Sheffield Hallam Unviersity

Cordoba University

PANORAMA STITCHING

Final Year Project

Author: Daniel Martin Marin

Supervisor: Jan Wedekind

March 2007

Contents

Acknowledgements

I would really like to thank to the person who has helped me, with his infinite patience, amazing knowledge and sense of humor, to carry out with the project. Without his help it would not have been possible to create this project. So I would like to express my special thank to Jan Wedekind.

Also I would like to express my gratitude to Sheffield Hallam University and University of Cordoba, for giving me the opportunity to have lived this unforgettable experience, that has helped me to grow up not only professionally, but also in a personal way.

And Finally, special thanks to the whole MMLV staff, who have helped me when I have needed it, and my family by giving me the support to carry out with this experience.

CHAPTER 1
INTRODUCTION

1.1.- INTRODUCTION

Photographic cameras limitate the field of view of the pictures. If the picture is taken close to the scene, and the important part of that scene is small, then the photo can be suitable. But the problem comes when the scene is much larger. Normally, a person has a much wider field of view than a normal camera, and can see everything he wants, but cannot capture that view in a single image. Panoramas take care of this problem, allowing the viewer to stitch together multiple images that contain the entire scene.

1.2.- MOTIVATION

Panorama is defined as an *unobstructed* and *wide* view of an extensive area in all directions. Since many years ago, many people have been interested in panoramic pictures, which consist on the union of a serie of pictures in order to build a larger panoramic photo, wich has a bigger field of view. Nowadays, these kind of images are becoming more common in our society.

The development of this software will be quite useful for those who like photography, which can use this program in order to transform single images in a larger panoramic picture by means of stitching these single images together.

The main aim of this project is the development of the panorama software, which can be an interesting tool for those people who are interested in the properties that photography can offer, and panoramic images are nowadays very common and represent part of the modern photography.

1.3.- STATE OF THE ART

Some similar softwares related with this project have been developed. They have been the base why this project has been carried out. As example of these, we can list the following:

- Hugin (http://hugin.sourceforge.net/)
- PTGui (www.ptgui.com/)
- Panoweaver (http://www.easypano.com/es/)
- FirmTools Panorama (http://panorama.firmtools.com/)
- Panotools (http://panotools.sourceforge.net/)
- Autopano (http://www.autopano.net/)

All these programs offer computation of panoramic pictures using normal taken photographs. But Hugin, mentioned before, is one of the most frequently software for panorama stitching. Combined with Autopano and Panotools, they can create wider panoramic pictures. The development of this complete software has been the base for the performance of this project, *panorama.*

CHAPTER 2
SPECIFICATIONS

2.1.- HARDWARE RESOURCES

The hardware used on this project is a HP Intel ® Pentium(R) Centrino Dual Core 1,66 Mhz laptop, with one Gigabyte RAM, nVidia(R) Geforce graphic card with 128 MB of main memory, and 100GB Hard Disk Memory.

A digital camera is required for taking the pictures. These pictures are transferred to a computer and then stitched using the softeare

2.2.- SOFTWARE RESOURCES

The software was implmented on a Linux Kubuntu distribution Operative System, and also the toolkit, software and libraries that has been necessary for the development of the project are open source software (Gedit, Viewer, Open Office are example of them).

The graphical user interface was developing using QT Designer (http://www.trolltech.com/products/qt/features/designer), which provides the project an easy and intuitive interface for the user.

The software has been developed using C++. This language is object oriented and there are compilers for creating optimized binary code. There have also been used:

- QT4 for graphical user interface,
- BOOST and STL libraries, for the implementation of some specific functions

2.3.- SYSTEM DESCRIPTION

This panorama software has been created in order to create a panoramic picture from series of normal photos, which have some determined features:

- **Input:**
  - Set of pictures
  - Set of correspondences:
  - - Index of first and second picture
  - - x- and y-coordenate denoting a point in the first picture
  - - x- and y-coordenate denoting the corresponding point in the second picture
    - Viewing angle of virtual camera
- **Output:**
  - Common viewing angle of camera
  - Yaw, pitch and roll angle for each camera position
  - A panorama picture

What this software does is reading these specific features after reading a file that contains all this information, and then can load the determined images, angles, that is, yaw, pitch and roll, coordenates and control points (input), and now it is ready to stitch the images, creating a larger panoramic picture (output) .

