Precise Merging of Multiple Phographs

Michal Seeman*

Department of Computer Graphics and Multimedia
Brno University of Technology
Brno/Czech Republic

Abstract

Photographs are records of projected images of the photographed scene. All the parameters of the projection are defined by the lens and the image sensor used.

To efficiently merge multiple photographs, the projection parameters must be adjusted and the lens distortion has to be computed and eliminated. This paper describes a method for merging of multiple photographs and for elimination of the error caused by the lenses. The presented method enables for creation of very large (wide) photographs with no geometrical distortion caused by lenses or even with a simulation of specific distortion.

Keywords: photography, lens, panorama, merging

1 Introduction

The aim of the presented work is to create a wide image by merging multiple photographs taken from the same place. For each photograph, an information about the lens, focal length, and camera direction should be used. Many commercial and freeware applications can be found for this purpose but most of them do not compute the lens distortion (Canon PhotoStitch). Instead of creating realistic projection of the scene, they only deform the images to fit together so some parts of the images could be misshaped. Some precise applications for merging images do exist but they are either very expensive or difficult to use.

One of the major drawbacks of the majority of commercially available software systems is that they do not allow for proper adjustment of the final projection of the merged photographs. The proposed system allows the user to project the photographs obtained from some point in a 3D space onto a sphere with that point in the centre. The user can select a most suitable view of the photographs to be merged and render the output image.

To merge the images precisely, the lens distortion has to be removed firstly. It gives the pure error-free information about the scene. The method is discussed in the section Distortion elimination.

2 Unit sphere image merging

The method proposed in this paper is based on the idea that the photographs taken from the same place can be projected on a unit sphere. The unit sphere with the photographs then can be projected on a plane that represented the merged photographs. After the user selects the projection, the final photograph containing the merged photographs can be rendered.

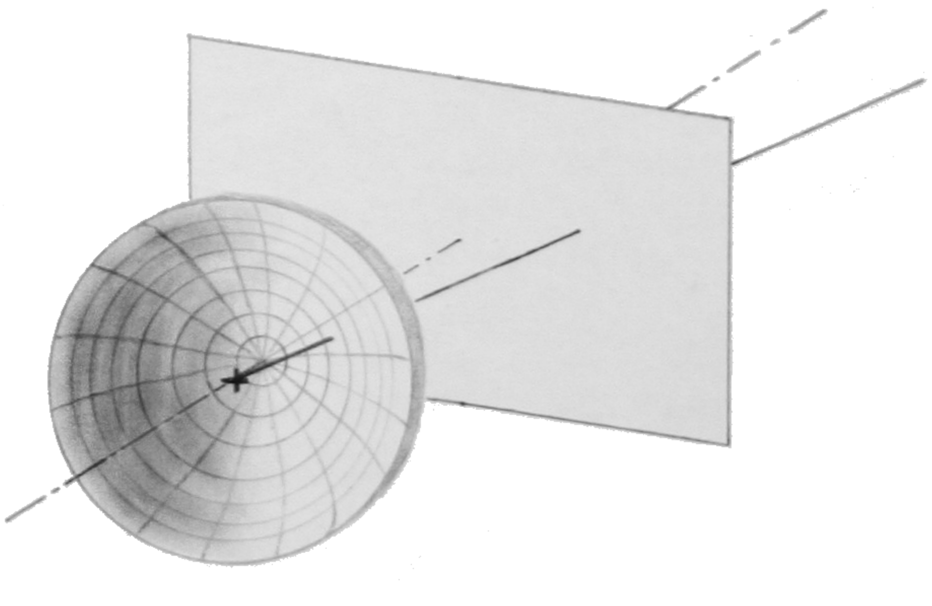


Figure 1: Unit sphere projection

*xseema00@stud.fit.vutbr.cz

The geometrical properties of the unit sphere are suitable also for description of the lens used during photographing. If an ideal reverse lens transformation existed, it would get a 3D vector from each place on the photograph; however, as the distance of the photographed objects is unknown, the length of the 3D vector would be unknown, only the direction would be valid. So length of all these vectors can be set as 1. Therefore, all the points of the photograph are considered to lie on the unit sphere. This sphere, seen from it's centre would simulate the photographed scene's appearance (see Figure 1).

