Project Report
on
Multi-Planar Reconstruction using VTK

Supervised By : **Prof. Srikanth T.K**

Submitted by:
srinivas.r.vaidya
(MT2013152)

@iiitb.org

14 - December - 2014

Contents

1 Introduction 2
- 1.1 Objective 2
- 1.2 Motivation 2

2 Functionality 3
- 2.1 Multi Planar Viewer Window 3
- 2.2 Screen Pick Points 4
- 2.3 MultiPlanar Reconstruction from standard plane 4
- 2.4 MultiPlanar Reconstruction from oblique plane 4
- 2.5 Zoom 5
- 2.6 Rotation 5
- 2.7 Traverse View 6

3 System Design 7
- 3.1 Algorithm to obtain MPR 7
- 3.2 Code description 7
  - 3.2.1 Variables 7
  - 3.2.2 Procedures 8
  - 3.2.3 Data flow in VTK Pipeline 9

4 How to run 10
- 4.1 Tools & Installation 10
- 4.2 Where to get the source code 10
- 4.3 How to run the code 10
- 4.4 How to plot points and render MPR 11
- 4.5 How to clear MPR 11
- 4.6 How to rotate the MPR image 11
- 4.7 How to traverse the planar view 11

5 Challenges faced, and what have been left open 12

6 References 13

Chapter 1

Introduction

1.1 Objective

In medical imaging computed tomography (CT) and magnetic resonance imaging (MRI) provide three-dimensional volumetric data sets of the human body, which contain these objects of interest. The data gained from CT and MRI, however, contain many objects of less or no interest. This makes volume-rendering without preprocessing often impossible or inaccurate. Furthermore the objects of interest are hardly located entirely within a single plane. One way to display tubular structures for diagnostic purposes is to generate longitudinal cross-sections in order to show lumen, wall, and surrounding tissue in a curved plane. This process is called Curved Planar Reformation (CPR). This process is sometimes referred to as Multi Planar Reformation (MPR).

1.2 Motivation

MPR are routinely used in examinations of the CP angles and cranial nerves, pituitary gland, solitary and multiple space occupying lesions of the brain, in the knee joint, and for examining the kypho-scoliotic spine[1].

Chapter 2

Functionality

2.1 Multi Planar Viewer Window

The window of the Multi Planar Viewer is divided into 4 parts:

- Top right renderer displays 3D dataset.
- Top left renderer displays an oblique axial plane.
- Bottom left renderer displays sagittal plane.
- Bottom right renderer displays axial plane.

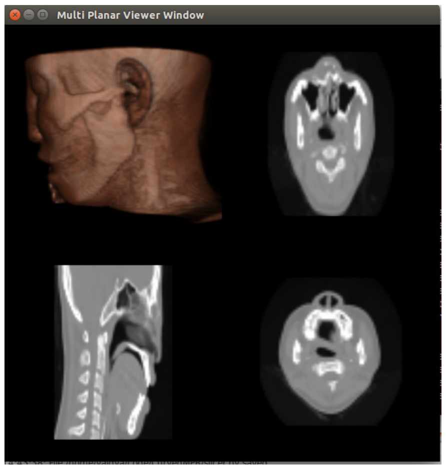


Figure 2.1: Multi Planar Viewer Window

2.2 Screen Pick Points

User can pick points to draw a polyline on any plane in multiplanar viewer window to reconstruct a multiplanar image. User can also clear the previously drawn polyline, and a draw a fresh polyline.

2.3 MultiPlanar Reconstruction from standard plane

The window displays MultiPlanar Reconstruction image generated from the polyline plotted on stadard plane(axial, sagittal) by user.

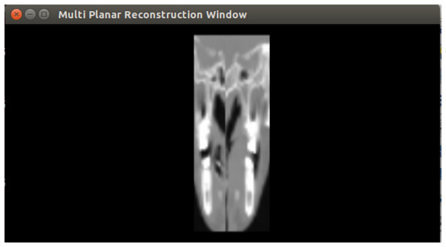


Figure 2.2: Multi Planar Viewer Window, shows MPR generated from points draw on axial plane

2.4 MultiPlanar Reconstruction from oblique plane

The window displays MultiPlanar Reconstruction image generated from the polyline plotted by user on an oblique plane.