After having created the panoramic picture, each single image can be selected and also rotated using the angles of viewing spinboxes, so the user can see how the images perfectly matches one with other.

CHAPTER 3
DESIGN

3.1.- INTERFACE

When running *panorama* program, a display window is shown, which is the interface of the main program, from where we can access to the diferrent options of the program. The interface has been created with Qt Designer, has the following appearance.

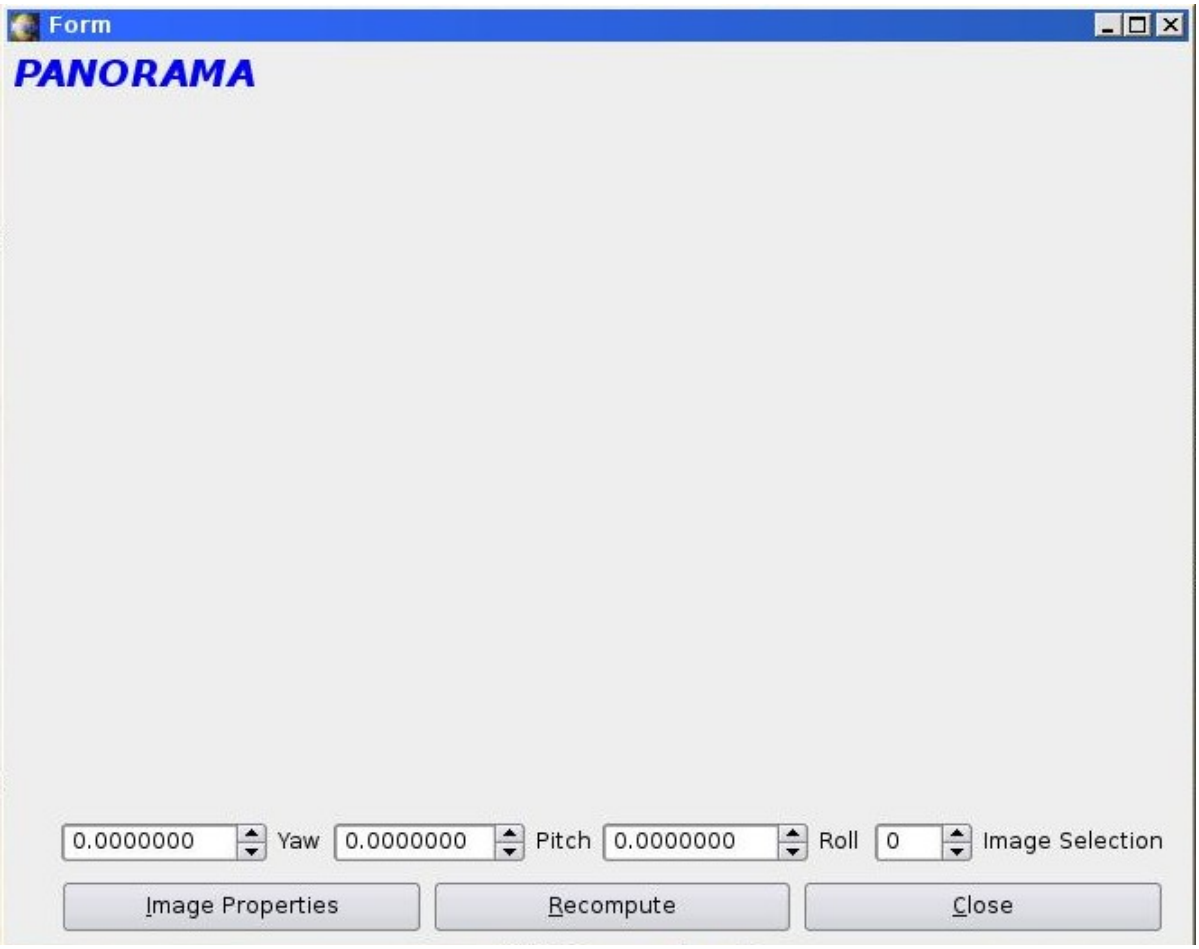

Figure 1: Panorama Display

The functions of each button are explained below:

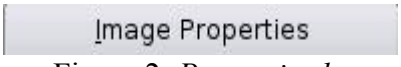

Figure 2: *Properties button*

When pressing Image Properties, the program let you search for a .pto file, that contains the properties of the images that we want to load and stitch.

