360° Full View Spherical Mosaic

Huang Wenfan U017865B

Huang Yehui U017844X

Rong Nan U018274R

Objective

- Full spherical mosaic 360 x 180.
- All images are taken with camera mounted on a tripod.
- Registration of Images taken with different exposure level.
- Automatic

Process Steps

- Image Registration
  - Fast Fourier Transform
  - Minimization of Summed Square Error
- Image Integration
  - Gamma Adjustment of the images
  - Blending of the overlapping images
- Image Viewing
  - Generating The Whole Mapping Texture Image
  - Realizing Spherical Texture Mapping by Using OpenGL

Image Registration

Registration Methods	Pros.	Cons.
Fast Fourier Transform	• Fast.	• Not accurate enough…
Feature Tracking	• Not sensitive for illumination change. • Quite accurate	• Require presence of good features. • Aperture Problem.
Minimization of summed Square Error	• General. • Quite accurate.	• Require a good initial guess to avoid local minimum. • Require no big change in lighting condition

Image Registration

- Fast Fourier Transform

- Given 2 N x M Images f_1, f_2 .
- If $f_1(x, y) = f_2(x - x_0, y - y_0)$
  - $f_1(x, y) \xrightarrow{\text{DFT}} F_1(u, v)$
  - $f_2(x, y) \xrightarrow{\text{DFT}} F_2(u, v)$
  - $F_1(u, v) = \exp[-i2\pi(u*x_0/N + v*y_0/M)]\, F_2(u, v)$
- $[F_1(u, v) \times F_2^*(u, v)] \;/\; |[F_1(u, v) \times F_2^*(u, v)]|$

  $= \exp[-i\, 2\pi\, (u*x_0/N + v*y_0/M)]$
- The Inverse Fourier Transform of $\exp[-i2\pi(u*x_0/N + v*y_0/M)]$ is an image whose maximum intensity is located at (x_0, y_0) if x_0, y_0 are both positive.

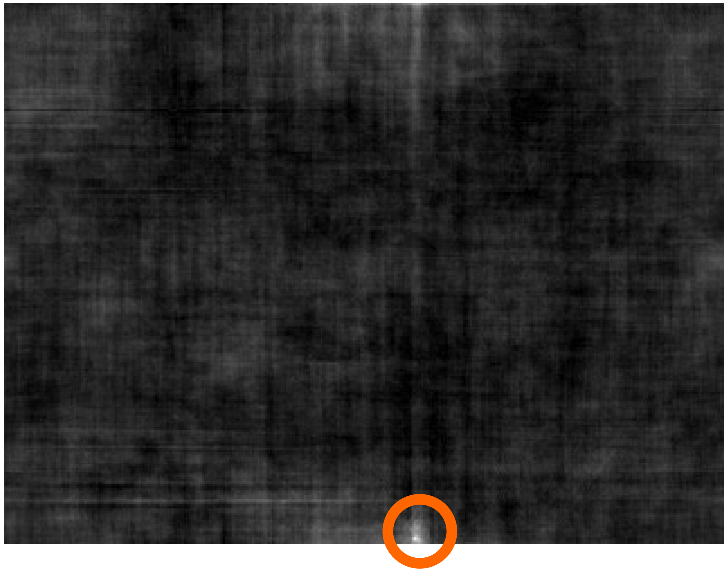
Image Registration

- Fast Fourier Transform

Example

Image f1

Image f2



IDFT

Image Registration

- Minimization of SSE

- Summed squared error of overlapping area of two images

$$E = \sum_i [I'(x'_i, y'_i) - I(x_i, y_i)]^2 = \sum_i e_i^2$$

- Find a 3 x 3 Matrix M transform (x, y) to (x', y') such that the summed square error is minimized.
- The summed squared error solution is given by Levenberg-Marquardt Iterative Algorithm
  - http://mathworld.wolfram.com/Levenberg-MarquardtMethod.html

Image Registration

- Minimization of SSE

- Our first try for spherical mosaicing.
- Transform the pictures from spatial coordinates to polar coordinates on the sphere. Then the relation-ship between 2 images become pure 2D translation.
  - For each pixel in the polar picture, we find the intensity by
    - X = tan(Elevation) * focalLength;
    - Y = focalLength * tan(Elevation)/cos(Azimuth);
- Find 2D translation minimizing the summed square error by L & M Iterative Algorithm.