3 Image distortion elimination

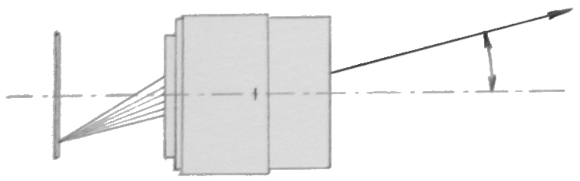


Figure 2: Lens function

The photograph is a projection of the objects in the scene onto a sensing plane (Figure 2). As true information about the scene is needed, both the image data and the lens parameters must be known because the lens performs the transformation from a 3D space into the flat (plane) coordinates.

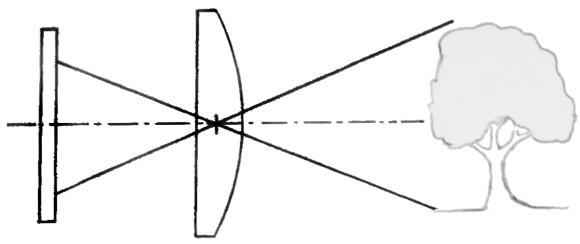


Figure 3: Lens projection

To implement the projection, the lens has to be described in some way. The general lens transforms any point in the space into a point lying at the sensors sensing plane (or to an invalid point if the point in space is out of the view). This transformation should be simplified and mathematically described so that is is usable in the software. Most of the real lenses are axis symmetrical. It means that all the optical elements in the lens are symmetrical according to the lens axis. The errors caused by the non-symmetry of the lenses are not considered in this paper. The only value to be transformed, if the lenses are considered symmetrical, is the angle from the lens