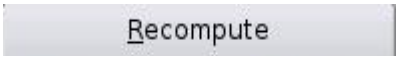

Figure 3: *Recompute button*

Recompute button is used to call the appropiate functions which manage the operations that make the images stitch together, converting them in an only large panoramic picture.

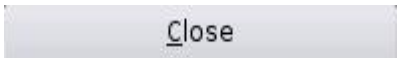

Figure 4: *Close button*

The only function of this button is to end the program and return to the console.

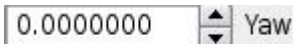

Figure 5: yaw spinbox

This spin box allows the user to rotate the image over the yaw direction. When you change the value, the image is automatically recomputed with the new position and moved to the desired place.

Figure 6: pitch spinbox

This spin box as a similar function than the yaw spin box, with the only difference that the image is moved in the pitch direction, instead of yaw.

Figure 7: Roll spinbox

As well as the other two spin boxes, this one rotates the image, but in the roll direction, that is, now is the angle of the images what rolls.

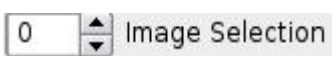

Figure 8: Image selection spinbox

This button allows the user to select the image that is previously loaded in order to change the desired values (yaw, pitch or roll) of the selected image.

3.2.- READING THE PTO FILE

When pressing the Image Properties button, the program automatically ask the user to search for a .pto file. This a file that is previously generated using Hugin software (mentioned before). This a text file and contains the different data from the images that the user wants to be stitched in a panoramic view.

This pto file contains, for each image, the name of the images that are going to be stitched. They are recognised and automatically loaded in memory so the program show them in the main window. Then it reads the three angles, that is, yaw, pitch and roll. They are subsequently used for the rotation matrices that make every single image to rotate in the desired angle. The file also contains the *horizontal field of view* (**hfov**) of every image. The field of view is the angular extent of the observable camera´s field of view. This value is later used, aswell as the rotation matrices, for rotating the images.

3.3.- CONTROL POINTS

The last data that the program reads from the pto file, are the control points (or key points). These are very important for the use of the software, because they point where are the images going to be stitched from, that is, the points that the images that are going to be stitched together have in common.

For example, let us take these two images that wanted to be stitched. Hugin is used to create the control points and compute the angles where the pictures are going to be stitched from.

Figure 9: Test1.jpg

Figure 10: Test2.jpg

Then, we can select the desired control points so Hugin can use them to stitch the images. In this example, there have been taken the following control points from Hugin. They are displayed in each image with a different colour. The control points are the common points that are found in every image. They are manually selected in order to obtain a better result of the stitch. We can see which control points have been taken for each image (figures 11 and 12).

Figure 11: Control Points 1

Figure 12: Control Points 2

A parser for reading the “pto” files was developed. Among other parameters the parser reads the coordenates of the control points and the camera angles computed by Hugin. The coordenates of the control points are displayed in the console as shown in figure 13.

