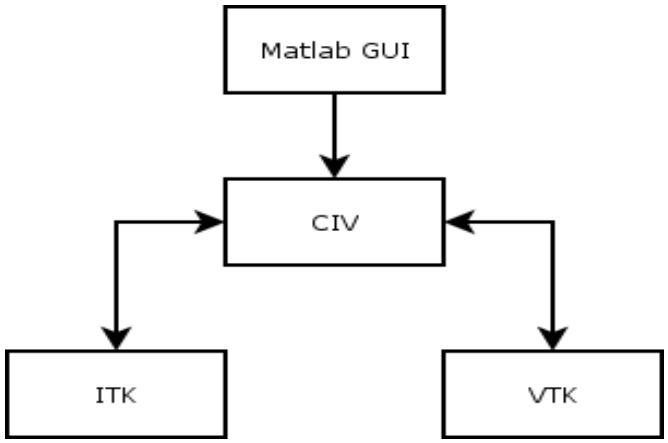
Kund: Context Vision, 582 23 LINKÖPING,
tel. 013-35 85 50, fax: 013-10 42 82, info@contextvision.se
Kontaktperson hos kund: Hagen Spies, tel. 013 – 35 85 64, hagen.spies@contextvision.se
Kursansvarig: Klas Nordberg, tel. 013-28 16 34, klas@isy.liu.se
Handledare: Johan Wiklund, tel. 013-281359, jowi@isy.liu.se, Anders Moe, tel. 013-28 26 39, moe@isy.liu.se

Innehåll

Dokumenthistorik

version	datum	Utförda förändringar	utförda av	granskad
0.1	2006-05-12	Första utkastet	AE,NI,LA,LG	Alla
1.0	2006-05-16	Slutgiltig version	LG, NI, AE	Alla

1 Översikt av systemet



Figur 1. Denna bild visar en översikt av systemet.

1.1 Grov beskrivning av systemet

Ett Matlab-GUI som anropar C++-funktioner som kompilerats till mex-filer, vilka antingen utför segmenteringar med ITK eller visualiseringar med VTK.

Programmet startas genom att i kommandofönstret i Matlab skriva kommandot *MCIV*. Detta startar upp GUI't. Via GUI't hanteras inläsning och nedsparning av bilder från/på fil. Inställning av de aktuella parametrarna för segmentering och visualisering hanteras också via GUI't. Då segmentering/visualisering önskas utföras anropas korresponderande mex-fil med inbild/volym och berörda parametrar. Segmenterings- och visualiseringsrutinerna är implementerade i C++ och sedan omgjorda till mex-filer för att kunna anropas från Matlab, *t.ex. mciv_MPR(inputimage).*

1.2 Beroenden till andra system

MS Windows XP

Matlab v 7.1

ITK v 2.4

VTK v 5.0

MS Visual Studio .NET 2003 (för kompilering)

1.3 Ingående delsystem

ITK – programbibliotek med bildbehandlingsfunktioner.

VTK – programbibliotek med visualiseringsfunktioner.

CIV – kopplingen mellan ITK och VTK.

Matlab GUI – användargränssnitt i Matlab.

2 Matlab GUI

2.1 Inledning

GUI't är implementerat med hjälp av den inbyggda GUI-guiden i Matlab. Man lägger in rutor och knappar som alla har varsin callback-funktion som talar om vad som händer när man ändrar värdet i en ruta eller trycker på en knapp. Om man vill ändra vad som händer för en viss knapp eller ruta så är det bara att söka upp rutans eller knappens funktionsnamn i MCIV.m.

2.2 Beskrivning av callbackfunktioner

Nedan följer en kort beskrivning av vad alla callbackfunktioner i GUI't gör.

about_Callback	Visar fönstret med "About MCIV" då man väljer "About MCIV" i menyn.
addseedbutton_Callback	Lägger till ett frö i listan med frön.
fastmarchingpipeline_Callback	Visar flödesschemat för fast marching.
geodesicpipeline_Callback	Visar flödesschemat för geodesic active contours.
import_original_Callback	Importerar originalbilden eller originalvolymen till Matlab.
import_segmentation_Callback	Importerar segmenteringsresultatet till Matlab.
import_seeds_button_Callback	Importerar frön som man satte vid visualiseringen.
methodbutton1_Callback	Hanterar vad som händer när man byter segmenteringsmetod eller visualiseringsmetod, samma för methodbutton2- methodbutton4.
nextlayer_Callback	Visar nästa bild av volymen i förvisningsfönstret.
open_Callback	Hanterar inläsning av filer.