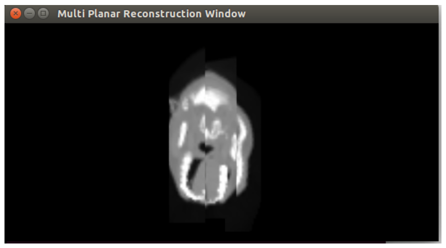


Figure 2.3: Multi Planar Viewer Window, shows MPR generated from points draw on oblique plane (45$\circ$ to axial plane)

2.5 Zoom

User can change the zoom level in all planar views in multiplanar viewer window and in multiplanar reconstruction window.

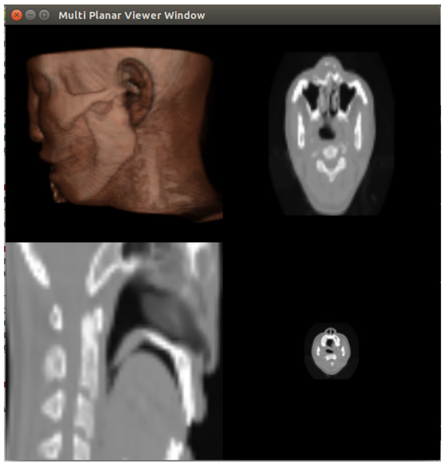


Figure 2.4: Multi Planar Viewer Window

2.6 Rotation

All planar views in multiplanar viewer window and in multiplanar reconstruction window can be rotated.

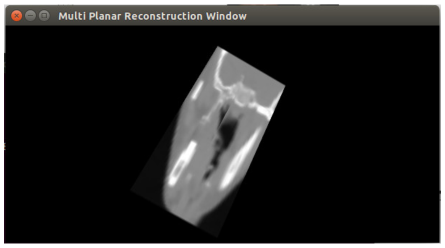
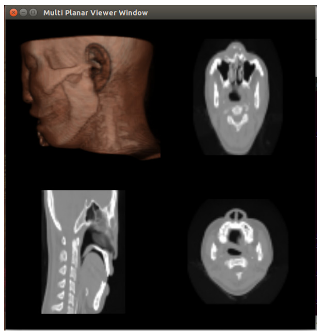


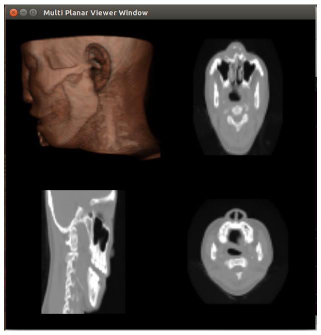
Figure 2.5: Multi Planar Viewer Window

2.7 Traverse View

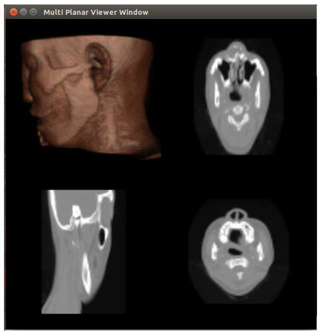
All planar views in multiplanar viewer window can be traversed with up and down with constant depth(which can be programmed).



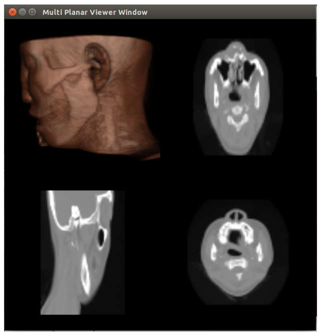
(a)



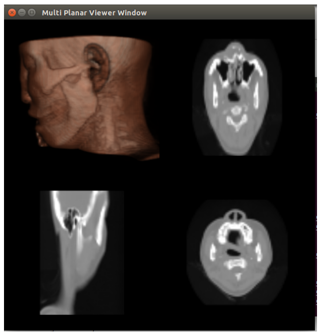
(b)