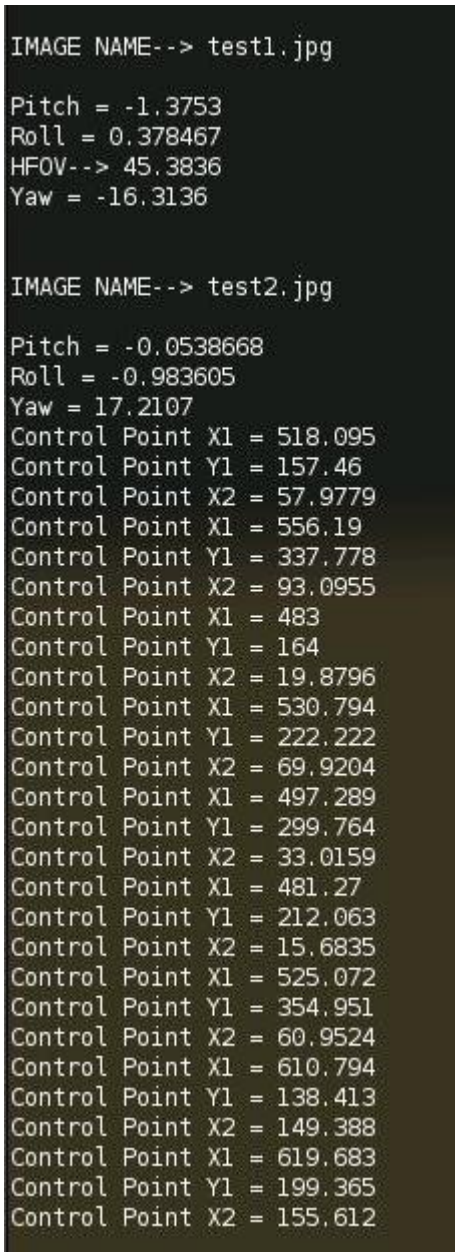

Figure 13: Control Points

After this, the program uses the camera angles to transform the images and display them in a single panoramic view.

3.4.- ROTATION MATRICES

A rotation matrix represents a 3D rotation, in 2D projective coordenates, where the origin of all coordenate system is coinciding .

Rotation has three grades of freedom: Yaw, Pitch and Roll

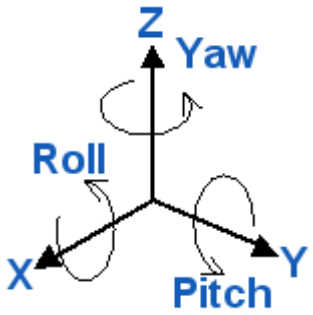

Figure 14: X,Y,Z axis

Changes to the image cause by a 3D rotation of the camera can be represented using a 2D vector-field

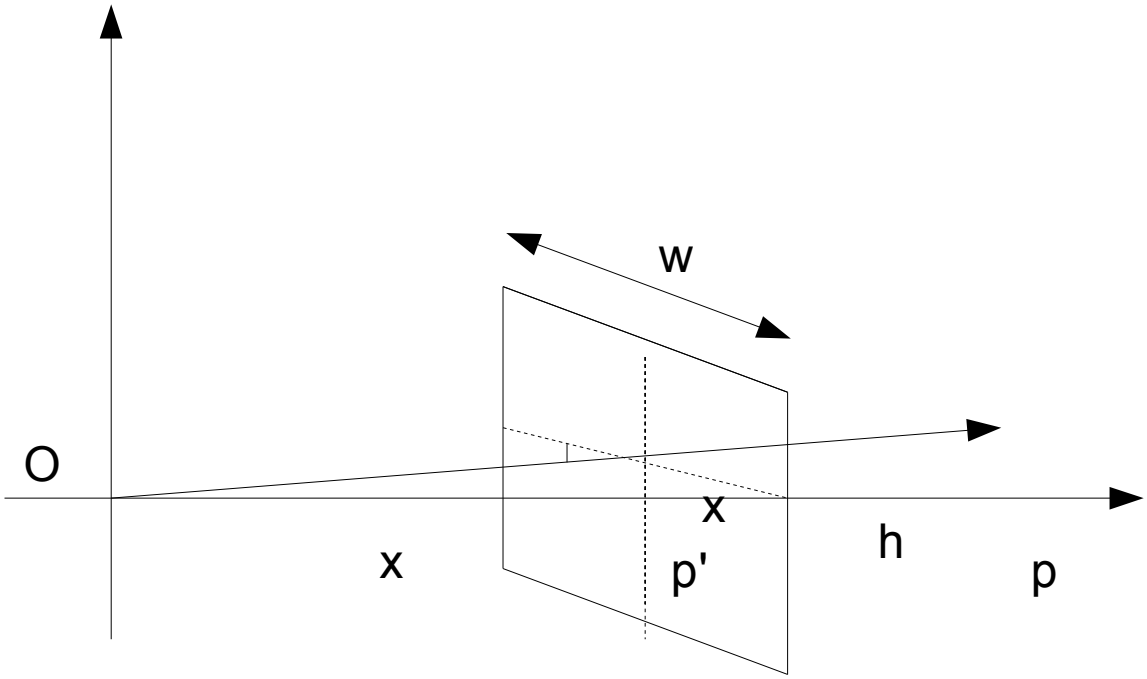

Figure 15: Image representation

In figure 16, *w* represents the width of the image, while *h* is the height. We want to calculate the image in position *p.*