parameter1_Callback	Hanterar vad som händer då man fyllt i ett värde i en parameterruta, samma för parameter2-parameter15.
previouslayer_Callback	Visar föregående bild i volymen i förvisningsfönstret.
removeallseeds_button_Callback	Tar bort alla frön i frölistan.
removeseedbutton_Callback	Tar bort det markerade fröet från frölistan.
resetbutton_Callback	Återställer alla värden.
save_segmentation_image_Callback	Sparar den segmenterade bilden, om det är en segmenterad volym så sparas varje snitt som en bild.
seedValue_button_Callback	Ändrar ett värde för ett frö som redan lagts till i listan med frön.
segmentationbutton_Callback	Ändrar läge till segmentering.
start_Callback	Startar olika segmenterings- och visualiseringsrutiner, beroende på vad man valt, med rätt inparametrar.
useoriginal_Callback	Hanterar vad som händer då man väljer att visualisera originalbilden eller originalvolymen.
usesegmented_Callback	Hanterar vad som händer då man väljer att visualisera den segmenterade bilden eller volymen.
usesegmentedoriginal_Callback	Hanterar vad som händer då man väljer att visualisera den segmenterade delen av originalbilden eller originalvolymen.
visualization_help_Callback	Visar fönstret med alla kommandon som man kan använda vid visualiseringen.
visualizationbutton_Callback	Byter läge till visualisering.
watershedpipeline_Callback	Visar flödesschemat för watershed.

2.3 Gränssnitt

De parametrar som behövs i MCIV-funktionerna återfinns i följande Matlab-variabler som endast är tillgängliga för GUI't.

watershed_iterations
watershed_conductance
watershed_time_step
watershed_threshold
watershed_maximum_flood_level
geodesic_iterations
geodesic_conductance
geodesic_timestep
geodesic_sigma
geodesic_alpha
geodesic_beta
geodesic_seeds
geodesic_propagation
geodesic_curvature
geodesic_advection
geodesic_lower_threshold
geodesic_upper_threshold
fast_marching_iterations
fast_marching_conductance
fast_marching_timestep
fast_marching_sigma
fast_marching_alpha
fast_marching_beta
fast_marching_seeds
fast_marching_stop_time
fast_marching_lower_time_threshold
fast_marching_upper_time_threshold
mip_zspacing
mpr_zspacing

slice_zspacing
surface_rendering_zspacing
surface_rendereing_iso_value

3 CIV

CIV är implementerat i C++ och består av en mex-funktion per segmenterings- och visualiseringsmetod. Varje mex-funktion är uppdelad i konvertering från Matlab till ITK/VTK, själva koden för den aktuella metoden och konvertering från ITK till Matlab (om det är en ITK-metod).

Inläsningen av bilder sker via Matlab och dessa skickas sedan tillsammans med de andra parametrarna med i funktionsanropen till de olika rutinerna. Om det är en ITK-metod så får man sedan tillbaka den behandlade bilden i Matlab.

3.1 Mex-filer

Om man vill göra funktioner, skrivna i C++, som ska kunna anropas från Matlab så måste man använda sig av mex-filer. Först måste man inkludera mex.h och sedan alla andra h-filer som behövs. Man behöver även ett skal i vilket man kan skriva sin kod, istället för main så har man mexFunction. Anropsnamnet i Matlab kommer att bli samma som filnamnet på cpp-filen. Alla mex-filer kommer att innehålla nedanstående rader:

```
#include "mex.h"

void mexFunction(int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])

{

//din kod

}
```

nlhs är en parameter som talar om hur många olika parametrar det finns på vänster sida om tilldelningsoperatorn

plhs är en pekare till det som står på vänster sida om tilldelningsoperatorn

nrhs är en parameter som talar om hur många olika parametrar det finns på höger sida om tilldelningsoperatorn

prhs är en pekare till det som står på höger sida om tilldelningsoperatorn

Exempel:

```
>> [a,b] = mex_test_function(c,d,e);
```

I detta fall är nlhs = 2, plhs[0] pekar på a och plhs[1] pekar på b, nrhs = 3, prhs[0] pekar på c, prhs[1] pekar på d och prhs[2] pekar på e.

a,b,c,d,e kan vara tal eller vektorer eller matriser

För att kompilera skriver man kommandona

>>mex –setup

>>mex mex_test_function

För mer detaljerad information om mex-filer hänvisar vi till Mathworks, se referenser.

