Realistic 3D Face Reconstruction Based on VTK and MFC

RONG Jian , ZHAO Cuilian , FAN Zhijian , HU Shidong
Shanghai Key Laboratory of Mechanical Automation & Robot, Shanghai University
Shanghai, 200072, China
myrzh_1986@163.com

Abstract--**Nowadays VTK is a visualization development tool widely used in the visualization area, and MFC is a popular interface development tool in Windows platform. This paper analyzes the inner mechanism of VTK, gives both the methods of integrating VTK into MFC and accomplishing interactive computation and display of multi-windows. Finally, based on the construction of a visualization software platform, 3D reconstruction of realistic face model with VTK interfaces is also introduced.**

Key words: VTK; MFC; Face Reconstruction; Realistism

I. INTRODUTION

3D modeling is a basic topic in the fields of computer vision and computer graphics. Due to its unique popularity, human face becomes a hot study point, many experts involves in 3D face modeling. Nowadays 3D face modeling[1] technologies have been in wide use in various fields including movie making, game entertainment, medicine, authentication of identities and man-machine interaction, it becomes a hot focus of computer graphics and receives wide concern.

Since Parke established the first face model[2] in 1970s, many researchers have been dedicated to the 3D face modeling. Especially, after the rapid development of computer vision and computer graphics since 1990s has given much technical support for 3D face modeling, many researchers have been trying to construct realistic face model by 3D reconstruction tools such as Matlab and OpenGL[3][4]. Matlab has many functions of image processing, but it has a low efficiency of code execution; OpenGL is powerful in its 3D graphic functions, but cost a long development cycle because of its feature of bottom layer.

VTK[5] is the most popular tool for visualization development in the medical field, which has powerful functions of image processing and data visualization. Researchers can directly utilize its existing algorithms of image and graphics to construct sub-systems to deal with image and graphics data, which prevents from unnecessary repeated tasks. On the other hand, VTK has display and interaction functions of 3D graphs and can be used to compare the performance of different algorithms in the same development platform. However, VTK lacks of practical and flexible user interfaces, using VTK to construct a practical system for 3D data visualization or image processing should be with the help of other GUI toolkits, such as Console, Qt and FLTK et al. Considering practicability and flexibility under Windows OS, the interfaces mentioned above are much worse than the GUI of MFC, which is very prevalent at present. The realization of integration VTK into MFC has been receiving wide concerns in medical field[6][7].

The realistic 3D face modeling involves in the comprehension and comparison of different algorithms about processing and fusion of image texture and point-cloud data, and the interactive processing of data from multiple sources and multiple stations. Pointing to the data flow feature of VTK, based on MFC, this paper focus on the integration of VTK and MFC and the interactive computation and display of multi-windows. Finally, based on the construction of a visualization software platform, 3D reconstruction of realistic face model with VTK interfaces is also introduced.

II. THE INNER MECHENISM OF VTK

VTK, based on the 3D function library-OpenGL, is a visualization development library with object-oriented technology. It was published by GE R&D department for the first time in December 1993, and it is maintained by the United States Kitware Inc. It provides more than 300 C++ classes, and due to its advantages such as object-oriented, open source, portable, supporting Windows, Unix, Linux and other platforms as well as cross-platform development, it is very powerful and has been received enthusiastic applause from universities and research institutes around the world.

VTK adopts the pipeline mechanism[8], a typical pipeline structure can be divided into two parts. The first part is composed of the data generating and data processing elements, such as data generating class Source, data filter processing class Filter and class Mapper, the interface between data and rendering. Among them, data generating class Source can be divided into two types: one generates data

This work was supported by fund of Shanghai Science & Technology Commission under Grant No.0851151102, China.

through algorithms, such as class vtkCylinderSource, the other is to read data from a file or data stream, such as class vtkJPEGReader. The second part of the pipeline structure is consisted of elements which forms a virtual world, such as classes Props and Actor, a carrier for the visualization of data; class Renderer, which generates images to display in the 3D scene; class RenderWindow, which is used as the virtual camera screen in the 3D rendering scene, and so on.