In order to calculate the distance from the center to the point p', we use the following formula:

$$OP' = \begin{pmatrix} p_1' \\ p_2' \\ p_3' \end{pmatrix} = \begin{pmatrix} (i_1 - w/2)\Delta S \\ -(i_2 - h/2)\Delta S \\ f \end{pmatrix} \cong \begin{pmatrix} (i_1 - w/2) \\ -(i_2 - h/2) \\ f/\Delta S \end{pmatrix}$$

Figure 16: Vector OP

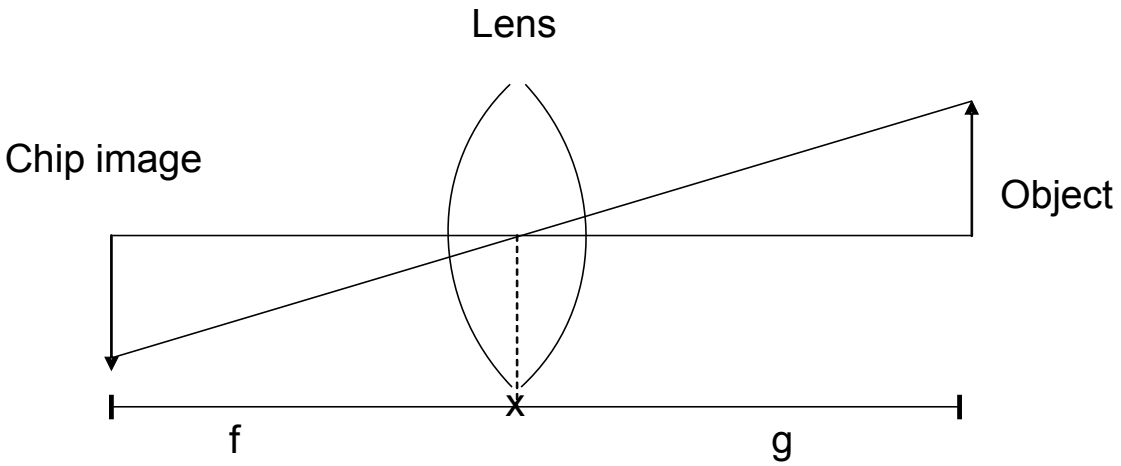

Figure 17: Object representation

Where i1 and i2 represent the image pixel. The width and the height are divided by 2 so they represent the center of the image, and finally ΔS is the physical size of a camera-pixel on the chip. The letter *f* is the focal length of the camera.

The angle represents the horizontal field of view, that is, the part of the image that we can see from one point, without moving. We can see it represented in figure 18 :

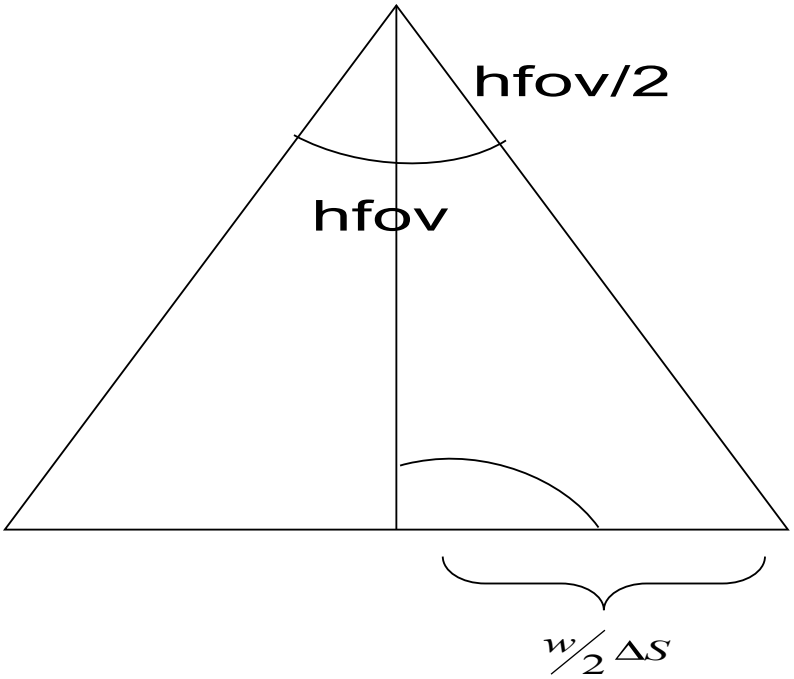

Figure 18: hfov angle

If we apply the definition of tangent,

$$tan\left(\frac{hfov}{2}\right) = \frac{w/2\,\Delta S}{f}$$

Figure 19: Tangent theorem

Then, we can get *v:*

$$v = f/\Delta S = \frac{w/2}{tan\left(\frac{hfov}{2}\right)}$$

Figure 20: Vector v

Now we have the vector OP, we can rotate the image from real point, to the new virtual point where the image will be placed. We obtain this point by multiplying the vector OP (see figure) with the rotation matrices (figures 21, 22 and 23)