Image Registration
- Minimization of SSE

- Find the partial derivative of the error function with respect to *Tx* and *Ty*
  - $e_i = I_r(x + Tx, y + Ty) - I_t(x, y)$
  - $e_i = I_r(Tx, Ty) + Tx * \partial I_r/\partial x + Ty * \partial I_r/\partial y - I_t(x, y)$
  - $J_i = | \partial I_r/\partial x \quad \partial I_r/\partial y |$
- Find the solution ΔT to update the translation.
  - $(\sum J_i^T J_i + \lambda I) * \Delta T = - \sum J_i^T e_i$ (1)
  - In the actual implementation $\lambda = 0$ already gives good result.

Image Registration

- Minimization of SSE

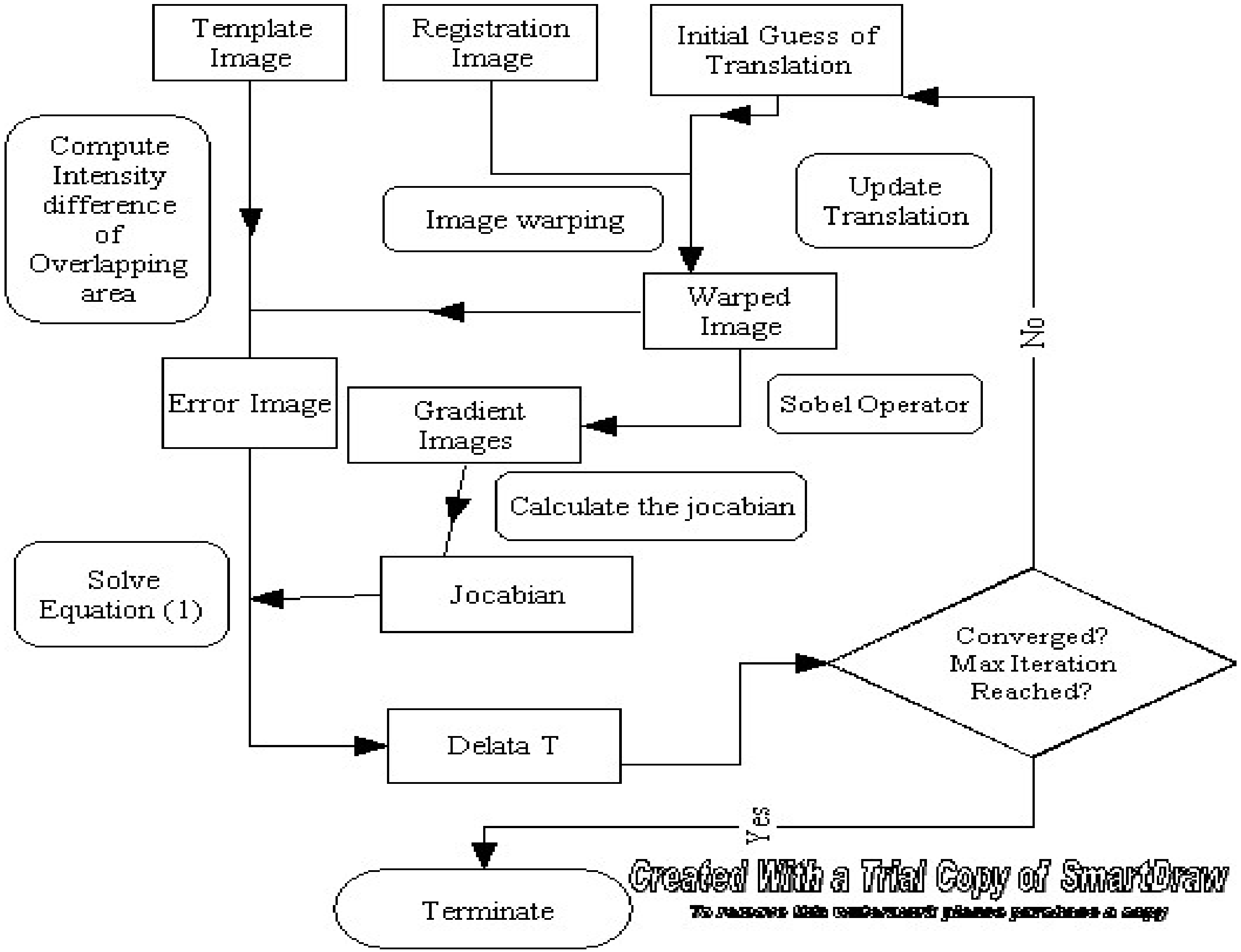
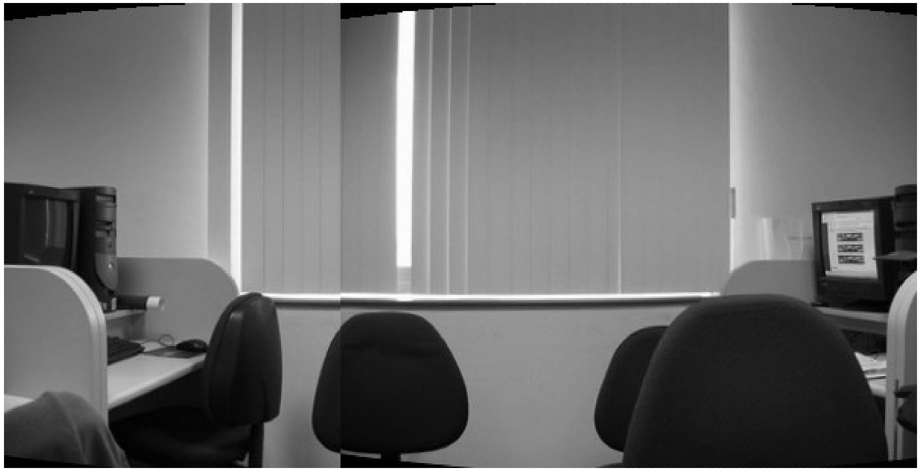