För mer detaljerad information om kompilering, se stycket 4 Kompileringsinstruktioner.

3.2 Exportera data till Matlab

När man är klar med sina operationer på bilden och vill skicka tillbaka bilden till Matlab så kan man göra det med hjälp av en iterator.

Exempel:

```
mxArray* mvar = mxCreateNumericArray(dim,dims,mxUINT8_CLASS,mxREAL);

unsigned char* data = (unsigned char*)mxGetData(mvar);

typedef itk::ImageRegionConstIterator<OutputImageType> ConstIteratorType;

ConstIteratorType imageout(thresholder->GetOutput(),thresholder->GetOutput()-
>GetRequestedRegion());

imageout.GoToBegin();

for (int i = 0; !imageout.IsAtEnd(); ++i,++imageout)

{

        data[i] = imageout.Get();

}

plhs[0] = mvar;
```

Beroende på vad man vill att bilden i Matlab ska ha för datatyp så kan man använda något annat än mxUINT8_CLASS.

3.3 Importera data från Matlab

Om man skickar med en bild, eller en volym, som inparameter till en funktion så kan man komma åt själva bilden genom mxGetData(prhs[0]) om bilden är den första parametern. Om man vill ha ett visst format på bilden så kan man göra en typomvandling med t.ex. (InternalPixelType*)mxGetData(prhs[0]). Bilden lagras internt som ett enda långt fält istället för som en matris.

Om en parameter är en skalär så får man tag på värdet med mxGetScalar(prhs[0]).

3.4 Funktioner

De mexfunktioner som klassen CIV kommer ha är de nedan listade, alla kommer att ha en bild eller volym som första inparameter.

Namn:	*Kort beskrivning:*
mciv_watershed2D	Utför segmenteringsrutinen watershed på en bild.
mciv_watershed3D	Utför segmenteringsrutinen watershed på en volym.
mciv_geodesic2D	Utför segmenteringsrutinen geodesic active contours på en bild.
mciv_geodesic3D	Utför segmenteringsrutinen geodesic active contours på en volym.
mciv_fastmarching2D	Utför segmenteringsrutinen fast marching på en bild.
mciv_fastmarching3D	Utför segmenteringsrutinen fast marching på en volym.
mciv_slice	Utför visualiseringsmetoden slice.
mciv_mpr	Utför visualiseringsmetoden MPR.
mciv_mip	Utför visualiseringsmetoden MIP.
mciv_surface	Utför visualiseringsmetoden surface rendering.
mciv_mxvtkconvert	Funktionen konverterar en bild från Matlab till en bild som VTK kan arbeta med.

Nedanstående tabell visar hur funktionsparametrar och returvärden med dess datatyper definieras. Funktionsparametrarna och returvärdena är virtuella och utgörs av prhs-respektive plhs-pekarna i mex-funktionerna. Namnen på funktionsparametrarna och returvärdena nedan är nödvändigtvis inte exakt samma som namnen på variablerna i koden.

Namn	Beskrivning	Datatyp

3.4.1 Funktion: mciv_watershed2D

Funktionsspecifika funktionsparametrar:

Iterations	Antalet iterationer som ska utföras av diffusionsfiltret.	Int
Conductance	Inparameter till diffusionsfiltret.	Double
Timestep	Inparameter till diffusionsfiltret, max 0.125 för 2D, 0.0625 för 3D.	Double
Threshold	Tröskel för att tröskla bort grunda regioner för att minska översegmentering, anges i procent, 0.0-1.0, av maxvärdet i höjdbilden.	Double
Maximum Flood Level	Nivån på det maximala watersheddjupet av intresse hos resultatet, anges i procent, 0.0 - 1.0, av maxvärdet i höjdbilden.	Double

Returvärde:

Segmented Image	Segmenteringsresultatet.	mxArray *

Beskrivning av funktionen:

Funktionen utför segmenteringsrutinen watershed på inbilden, segmenteringen utförs med hjälp av ITK. Som output fås en bild med etiketter i varje pixel som talar om vilket område pixeln hör till.