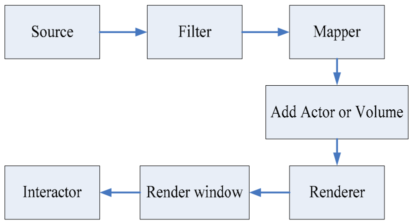

Figure.1 The pipeline of VTK

As shown in figure.1, Source is the beginning of a VTK pipeline, and data source is generated by reading files or other ways. Filter preprocesses the data as an independent calculation module, it can not only accept a number of data as inputs, but also generate a number of outputs. After processed by a Filter, the original data can be changed into the data format which meets the requirements of VTK. Then the Mapper, the interface between the filtered data and the graphic model, transforms filtered data into graphic data. To display data, first, a RenderWindow should be created. A RenderWindow has an interactive device--Interactor, which is used for handling the input of the window area received from keyboard and mouse and completing the interaction with users. An Actor is an entity in the display window, it receives the data property transmited from the Mapper. Then a Renderer should be created, displaying the final result of the rendering in the window. A number of Actor objects can be demonstrated simultaneously in a Renderer.

III. THE INTERGRATION OF VTK AND MFC AND THE INTERACTION OF MULTI-WINDOWS

As described above, VTK offers no graphical user interface, it's only a C++ class library for visualization and image processing. The way to use the class library of VTK in MFC is the same as other class libraries and follows same rules. Everytime when a new project is built, we must set the path for the header and the library files. The libraries of VTK include the static libraries (.lib suffix) and the dynamic libraries (.dll suffix). The parameters for static library should be set as follows:

(1) Modify the value of [Project/setting]->Link->Input->Object/library modules to: （vtkMFC.1ib, vtkRendering.lib, vtkFiltering.lib） et al, a blank is used as the interspace between every two libraries. Among them, vtkMFC.lib is the key library for the integration of VTK and MFC.

(2) Set the path for the header files: assumed that the header files are locates at “C：\VTK\Include”, we should set as follow: [Tools/Options]->Directories->Show directories for:->Include files-> C：\VTK\ Include.

(3) Set the path for the libraries: assume that the libraries are located in “C：\VTK\Library”, we should set as follow: [Tools/Options]->Directories->Show directories for:-> Library files->C：\VTK\ Library.

With respect to using dynamic library, apart from the setting described above, the path of VTK dynamic libraries (.dll) should also be added to the environment variable PATH of the operation system, or copy these libraries to the content of the system. Thus, the executed program(*.exe) generated after compilation can be run.

The integration of VTK and MFC is to embed VTK render window to the view window of MFC, thus they can be controlled as a whole in the process of interaction. The only necessary setting in integration is to set the parameters of function SetParentId of the class vtkRenderWindow as the handle of current view or dialogue, the key codes are as follows:

```
vtkRenderWindow* renwin=
vtkRenderWindow::New();
renwin->SetParentId (this-> m_hWnd) ;
```

The result of integration is shown in figure.2.

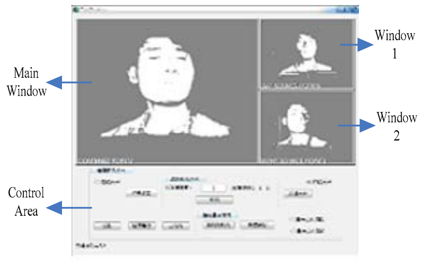

Figure.2 The result of integration

3D reconstruction usually needs data processing of multiple stations and sources related to the measurement view points, and accomplishing the real-time visualization and interaction of data computation from multiple sources can dramatically facilitate the reconstruction. This function can be realized by constructing multi-window software platform based on the integration of VTK and MFC. figure.2 shows an example of multi-windows, Window 1 and Window 2 show two different pieces of data respectively, the Main Window shows both of the two pieces data simultaneously.

The software system is developed based on the dialogue application program of MFC, the control of visualization and data processing utilizes the class of vtkAssembly, an interface class of VTK. It offers an function named AddPart, which is similar to a Renderer and can assemble multiple Actor objects to a unity to show and interact as a whole. The key codes are as follows:

```
vtkAssembly * vtkAssem=vtkAssembly::New();
vtkAssem->AddPart(vtkActor1);
vtkAssem->AddPart(vtkActor2);
```

IV. KEY TECHNIQUES OF REALISTIC FACE RECONSTRUCTION

Realistic face reconstruction is a progress to research how to generate specific face models and realize the realistism of face model. According to the type of data source, at present the main methods of realistic face reconstruction can be divided into two categories: (1) reconstruction based on images or pictures, including reconstruction based on CT image and common pictures. (2) reconstruction based on point-clouds data, including reconstruction based on 3D laser scanning and structural light 3D scanning points data. This paper focuses on 3D point-clouds data based reconstruction.