$$R_x(\alpha) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{pmatrix}$$

Figure 21: Yaw Matrix

$$R_y(\beta) = \begin{pmatrix} \cos(\beta) & 0 & -\sin(\beta) \\ 0 & 1 & 0 \\ \sin(\beta) & 0 & \cos(\beta) \end{pmatrix}$$

Figure 22: Pitch Matrix

$$R_z(\delta) = \begin{pmatrix} \cos(\delta) & -\sin(\delta) & 0 \\ \sin(\delta) & \cos(\delta) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Figure 23: Roll Matrix

The Rx matrix represents the yaw rotation, that is, the rotation throughout the X axis, while the Ry matrix represents the rotation throughout the Y axis, and the Rz matrix represents the rotation throughout the Z axis.

$$R(\alpha,\beta,\delta) = R_x(\alpha)\cdot R_y(\beta)\cdot R_z(\delta)$$

Figure 24: Rotation Matrices Product

If we multiply the three rotation matrix, we will obtain a new matrix *R*. Then we multiply this new matrix by the vector *OP* (see figure 16), obtaining a new vector, *q:*

$$\vec{q} \cong \vec{q'} = \begin{pmatrix} \overline{i_1} - \overline{w}/2 \\ \overline{i_2} - \overline{h}/2 \\ \overline{f}/\overline{\Delta S} \end{pmatrix}$$

Figure 25: Vector q

$$\vec{q'} = \left(\overline{f}/\overline{\Delta S} \Big/ q_3 \right)\vec{q} = \left(v/q_3 \right)\vec{q} = \left(\frac{\overline{w}/2}{tan(\overline{hfov}/2)q_3} \right)\vec{q}$$

Figure 26: Vector q'

Finally we can calculate the new position of the image, which is determined by the following points:

$$Xs = \left(q(0)/q(2)\right)x\left(\frac{width/2}{tan(hfov/2)} \right) + width/2$$

Figure 27: x point

$$Ys = \left(q(1)/q(2)\right) x \left(\frac{width/2}{tan(hfov/2)} \right) + height/2$$

Figure 28: y point

CHAPTER 4
RESULTS

4.1.- Panorama Picture

Now the image is stitched, and we can test the yaw, pitch and roll points of every image just selecting it from the *image selector spinbox* (see figure 29)

Figure 29: Final Image

The yaw angle rotates from -180º to 180º; the pitch angle between -90º and 90º, and finally the roll angle is between -180º and 180º. All the angles measurements are taken in radians.

Here there is another example panorama:

Figure 30: Airport 1

Figure 31: Airport 2

Figure 32: Airport 3

Figure 33: Airport 4

Figure 34: panorama Airport

Taking the pictures

In order to make a panorama picture, there are some steps that must be followed when taking the pictures with a camera:

1.- The center of projection must be the same for all pictures
2.- The project must cover the field of view of the virtual panoramic camera
3.- There must be some overlap between neighbouring pictures
4.- The graph of overlap must not be partitioned
5.- The focal length of the camera should not change
6.- Aperture, exposure time and sensitivy of the camera should not change

CHAPTER 5
CONCLUSIONS

5.- Conclusions

As a result, this software is able to:

- Import Hugin panorama descriptions.
- Perform 3D image rotations
- Render panoramas
- allow to interactively change the 3D rotation for each picture

The software is an initial step for adoption of panorama software by the MMLV

5.1.- FUTURE WORK

- Incorporate compensation for camera distorsion
- Reimplement panotools panorama optimizer in an object oriented language

CHAPTER 6
IMPROVEMENTS

OpenGL Implementation

OpenGL is an open source graphical library used to develop 2D and 3D graphics. Developed by Silicon Graphics, is a method for rendering 3D graphics. It consists in about 250 different functions that make possible to draw three dimensional scenes from simple primitives.