3.4.2 Funktion: mciv_watershed3D

Funktionsspecifika funktionsparametrar:

Iterations	Antalet iterationer som ska utföras av diffusionsfiltret.	Int
Conductance	Inparameter till diffusionsfiltret.	Double
Timestep	Inparameter till diffusionsfiltret, max 0.125 för 2D, 0.0625 för 3D.	Double
Threshold	Tröskel för att tröskla bort grunda regioner för att minska översegmentering, anges i procent, 0.0-1.0, av maxvärdet i höjdbilden.	Double
Maximum Flood Level	Nivån på det maximala watersheddjupet av intresse hos resultatet, anges i procent, 0.0 - 1.0, av maxvärdet i höjdbilden.	Double

Returvärde:

Segmented Image	Segmenteringsresultatet, en tom bild om fel inträffat.	mxArray *

Beskrivning av funktionen:

Funktionen utför segmenteringsrutinen watershed på involymen, segmenteringen utförs med hjälp av ITK. Som output fås en volym med etiketter i varje pixel som talar om vilket område pixeln hör till.

3.4.3 Funktion: mciv_geodesic2D

Funktionsparametrar:

Iterations	Antalet iterationer som ska utföras av diffusionsfiltret.	Int
Conductance	Inparameter till diffusionsfiltret.	Double
Timestep	Inparameter till diffusionsfiltret, max 0.125 för 2D, 0.0625 för 3D.	Double
Sigma	Parameter som bestämmer bredden på det Gaussiska gradientfiltret.	Double
Alpha	Parameter till sigmoidfiltret.	Double
Beta	Parameter till sigmoidfiltret.	Double
Seeds	Punkter där fast marching startar, lista med x-, y- och z-koordinater samt värde för varje punkt.	vector<double>
Propagation	Skalningsparameter för propagering.	Double
Curvature	Skalningsparameter för kurvaturen.	Double
Advection	Skalningsparameter för advektion.	Double
Lower threshold	Anger den nedre gränsen för vad som ska visas.	Double
Upper threshold	Anger den övre gränsen för vad som ska visas.	Double

Returvärde:

Segmented Image	Segmenteringsresultatet.	mxArray

Beskrivning av funktionen:

Funktionen utför segmenteringsrutinen geodesic active contours på inbilden, segmenteringen utförs med hjälp **av** ITK. Som output fås en binärbild.

3.4.4 Funktion: mciv_geodesic3D

Funktionsparametrar:

Iterations	Antalet iterationer som ska utföras av diffusionsfiltret.	Int
Conductance	Inparameter till diffusionsfiltret.	Double
Timestep	Inparameter till diffusionsfiltret, max 0.125 för 2D, 0.0625 för 3D.	Double
Sigma	Parameter som bestämmer bredden på det Gaussiska gradientfiltret.	Double
Alpha	Parameter till sigmoidfiltret.	Double
Beta	Parameter till sigmoidfiltret.	Double
Seeds	Punkter där fast marching startar, lista med x-, y- och z-koordinater samt värde för varje punkt.	vector<double>
Propagation	Skalningsparameter för propagering.	Double
Curvature	Skalningsparameter för kurvaturen.	Double
Advection	Skalningsparameter för advektion.	Double
Lower threshold	Anger den nedre gränsen för vad som ska visas.	Double
Upper threshold	Anger den övre gränsen för vad som ska visas.	Double

Returvärde:

Segmented Image	Segmenteringsresultatet, en tom bild om fel inträffat.	mxArray

Beskrivning av funktionen:

Funktionen utför segmenteringsrutinen geodesic active contours på involymen, segmenteringen utförs med hjälp **av** ITK. Som output fås en binärvolym.

3.4.5 Funktion: mciv_fastmarching2D

Funktionsparametrar:

Iterations	Antalet iterationer som ska utföras av diffusionsfiltret.	Int
Conductance	Inparameter till diffusionsfiltret.	Double
Timestep	Inparameter till diffusionsfiltret, max 0.125 för 2D, 0.0625 för 3D.	Double
Sigma	Parameter som bestämmer bredden på det Gaussiska gradientfiltret.	Double
Alpha	Parameter till sigmoidfiltret.	Double
Beta	Parameter till sigmoidfiltret.	Double
Seeds	Punkter där fast marching startar, lista med x-, y- och z-koordinater, samt värde för varje punkt.	vector<double>
Stoptime	Parameter som anger när fronten som representerar konturen i fast marching ska sluta propagera, bör vara något större än upper time threshold.	Double
Lower Time Threshold	Den lägre tidsgränsen för vad som ska visas.	Double
Upper Time Threshold	Den övre tidsgränsen för vad	Double

	som ska visas.	