Generally realistic reconstruction from point-clouds includes data reading, data preprocessing, triangle meshing, texture mapping, visualization et al, as shown in figure.3. The ways of data preprocessing relies on the method of measurement. Triangle meshing aims to build a mesh model, which is the necessary preparation for texture mapping. Texture mapping is the fundamental of realistism. Both of triangle meshing and texture mapping are key techniques for realistic reconstruction.

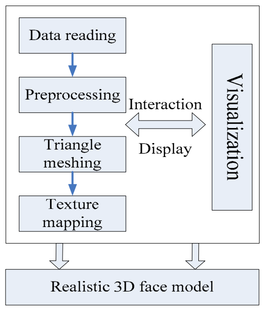

Figure.3 General realistic reconstruction framework

A. *Triangle Meshing*

During the process of 3D reconstruction, how to change the 3D point-cloud into a solid model is a vital issue and triangle meshing can change points data into solid model directly. There are many algorithms in the 3D points cloud meshing, yet most of them emphasize their results and the correctness is usually determined by experiences and visual effect. Only few can be proved theoretically, most of them being based on Voronoi chart[9] and Delaunay[10] triangle generation algorithms.

VTK offers class vtkDelaunay2D and vtkDelaunay3D, both of which are packaged with Delaunay triangle generation algorithm. Class vtkDelaunay2D is used to generate 2D Delaunay triangle mesh, commonly used for surface rendering[10], while class vtkDelaunay3D is used to generate 3D Delaunay tetrahedron, generally used for volume rendering[11]. The delaunay triangle generation algorithm belongs to a Filter in VTK, which accepts vtkPolydata topology data as an input.

B. *Texture Mapping*

Texture mapping is an important technique in computer graphics and the basis of the realistic 3D model reconstruction. According to the difference between texture definition fileds, texture can be classified into 2D texture and 3D texture. This paper focuses on 2D texture mapping.

The substance of 2D texture mapping is to map from a 2D texture plane to the surface of 3D object. Texture mapping usually requires the establishment of two mappings: one is from the texture space to object space and the other is from the object space to screen space, with the former more important, described in math expressions:

$$(U, V)=F(X, Y, Z) \tag{1}$$

(U, V) and (X, Y, Z) stand for a point in a texture space and an object space respectively.

Thanks to the measurement equipment ensuring that the texture space matches the object space one-to-one, namely (I, J) matches (X, Y, Z) one-to-one, we just need to accomplish the transform between the pixel coordinates (I, J) and the related texture coordinates (U, V) of a point for the texture mapping. The resolution of the captured pictures is 1280x1024, according to which the transform expressions are :

$$U=J/1280.$$
$$V=I/1024. \tag{2}$$

VTK offers an interface class vtkTexture for 2D texture mapping, which accepts data reading class vtkBMPReader as an input.

V. FACE RECONSTRUCTION WITH VTK

With the measurement data of a specific experimenter, this paper develops a multi-window visual reconstruction software system based on VTK and MFC, thus reconstructing a realistic 3D visual face model. The interface of the software is shown in figure.2, which is mainly consisted of three windows and a reconstruction control area. During the reconstruction, the data measured from the left and the right side are processed respectively, and visualized respectively on Window 1 and Window 2. The Main Window visualizes both of the two source data and finally displays the intact realistic face model. The buttons in the control area control the whole progress of the reconstruction.

The original data derives from a double-viewpoint structural measurement system[12], capturing data respectively from two measurement stations with a certain angle. The original data for once measurement from each station includes a 3D point-clould and an image. The format of the images is BMP with resolution of 1280x1024; the 3D point-clouds are saved as TXT files, with each point including（X, Y, Z, I, J）five parameters, (X, Y, Z) stands for the 3D coordinates and (I, J) stands for the pixel coordinates. The total points number of both point-clouds measured from each side is about 35,000.