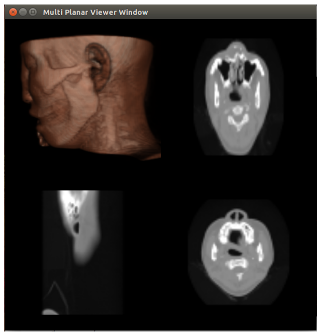
(c)



(a)



(b)



(c)

Figure 2.6: Traverse views of sagittal plane

Chapter 3

System Design

3.1 Algorithm to obtain MPR

Step 1: User pick points to draw a polyline(or piecewise linear curve) on any plane.
Step 2: Corresponding world space coordinates of the clicked screen coordinates is obtainted.
Step 3: If user has plotted n points, then are (n-1) edges. Length of each edge is computed by calculating (euclidean) distance between two vertices of edge.
Step 4: Angle of each edge is computed.
Step 5: Using angle and normal information of the plane on which this points are plotted, the orientation of the slicer is calculated and a slice is obtained.
Step 6: Using distance(as obtained by step 3), the slice is stripped to obtain only area of interest.
Step 7: Repeat Step 3 to Step 6 for every edge in the polyline.
Step 8: Using distance(as obtained by step 3) render window location of final stripped slice is determined.
Step 9: Render to view MPR

3.2 Code description

3.2.1 Variables

- *clickPointsList:* A list which holds user click points in world space coordinates.

- *aSlice:* List of vtkImageReslice class object, each objects interpolation method, reslice axis, OutputDimensionality are set, and reads data directly from vtkVolume16Reader.

- *color:* List of vtkImageMapToColors class object, each objects lookuptable is set to a vtkLookupTable object. color object fetchs input from aslice object.

- *actor:* List of vtkImageActor class object. Each actor object inputs data from color object.

- *mpv_renderer:* List of 4 objects of vtkRenderer class, which are responsible for 4 renderers in multiplanar viewer window.

- *numberOfSlices:* Holds number of slices that is composed to obtain final MPR image. Its value is (no:of click points) - 1

- *extendLength:* A List which specify the extent length(strip excess or unimportant part) of each slice

- *setLocation:* A List which specify render window location of each slice strip.

3.2.2 Procedures

- findDistanceBetweenTwoPoints: returns euclidean distance between two vertices which is passed to this function

- findAngleBetweenTwoPoints: returns angle between two vertices which is passed to this function

- computeMPR: this function performs main algorithm. Three parameters are passed are click points, standard plane on which points are drawn, oblique angle($= 0$; if operating in standard plane)