Returvärde:

Segmented Image	Segmenteringsresultatet, en tom bild om fel inträffat.	mxArray

Beskrivning av funktionen:

Funktionen utför segmenteringsrutinen fast marching på inbilden, segmenteringen utförs med hjälp av ITK. Som output fås en binärbild.

3.4.6 Funktion: mciv_fastmarching3D

Funktionsparametrar:

Iterations	Antalet iterationer som ska utföras av diffusionsfiltret.	Int
Conductance	Inparameter till diffusionsfiltret.	Double
Timestep	Inparameter till diffusionsfiltret, max 0.125 för 2D, 0.0625 för 3D.	Double
Sigma	Parameter som bestämmer bredden på det Gaussiska gradientfiltret.	Double
Alpha	Parameter till sigmoidfiltret.	Double
Beta	Parameter till sigmoidfiltret.	Double
Seeds	Punkter där fast marching startar, lista med x-, y- och z-koordinater, samt värde för varje punkt.	vector<double>
Stoptime	Parameter som anger när fronten som representerar konturen i fast marching ska sluta propagera, bör vara något större än upper time threshold.	Double
Lower Time Threshold	Den lägre tidsgränsen för vad som ska visas.	Double

Upper Time Threshold	Den övre tidsgränsen för vad som ska visas.	Double

Returvärde:

Segmented Image	Segmenteringsresultatet, en tom bild om fel inträffat.	mxArray

Beskrivning av funktionen:

Funktionen utför segmenteringsrutinen fast marching på involymen, segmenteringen utförs med hjälp av ITK. Som output fås en binärvolym.

3.4.7 Funktion: mciv_slice

Funktionsparametrar:

segmentation	Anger om ursprungsbilden eller segmenteringsresultatet ska visas.	Bool
zspacing	Avståndet mellan snitten.	Double

Returvärde:

visualized_image	En volym som användaren kan rotera och zooma i visualiseringsfönstret.	mxArray

Beskrivning av funktionen:

Funktionen skär ett snitt genom en volym.

3.4.8 Funktion: mciv_mpr

Funktionsparametrar:

segmentation	Anger om ursprungsbilden eller segmenteringsresultatet ska visas.	Bool
zspacing	Avståndet mellan snitten.	Double

Returvärde:

visualized_image	En volym som användaren kan rotera och zooma i visualiseringsfönstret.	mxArray

Beskrivning av funktionen:

Funktionen skär tre ortogonala snitt genom en volym.

3.4.9 Funktion: mciv_mip

Funktionsparametrar:

Zspacing	Avståndet mellan snitten.	Double

Returvärde:

Visualized Image	En volym som användaren kan rotera och zooma i visualiseringsfönstret.	mxArray

Beskrivning av funktionen:

Funktionen läser av de största värdena längs ett antal strålar genom volymen.

3.4.10 Funktion: mciv_surface

Funktionsparametrar

Zspacing	Avståndet mellan snitten.	Double
Isovalue	Tröskelvärde för ytan.	Double

Returvärde:

Visualized Image	En volym som användaren kan rotera och zooma i visualiseringsfönstret.	mxArray

Beskrivning av funktionen:

Funktionen visualiserar en yta av en volym givet ett visst minimalt tröskelvärde.

3.4.11 Funktion: mciv_mxvtkconvert

Funktionsparametrar

mvar	Den inlästa bilden från Matlab.	const mxArray*

Returvärde:

vtkdata	Den konverterade bilden till VTK.	vtkImageData*

Beskrivning av funktionen:

Funktionen konverterar en bild från Matlab till en bild som VTK kan arbeta med.

4 Kompileringsinstruktioner

Starta Matlab och kör:

>>mex -setup

Lokalisera filen "mexopts.bat" i Matlabs katalog i "Documents and settings".