To make this software more complete, it has been included OpenGL rendering for the images. Some functions from this library have been used in order to make faster the visualisation of the images, increase the speed of rotation, add transparency to the images or flash the selected picture.

We can clearly notice that when the final panoramic image is loaded, the visualisation is much faster than when the software rendering was used. OpenGL uses hardware resources for rendering the textures, that is, it makes use of the graphic card for rendering the images.

When selecting an image, it automatically starts to flash, in order to know which picture we are working with. The “flash” button can be enabled or disabled just clicking on it.

There has also been added transparencies on the images, in order to better notice where the pictures have been stitched from. The darker side of each picture indicates that this is the common part with the next picture.

Also a spin box has been added, which is used to bring the final panoramic image nearer or closer just scrolling over the spin box. We can manually adjust the perfect size of the image so we can see it better.

Finally, there have been designed a function for determined keys of the keyboard in order to select or rotate the image(s). For example:

- Up arrow key: moves the image up over the pitch axis
- Down arrow key: moves the image down over the pitch axis
- Left arrow key: moves the image to the left over the yaw axis
- Right arrow key: moves the image to the right over the yaw axis.
- Page Down key: rotates the image over the positive roll axis
- End key: rotates the image over the negative roll axis.
- Keyboard numbers: select the image 1, 2, 3... from the loaded pictures.

When an image is selected, it automatically loads the new values of the yaw, pitch and roll that it has.

In order to notice the speed difference, it is recommended to have a compatible graphic card, which supports OpenGL rendering.

Some of the most important OpenGL functions that have been used in this software are explained below:

- **glClearColor()** --> set up the screen to the specified color.

- **glShadeModel()** --> set up the color of the polygons.

- **glBindTexture() -->** binds the texture to the specified *target.*

- **glParameteri()** --> parametizers two filters, one when the texture is represented bigger than the reality(GL_TEXTURE_MAG_FILTER) and the other one for when is smaller (GL_TEXTURE_MIN_FILTER)

- **glGenTextures()** --> generates the textures with the specified size from the image

- **glEnable(GL_BLEND)** --> makes part of the image translucid.

- **glViewPort()** --> Specifies the size of the area of drawn.