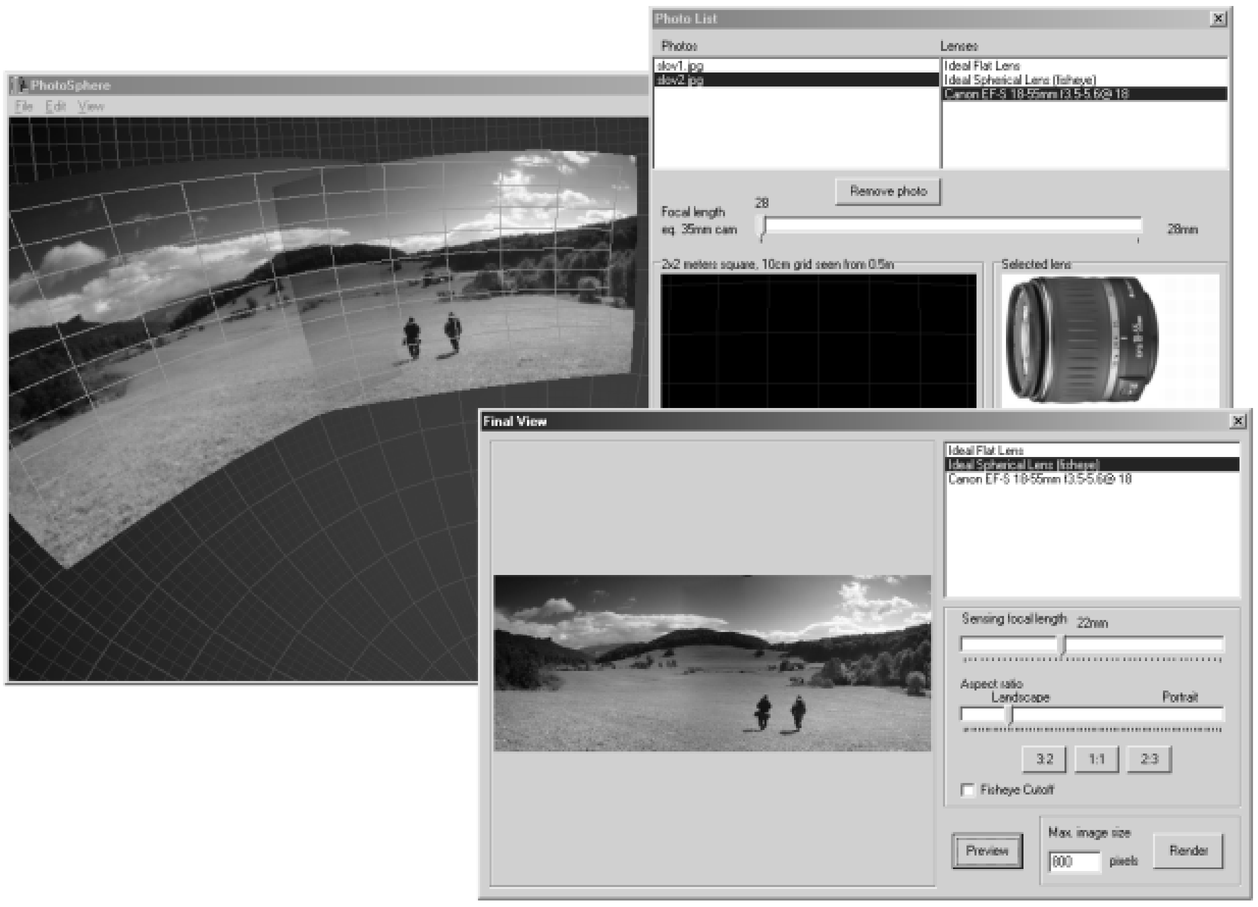


Figure 4: Screenshot from the program

axis to the pointing vector. So the lens function can be represented by relation between the distance from the sensor centre and the angle from lens axis (Figure 3). For better usage features, let us define a function which uses tan (angle) instead of the angle itself. This function is be easier to work with, from several points points of view. For example, this function is linear for the ideal lens.

The function can be described as

sensor distance = l (scene angle)

and we can define inverse function

scene angle = l^{-1} (sensor distance)

The functions of the ideal lenses are given analytically and functions of the real lenses are interpolated based on a number of measured points (samples). The inverse (projection) function could be evaluated easily. Now as both simple lens functions, forward and inverse are known, the complex lens functions can be found.

sensor position = L (scene vector)

scene vector = L^{-1} (sensor position)

Let us assume that all of the photographs were taken in different direction, though from the same place. The angle has to be represented in some way. The proposed way is to use matrix of 3*3 floating point numbers that form a vector base. Each vector of the base is of unit size and the vectors are mutually perpendicular. The direction could be obviously represented by three angles, but the vector base provides more comfort for further calculations. A transformation must be found that converts one photograph's co-ordinates into another one's. Such transformation can be evaluated easily. The co-ordinate vector has to be multiplied by the first lens's matrix and then by the inverted matrix of the second lens. This is simple to implement for each matrix is normal.

GlobalVector = Vector · Matrix

Vector = GlobalVector · $Matrix^{-1}$

where *Vector* is *GlobalVector* in the *Matrix* base's co-ordinates

4 Rendering the image

To evaluate a pixel's value, it's necessary to compute the pixel's 2D coordinates in the source photograph(s). This can be done using the above described projections:

The new image pixel's co-ordinates are transformed by the new image's lens function. The transformation creates 3D position vector. This vector is multiplied by the new image's direction matrix and then by the source image inverted direction matrix, as described in the previous section. The result is 3D position vector of the pixel, but seen from the source photograph direction. The last step is to transform the vector by the photograph's lens function. It computes the pixel's coordinates on the source photo.

SourcePixel = L(L^{-1}(NewPixel) · NewMatrix · $SourceMatrix^{-1}$)

But the resulting location cannot be directly used for obtaining the pixel's value as the alignment problems can occur due to inproper sampling. So multiple surrounding pixels should be used to avoid alias effects. And if the final pixel location is contained in more source images, the value of such pixel is computed as the weighted average of the colours of all the images. The source image's pixel's value gets less weight if the pixel is positioned near to the source image's border. Such approach removes sharp edges at image seams as can be seen in Figure 4. The image shown in Figure 4 is the preview image that does contain the visible seams. During the final rendering, the visible seams are removed.

Viewing one image data from another image's direction is relative, no matter if one of them is the newly generated image. The process is similar to the real photographing. One important think is that we have to choose a lens - either real or ideal, specified analytically.

5 Finding mutual direction

The user usually cannot set the direction precisely. The reason is that he cannot see the finest details of the images; the quality of the images during the real-time redrawing cannot be compared to the final rendered image. For the final rendering, the precise direction has to be found .