I "mexopts.bat":

1. Lägg till följande rader direkt under "General Parameters" (enligt den filstruktur vi använt):

   set MCIV=C:\MCIV
   set ITKinclude=%MCIV%\ITK\itk-2.4.0\include\InsightToolkit
   set ITKlib=%MCIV%\ITK\itk-2.4.0\lib\InsightToolkit
   set VTKinclude=%MCIV%\VTK\vtk-5.0.0\include\vtk-5.0
   set VTKlib=%MCIV%\VTK\vtk-5.0.0\lib

2. Lägg till följande rad sist i "set INCLUDE="-kommandot:

   ;%VTKinclude%;%ITKinclude%;%ITKinclude%\Algorithms;%ITKinclude%\BasicFilters;%ITKinclude%\Common;%ITKinclude%\Utilities\vxl\vcl;%ITKinclude%\Utilities\vxl\core

3. Lägg till följande rad sist i "set LIB="-kommandot:

   ;%VTKlib%;%ITKlib%

4. Lägg till följande rad sist i "set LINKFLAGS="-kommandot:

   vtkCommon.lib vtkDICOMParser.lib vtkexoIIc.lib vtkexpat.lib vtkFiltering.lib vtkfreetype.lib vtkftgl.lib vtkGenericFiltering.lib vtkGraphics.lib vtkHybrid.lib vtkImaging.lib vtkIO.lib vtkjpeg.lib vtkMPEG2Encode.lib vtkNetCDF.lib vtkpng.lib vtkRendering.lib vtksys.lib vtktiff.lib vtkVolumeRendering.lib vtkWidgets.lib vtkzlib.lib itkCommon.lib itkvnl.lib itkvcl.lib itkvnl_algo.lib ITKniftiio.lib ITKNrrdIO.lib ITKNumerics.lib ITKSpatialObject.lib ITKStatistics.lib itksys.lib itkzlib.lib ITKznz.lib ITKAlgorithms.lib ITKBasicFilters.lib ITKDICOMParser.lib ITKEXPAT.lib ITKFEM.lib itkgdcm.lib ITKIO.lib ITKMetaIO.lib itknetlib.lib itkvnl_inst.lib

I Matlab, kompilera med:
>> mex mciv_mpr.cpp
>> mex mciv_slice.cpp
>> mex mciv_mip.cpp
>> mex mciv_surface.cpp
>> mex mciv_geodesic2D.cpp
>> mex mciv_geodesic3D.cpp
>> mex mciv_fastmarching2D.cpp
>> mex mciv_fastmarching3D.cpp
>> mex mciv_watershed2D.cpp
>> mex mciv_watershed3D.cpp

Det bör nämnas att vi använder statisk länkning för ITK och dynamisk länkning för VTK. Därför måste samtliga VTK dll-filer finnas i samma katalog som MCIV.

Om man vill kompilera ITK/VTK själv:

I sökvägarna ovan finns katalogerna "itk-2.4.0" och "vtk-5.0.0". Dessa motsvarar installationskatalogerna för de färdigkompilerade versionerna av ITK 2.4 och VTK 5.0. För information om hur man kompilerar ITK och VTK hänvisar vi till deras hemsidor, se referenser. Observera att ITK/VTK måste vara kompilerat med samma kompilator som den man ska kompilera mciv-filerna med.

5 Lägga till egna funktioner (ITK)

5.1 Instruktioner för att lägga till egna funktioner till ITK

Det är rekommenderat att man söker upp en h-fil (och dess tillhörande txx, om den finns) som är lik den funktion man tänker konstruera. Då får man en viss känsla av hur koden ska se ut ganska snabbt. De flesta av de vanligaste filtertyperna finns redan implementerade i ITK. Många gånger är de uppdelade i filter och operator, där filtret använder sig av operatorn, t.ex. itkDerivativeImageFilter och itkDerivativeOperator.

ITK´s h-filer är för det mesta väl dokumenterade och tar därför inte lång tid att förstå och anpassa sin kod efter. Ibland kan det vara intressant att titta i doxygen-dokumentationen för att se ITK´s klasshierarki, eller för att få en uppfattning om vad något i koden står för (exempelvis använde vi den för att förstå macro till viss del).

Exempel: Jämför vår funktion itkInvertImageFilter.h med ITKs funktion itkSinImageFilter.h, vilken vi använde som guide. Detta kan ge en grundläggande uppfattning till hur man kan utgå från ett filter och skapa ett eget.

Här följer ett mer ingående exempel av delar man kan stöta på i ett filter. I detta fall valdes MeanImageFilter.h som exempelfilter.