- **glBindTexture()** --> Selects the active texture

- **glTexCoord2f()** --> allocates the texture in a specified position

- **glTexEnvf()** sets texture environment parameters

- **glTranslatef()** --> multiplies the current matrix by a translation matrix

- **glRotatef()** --> multiplies the current matrix by a rotation matrix

- **glBlendFunc()** --> specifies pixel arithmetic

- **gluPerspective()** --> sets up a perspective projection matrix

- **glTexImage2D()** --> specifies a two-dimensional texture image

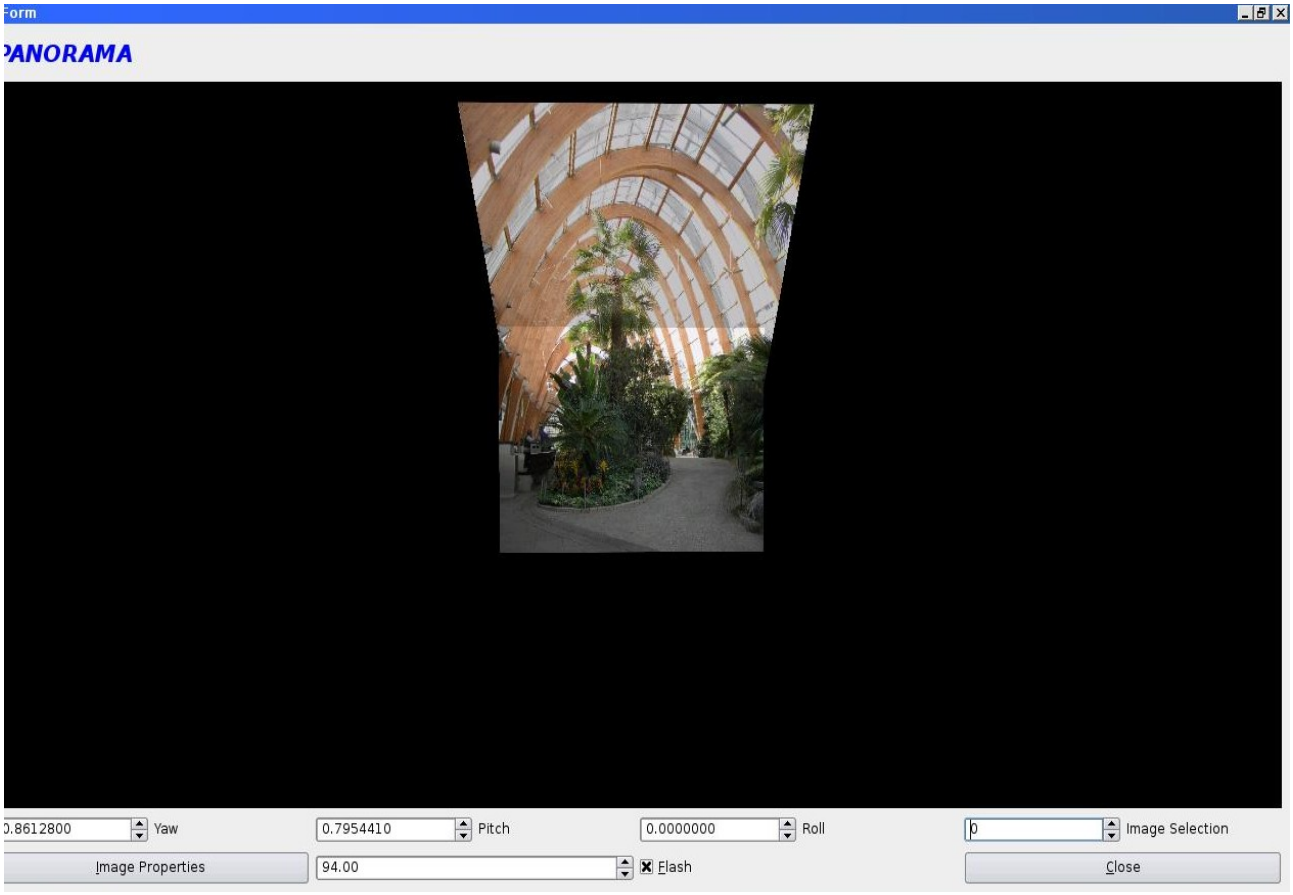

Figure 35 : Example of new interface

Figure 36 : Winter Gardens panorama

CHAPTER 6
FIGURE INDEX

Figure Page Number

REFERENCES

[1] The panorama factory. Website. Http://www.panoramafactory.com/equiv35/equiv35.html Last accessed March 20th ,2007

[2] Focal Lens. Website. Http://local.wasp.uwa.edu.au/~pbourke/modelling-rendering/lens/ Last accessed March 20th , 2007

[3] Qt (biblioteca). Website. http://es.wikipedia.org/wiki/Qt_%28biblioteca%29. Last accesed March 18th , 2007

[4] Rotation Matrices. Website. Http://es.wikipedia.org/wiki/rotation_matrix Last accessed March 10th , 2007

[5] C++ Syntax and fundamentals. Kyle Loudon. O.Relly, May, 2003. First Edition

[6] Spherical Panorama Project. Website http://vision.bc.edu/~dmartin/teaching/cs342/2006f/projects/Final/Results/Alec%20&%20Lawrence/ Last accessed March 5th , 2007

[7] Transformations in 3D . Website. http://home10.inet.tele.dk/moelhave/tutors/3d/transformations/transformations.html Last accessed March 5th , 2007

[8] El tutorial del desarrollo visual. Antonio Larrosa Jiménez. Revision 1.00.01, 2003 Antonio Larrosa

[9] Rodrigues' rotation formula. Website. http://en.wikipedia.org/wiki/Rodrigues'_rotation_formula Last accessed March 3rd , 2007

[10] Fast Construction of Dynamic and Multi-resolution 360 panorama from Video Sequences. Zhu, Z. and Xu, G. and Riseman, E.M. and Hanson, A.R. Journal of Image and Vision Computing. Volume 24, pages 13—26. 2006

[11] Multi-image matching using multi-scale oriented patches. Brown, M. and Szeliski, R. and Winder, S. Journal: Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on. Volume 1, 2005

[12] The OpenGL Redbook: OpenGL Programming Guide. Shreiner, D. and Woo, M. and Neider, J. and Davis, T. Publisher: Addison-Wesley Professional, 2003