Let us examine the problem. The image can be moved around the unit sphere and turned around its axis. So the position can be changed in three ways. To see the complexity of this problem if it was to be solved by examining each possible combination of pixels. Let us assume that both the horizontal and vertical positions can be of so many states as the size of the image in pixels. And if the image grows, also rotation needs to be set finer because the incorrect rotation becomes more visible near the border. So the cardinality of rotation variable grows with the image size too. We have n^3 possible combinations. For each of the combination the quality has to be computed. The complexity of the quality computation is n^2. So the whole complexity will by n^5. To find the direction in an acceptable time, a better algorithm must be found. And a way to improve (decrease) the complexity does exist. The problem just need to be turned inside out. The outer loop passes all image pixels and for each one we look for the ideal position. Since now it seems that it does not change the computation time. But for each pixel, only two dimensions are searched, so the complexity changes to n^4. And from the pixel position it can be said that moving it can affect in translation or rotation of the image. Also, average values of translation and rotation can be used. This way, the program can place the image onto the right position. Then the final image can be rendered.

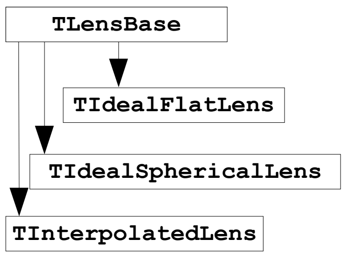
Here is a simplified algorithm. In fact, the centre of the rotation has to be found in another loop.

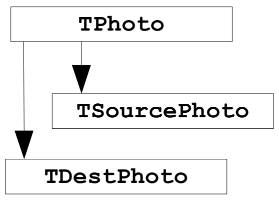
```
for(P = all image pixels)
{
 for(N = pixels close to P)
  Find pixel in the other images which suits best to P
 Translation += BestP - N
 Rotation += angle from N to BestP
       around Centre
 Count += 1
}
Translation /= Count
Rotation /= Count
```

6 Implementation

The whole project is implemented in C++. The main class is TPhoto. It holds the image data, reads and stores it to files. TPhoto also contains the direction matrix, focal length value and pointer to the TLensBase virtual class representing the photograph's projection lens. The lens has the same functionality no matter if it's used for a source photograph or for the rendered image. It can be used by virtual functions AngleToFilm and FilmToAngle. Their deal is discussed in the section Lens parameters. From TLensBase three real classes are derived: TIdealFlatLens and TIdealSphericalLens are mathematical models of these two ideal lenses. TInterpolatedLens reads measured data about a real lens and simulate it's behaviour.

Open-GL library for the scene visualization. It offers everything the project needs for drawing and by now it's accessible on nearly every personal computer.





7 Future work

Obviously, many aspects of the presented system can be improved. When some photographs are taken using different exposure or when the light conditions in the scene changes too fast, some photographs could be darker than the others. So brightness and contrast calibration for each photograph should be included and some lighting corrections of the lenses should be included as well.

In the sample panorama shown in Figure 4, grey gradual filter was used to suppress bright sky. Inexact balance of the filter caused the annoying dark stripe at the seam. Therefore, also important would be to implement a brightness gradient correction and vignetting correction.

Another question is how to get most of existing lenses measured and whether to include some asymmetric error corrections. The solution might be to implement a calibrating program, which would create a lens model from a reference photograph or a set of reference photographs.

8 Conclusions

This project demonstrates a novel way of merging photographs. The unit sphere projection enables easy implementation of lens distortion elimination. It also helps to create a comfortable user interface.

A demonstration application has been developed, which respects the physical parameters of the projection. And it can be used even by a photographer without much technical skill.

9 References

[1] Žára, J., Beneš, B., Felkel, P.: Moderní počítačová grafika, 1998, Computer Press, Czech Republic, ISBN 80-7226-049-9

[2] Canon Technology: Digital Cameras, 2005, Canon, Inc., USA (available at http://www.canon.com/technology/index.htm)

[3] Segal, M., Akeley, K., Frazier, C., Leech, J., Brown, P.: The OpenGL Graphics System: A Specification, Version 2.0, October 22, 2004, Silicon Graphics, Inc., USA (available at http://www.opengl.org)

[4] Watt, A., Watt, M.: Advanced Animation and Rendering Techniques, Addison-Wesley 1992USA, ISBN 0-201-54412-1