As shown in figure.4, the face images of the left and right sides are respectively obtained from the left and the right measurement station.

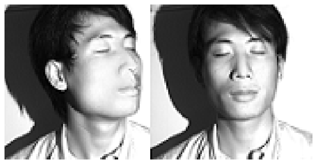
Figure.4 The images obtained by measurement

A. Data Reading And Initial Visualization

The left and right sides of data are read and visualized respectively. Taking the left side for example, VTK pipeline is executed as follows.

```
//Reading points from “left.txt” one by one
points->InsertNextPoint( x， y， z)；
//Build the topology for the points
leftPolydata ->SetPoints(points)；
//Mapping the points to the graphic library
leftmapMesh->SetInput(leftPolydata)；
//Generate an actor for the points
leftActor->SetMapper(leftmapMesh)；
//Add renderer to the rendering scene
renWin->AddRenderer(ren)；
//Interacter operation
iren->SetRenderWindow(renWin)；
//Add actor to the renderer
Ren->AddActor(leftActor)；
//Run the rendering
iren->Initialize()；
renWin->Render()；
```

As shown in Figure.2, Window 1 and Window 2 show the original point-coulds obtained from both stations stations respectively, and the Main Window shows them together.

B. Data Preprocess

Data preprocessing includes all the processes before triangle meshing, definitely including the thinning, segmentation and fusion of each point-cloud. Due to the particulararity of the captured data, all of these processes are not completed with the algorithms offered by VTK but developed by ourselves. Here VTK is not the interface of data processing but just the tool for visualization.

In order to improve the efficiency of the model reconstruction, it’s necessary to carry out thinning for the original point-clouds on the premise of certain modeling precision, and different thinning rules is adopted in different face areas. For the areas of the nose and eyes, a 3x3 matrix is used, we keep one point for each three rows and three lines; for the areas of the face and visor, a 9x9 matrix is used and one point is left for each nine rows and nine lines.

Because of the hairs and the veil of the light, which is the foundation of the segmentation, there are obvious blank regions between the neck and the nose and between the face and the nose in the obtained point-cloud, thereby achieving the data segmentation of face. figure.5 shows the result after thinning and segmentation.

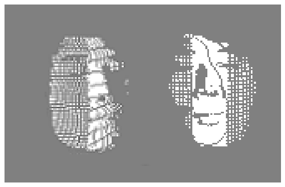
Figure.5 The result after thinning and segment

The final goal of the reconstruction system is to build a single front face model, so it is necessary to carry out point-coulds fusion of both sides to form a complete face. The accomplishment of the segmentation is based on the middle-vertical plane of the face, on which the middle-vertical line of the nose lies. The left side points data retains the right half face, and the right ones retains the left half. Both halves of face are visualized together in the Main Window and form a complete face, as shown in figure. 6.

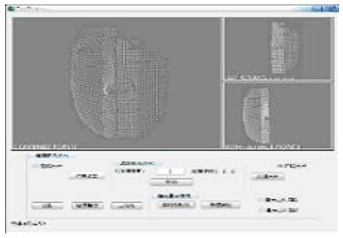
Figure.6 The result of fusion

C. Triangle Meshing

Face reconstruction is aimed at building the surface model of the face, which can be realized with the algorithm of surface rendering. Choosing the interface class vtkDelaunay2D packaging the 2D triangle meshing algorithm can implement the surface rendering for the solid face model. Taking data of the left side for example, the implement of the triangle meshing and the key codes are shown as follows:

```
vtkDelaunay2D *leftDelaunay2D=
vtkDelaunay2D::New（）;
//The implement of triangle meshing
leftDelaunay2D ->SetInput(leftPolydata);
```

The intact mesh model of the face visualized in the Main Window is shown in figure.7.