- displayClickPoints: displays polyline from user click points

- Mouse handler functions:
  - MouseMoveCallback
  - LeftButtonPressEvent

- Keyboard handler function:
  - KeyPressEvent

3.2.3 Data flow in VTK Pipeline

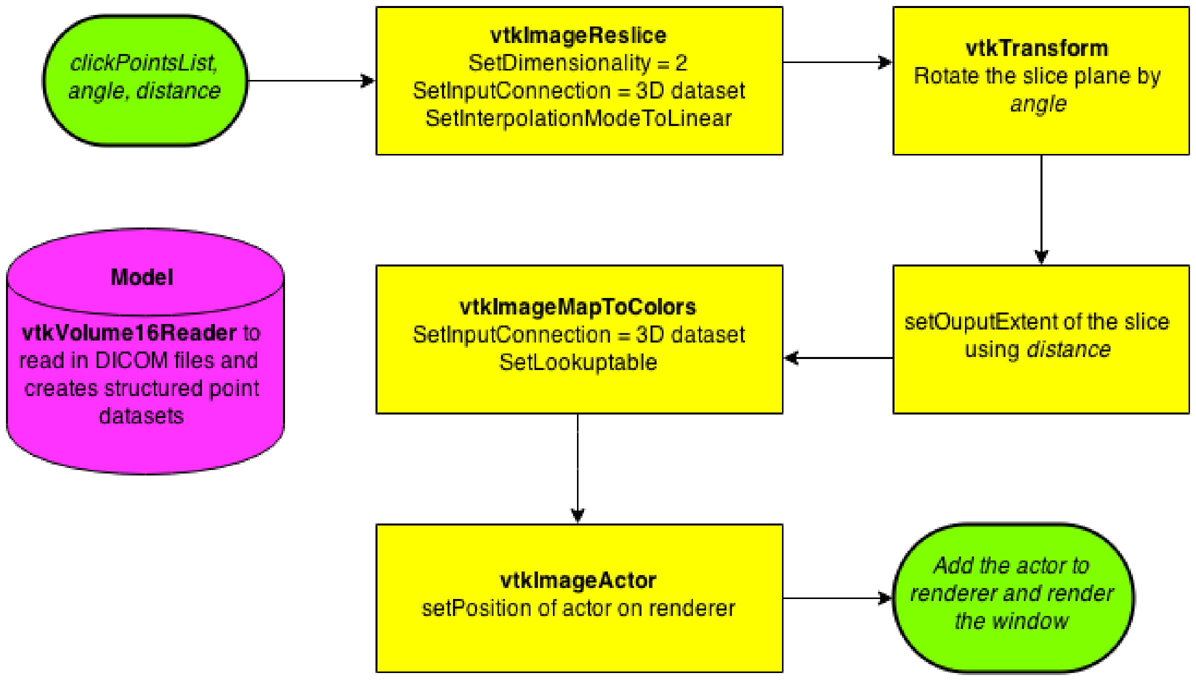

clickPointsList, angle, distance
vtkImageReslice
SetDimensionality = 2
SetInputConnection = 3D dataset
SetInterpolationModeToLinear
vtkTransform
Rotate the slice plane by angle
Model
vtkVolume16Reader to read in DICOM files and creates structured point datasets
vtkImageMapToColors
SetInputConnection = 3D dataset
SetLookuptable
setOuputExtent of the slice using distance
vtkImageActor
setPosition of actor on renderer
Add the actor to renderer and render the window

Figure 3.1: Data flow in Visualization Toolkit pipelne performed for each edge in polyline.

Chapter 4

How to run

4.1 Tools & Installation

On linux machine install python-vtk and python:

```
$sudo apt-get install python
$sudo apt-get install libvtk5-dev python-vtk
```

To test if python-vtk is correctly installed, the following command should start python-vtk interpreter.

```
$python-vtk
```

If still unable to run python-vtk interpreter, then

```
$sudo apt-get update
```

will resolve any dependencies.

4.2 Where to get the source code

Fork the repository or one could download the source code from the link given below:

https://github.com/srinivasrvaidya/Multi-Planar-Reconstruction-using-VTK

4.3 How to run the code

After installation, move to folder containing the source code.

```
$python slicer.py
```

Two windows,
1) Multiplanar viewer window, and
2) Multiplanar reconstruction window, pop up.

4.4 How to plot points and render MPR

On any plane in the multiplanar viewer window using mouse, pick points.
To render, press "R" or "r"

4.5 How to clear MPR

To clear MPR, press "C" or "c"

4.6 How to rotate the MPR image

Left mouse button down, and move the curser to get desired orientation.

4.7 How to traverse the planar view

Click on the desired plane, and use scroll to traverse up or down.

Chapter 5

Challenges faced, and what have been left open

Some of challeges faced are,
- To generate oblique MPR
- To set location of slice actor on the render window.
- To strip the desired location of the slice
- To get world coordinates from screen coordinates.

Some functionality left open,
- To display polyline plotted by user.
(Soln: Need to Append all your polygonal geometry with vtkAppendPolyData and pass it to vtkDepthSortPolyData.)
- Number of slices ≥ 5; size of strip displayed on MPR window decreases.
(This issue could be error in setLocation, or auto truncation by vtk, or both)

Chapter 6

References

[1] *"Multi-planar Reconstructions in Routine Applications"* by Greg Brown, published at www.users.on.net