- Template: De flesta ITK inkluderingsfiler börjar med en template, vilket inte är mycket att bry sig om, men den ska finnas där (ITK är uppbyggt med templates). Den tar oftast upp inbild och utbild som klasser om det är ett filter, vilka är samma som används nedanför i själva filterklassen.

- Macro: Dessa kan finnas i många former. Här följer de i MeanImageFilter.h med förklaringar.

    - itkStaticConstMacro(namn, typ, källa)
      Denna används främst för att plocka ut statiska konstanter ur data. Vanligen bilddimension.

    - itkNewMacro(Self)
      Denna finns alltid med. Den skapar filterobjektets initieringsanrop. (::New())

    - itkTypeMacro(namn, typ);
      Denna anger vilken huvudklasstyp som ärvs.

    - itkSetMacro(namn, typ);
      Denna skapar ett set-anrop för att ge m_”namn” ett värde av typen ”typ”. Anropet kommer se ut som .Set”namn”(värde); från filtret.

    - itkGetConstReferenceMacro(namn, typ);
      Denna skapar ett get-anrop som returnerar en konstant referens till variabeln m_”namn” av typen ”typ”. Fungerar på motsvarande sätt som Set.

- Typedef: Finns fler av dessa än något annat i ITK. Generellt så är det för att korta ner namnen och slippa ändra på flera ställen i koden.

- Variabler: Som tidigare nämnt använder man variabler med prefixet ”m_”. Ex: m_Radius.

Vidare så har vi itkMeanImageFilter.txx.

- Template: Även här dyker template upp. Dessa finns vid varje funktion i txx-filen.

Finns inte mycket mer i txx-filen som inte är typisk C++-kod med använding av olika klasser. Något att observera är funktionen ThreadedGenerateData(), som är det som kommer köras vid ett typiskt Update() anrop i ITK. I t.ex. itkDerivativeImageFilter.h är det istället en GenereateData() vilken inte är ”threaded”, men skickar vidare data till en funktion som är ”threaded”.

Det mesta övriga kommer troligtvis vara ganska tydligt för en erfaren programmerare.

5.2 Egen funktion – itkInvertImageFilter.h

Som ett försök att skapa en egen filterfunktion till ITK designade vi ett filter som inverterar bilden. Detta var den enklaste typ av egen funktion vi kunde tänka oss efter att ha studerat några av ITKs funktioner. Filtret appliceras punktvis och har ingen omgivning, d.v.s. det är ett 1x1 filter. NxN filter går givetvis också att konstruera, och vi var inne på det från början, men fick lite problem med att resultatbilden av ett relativt simpelt filter bara blev svart. Misstankarna gick mot att vi missat något som t.ex. att det kanske har ett riktningsberoende eftersom vi utgick från deriveringsfilter som guide. Men teoretiskt ska det inte vara speciellt mycket svårare att filtrera med större filter.

itkInvertImageFilter.h är ett väldigt simpelt filter. I grund och botten är det bara en punktvis operation, vilket i detta fall är ”1 minus bildpunktsvärde”. Just på grund av detta val av punktvis behandling så behöver filtret ett stödfilter som sätter bildens intervall till mellan 0 och 1 (decimalform tillåten såklart). Detta stödfilter är inte inkluderat i själva inkluderingsfilen, utan krävs att man vet om att man ska behandla sin bild med först i huvudprogrammet man skriver. Stödfiltret kallas itkRescaleIntensityImageFilter.h och används faktiskt två gånger i huvudprogrammet som använder filtret. Detta för att vi i vår utbild faktiskt vill ha värden som ligger mellan 0 och 255. (Vilket var ett av våra större problem vid kompileringen)

Eftersom vi gjorde ett sådant simpelt filter så har vi inga parametrar man kan ställa in på filtret.

Det egendesignade inverteringsfiltret finns tillsammans med huvudprogrammet med som bilaga om man vill se koden i detalj.

5.3 Placering av kod

Själva funktionerna kan läggas till på flera sätt. När vi konstruerade vår funktion bestämde vi oss för att vår h-fil skulle vara fristående, d.v.s. vi hade h-filen liggandes i samma katalog som vår program-fil som använde den. Men vi har även testat att lägga till h-filer och txx-filer i ITKs inkluderingskataloger och sedan kompilerat det och sett att det fungerar. Det borde även vara möjligt att skapa sin egen ITK-katalog om man observerar att lägga till denna i ITKs makefiler. Det är upp till programmeraren att bestämma var de vill ha dem.