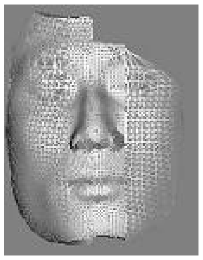

Figure.7 The intact mesh model of the face

D. Texture Mapping

Assuming the path of the image is "C:\left.bmp", the key codes of texture mapping are as follows:

```
vtkFloatArray *leftuvCoord=
vtkFloatArray::New( );
//Calculate texture coordinates one by one
//(I, J) are reading from "left.txt"
leftuvCoord ->InsertNextTuple(J/1280.,
I/1024.);
//Read the image
leftbmpReader ->SetFileName(C:\\test.bmp);
//Build the texture mapping
vtkTexture * lefttexture= vtkTexture::New( );
lefttexture->SetInputConnection(leftbmpReader
->GetOutputPort( ) );
leftpolydata->GetPointData()->SetTCoords(leftu
vCoord);
leftActor->SetTexture(lefttexture);
```

The final complete 3D realistic face model is shown in Figure.8.

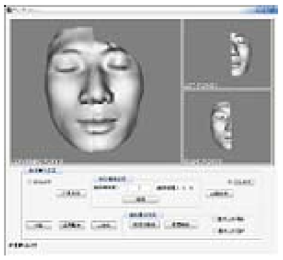

Figure.8 The final realistic 3D face model

VI. CONCLUTIONS

This paper briefly introduces the feature and functions of VTK. And then a multi-window system of realistic 3D face reconstruction is designed and realized based on the functions of image processing and data visualization of VTK and as the same time with the help of MFC. With the fully use of the powerful functions of image processing and data visualization and the close connection between MFC and Windows, the integration of VTK and MFC is achieved. On one hand, the establishment of multi-window software platform is based on the integration of VTK and MFC, which makes it possible to compute interactively and display multiple stations and sources measurement data real-time, facilitating the model reconstruction based on data of this type. On the other hand, using VTK to establish visualization 3D reconstruction systems can reduce numerous repeated tasks, which researchers focus more on the algorithms and system design. However, now this system can just server for the data acquired by the equipment mentioned above, the future study is to make a general interface for more kinds of point-clouds data.

REFERENCES

[1] XU Cheng-hua, WANG Yun-hong, TAN Tie-niu. Overview of Research on 3D Face Modeling [J] Journal of Image and Graphics, 2004,(08).(in Chinese)

[2] F. I. Parke. A parametric model for human faces. Technical Report UTEC-CSc-75-04, University of Utah, Salt Lake City, Utah, USA, 1974.

[3] Lee Y C, Terzopoulos D, Waters K. Realistic modeling for facial animation [A], Computer Graphics, Annual Conference Series, SIGGRAPH, 1995. 55-62.

[4] XU Lin, YUAN Bao-Zong, GAO Wen. Development and Prospect on Realistic Facial Modeling[J] Journal of Software, 2003,(04) (in Chinese).

[5] Schroeder WJ , Avila L S , Hoffman W. Visualizing with VTK: A tutorial [J] . IEEE Trans. on Computer Graphics and Applications ,2000 , 20 (5) : 20～27.

[6] YU Wei-wei, XI Ping, HE Fei. Study and Realization for 3D Reconstruction of Medical Models Based on VTK and MFC[J]. Journal of Engineering Graphics, 2009, 4, 125-130. (in Chinese)

[7] Wang Hongjian,Pu Xiaoming, 3D medical CT images reconstruction based on VTK and visual C++[C]. 3rd International Conference on Bioinformatics and Biomedical Engineering, Beijing, China,2009,1(1): 231-234.

[8] William Schroeder, Kenneth Martin, BillLorensen. The Visualization Toolkit-An Object-oriented To 3D Graphics (Third Edition), Kitware Inc, 2002

[9] N.Amenta and M. Bern. Surface reconstruction by Voronoi filtering. Discr. Comput. Geom.,1999, 22: 481-504

[10] Gopi M, Krishnan S, Silva C T. Surface reconstruction based on lower dimensional localized delaunay triangulation[J]. Eurographics,2000, 19 (3) : 467～468

[11] Yin Xuesong, Zhang Qian. Review on Four Volume Rendering Algorithms[J]. Computer Engineering and Applications, 2004,(16) (in Chinese)

[12] XIONG Yao-yang, CHEN Xiao-bo, et al. Development of three dimensional facial measurement system based on structured light projection [J] , Journal of Shanghai Jiaotong University(Medical Science),2009,(07)(in Chinese).