5.4 Kompilering (ej mex)

Om man inte är intresserad av att använda mex-filer, utan hellre vill göra en ”klassisk” kompilering till .exe-fil, rekommenderas att man besöker ITK’s hemsida (se referenser) och under ”documentation” följer länken i stycket:

- Tutorial Material. Presentations used for tutorials presented in conferences such as IEEE Visualization 2002 and 2003, MICCAI 2003 and SPIE Medical Imaging 2003 and 2004.

Detta dokument beskriver ingående hur man använder cmake för att kompilera filer, vilket undviker behovet av att manuellt leta upp h-filer och lib-filer för kompileringen. Vi hänvisar till dessa eftersom de har väldigt beskrivande och hjälpfulla bilder för både unix och windows kompilering, vilket skulle vara svårt att upprepa i textform.

6 Sammanfattning

6.1 ITK och VTK genom Matlab

Uppgiften kan sägas ha varit att kombinera ITK och VTK med Matlab. Utgångspunkten var att det skulle vara ett lätt sätt att utvärdera ITK och VTK för att se om dessa kan vara användbara.

Matlab fungerade utmärkt som GUI för ITK och VTK, men på grund av att Matlab inte kan hantera trådar fick vi anropa mexfunktioner istället för ett C++ huvudprogram. Detta innebar att vi var tvungna att konvertera data mellan Matlab och ITK och VTK istället för att lagras i C++ klasser. Detta påverkade inte prestandan mycket vad vi kunnat se.

När man väl bestämt hur implementeringen skulle ske, var det ganska enkelt att utföra eftersom det fanns en hel mängd med exempel och information.

6.2 Utvärdering av metoder och bilddata

Denna del kan beskrivas som en undersökning av många olika områden. Bland annat om metoderna påverkades mycket av om bilderna var förbättrade eller inte, eller vilken metod av de olika segmenteringarna som bäst kunde markera ett önskat område av en medicinsk bild, eller om metoderna överhuvudtaget fungerade bra nog.

Man kunde konstatera att själva programmet, kopplingen av Matlab till ITK och VTK, fungerade utmärkt. Att avgöra om bilderna som var original eller förbättrade gav bäst resultat var desto svårare. Ibland var det den ena typen, ibland den andra, och ibland fick man inte fram något riktigt bra resultat med något av dem. När man inte fick fram något resultat var det mer på grund av att det var svårt att ställa in parametrarna än att bilderna var dåliga. Allmänt så fick metoden med minst antal parametrar bäst resultat.

En segmentering i 2D tar ungefär 5-10 sekunder medan en segmentering i 3D tar 15-20 minuter på en dator med 2 GHz processor och 1 GB internminne. För att kunna läsa in volymer bestående fler än ungefär 50 bilder, med upplösningen 512*512 pixlar och 16-bitars gråskala, så krävs mer än 1 GB internminne.

6.3 Utökning av ITK

Att utöka ITK var det sista som behandlades och fanns inte med som plan från början. När man väl kommit förbi länkningsproblemen genom att använda CMAKE istället för att manuellt försöka ställa in inkluderingsfiler och libraries, så gick det ganska fort.

Man får intrycket av att utöver det mycket stora utbudet av funktioner i ITK, kan man ganska lätt lägga till nya funktioner genom att använda de befintliga funktionerna som referenser.

6.4 Slutsats

Vårat system klarar att segmentera och visualisera medicinska bilder med ganska bra resultat, under förutsättning att man ställt in bra parametrar. Det krävs dock en ganska bra dator för att det inte ska ta allt för lång tid att segmentera i 3D.

Referenser

Projektmodellen LIPS, Tomas Svensson & Christian Krysander

ITK Software Guide, *http://www.itk.org/ItkSoftwareGuide.pdf*

ITK Tutorials, *http://www.itk.org/HTML/Tutorials.htm*

ITK Doxygen component information, *http://www.itk.org/Doxygen16/html/index.html*

The VTK's User Guide, version 4.2, Kitware, Inc. publishers

Mathworks, *http://www.mathworks.com/support/tech-notes/1600/1605.html*

Mathworks, *http://www.mathworks.com/access/helpdesk/help/techdoc/apiref/index.html*

Hemsida ITK, *http://www.itk.org*

Hemsida VTK, *http://www.vtk.org*