Image Registration

- Minimization of SSE

- This approach works fine for images on the equator.



- However, it’s quite difficult to register images shot with a tilted camera with different panning angles.
- How to generate a full view spherical mosaic?

Image Registration

- Minimization of SSE

- Looking for the relative camera rotation matrix to register two images. (R. Szeliski, 1997)

- A point P in 3D space is projected to position x on the image plane of a camera rotated at the origin by:
  - $\mathbf{x} \sim \mathbf{VRp}$, where V is the perspective projection matrix. R is a rotation matrix.
  - $\mathbf{p} \sim \mathbf{R}^{-1}\mathbf{V}^{-1}\mathbf{x}$, where p is a ray in 3D space.

- If two cameras with different rotation angles shot the same point in 3D, the projection of this point on the two images planes is related by.
  - $\mathbf{x}_2 \sim \mathbf{V}\,\mathbf{R}_2\,\mathbf{R}_1^{-1}\mathbf{V}^{-1}\mathbf{x}_1$
  - $\mathbf{x}_2 \sim \mathbf{V}\,\mathbf{R}_2\,\mathbf{V}^{-1}\mathbf{x}_1$, if we consider camera 1 as reference frame

Image Registration

- Minimization of SSE

- Then the relative rotation matrix **R** is found by incremental updating using LM algorithm.
- **R** is updated by
  - **R**(Ω) =

$$\begin{bmatrix} 1 & -wZ & wY \\ wZ & 1 & -wX \\ -wY & wX & 1 \end{bmatrix}$$

  - J_i =

$$\begin{bmatrix} \partial I_r/\partial x & \partial I_r/\partial y \end{bmatrix} \times \begin{bmatrix} -xy/f & f + x^{\wedge}2/f & -y \\ -f - y^{\wedge}2/f & xy/f & x \end{bmatrix}$$

- Following same algorithm presented in translation.
- We also worked out the formula for 2 rotation angles, however, the running result is not that good compared to 3 angles.

Image Registration

- Minimization of SSE

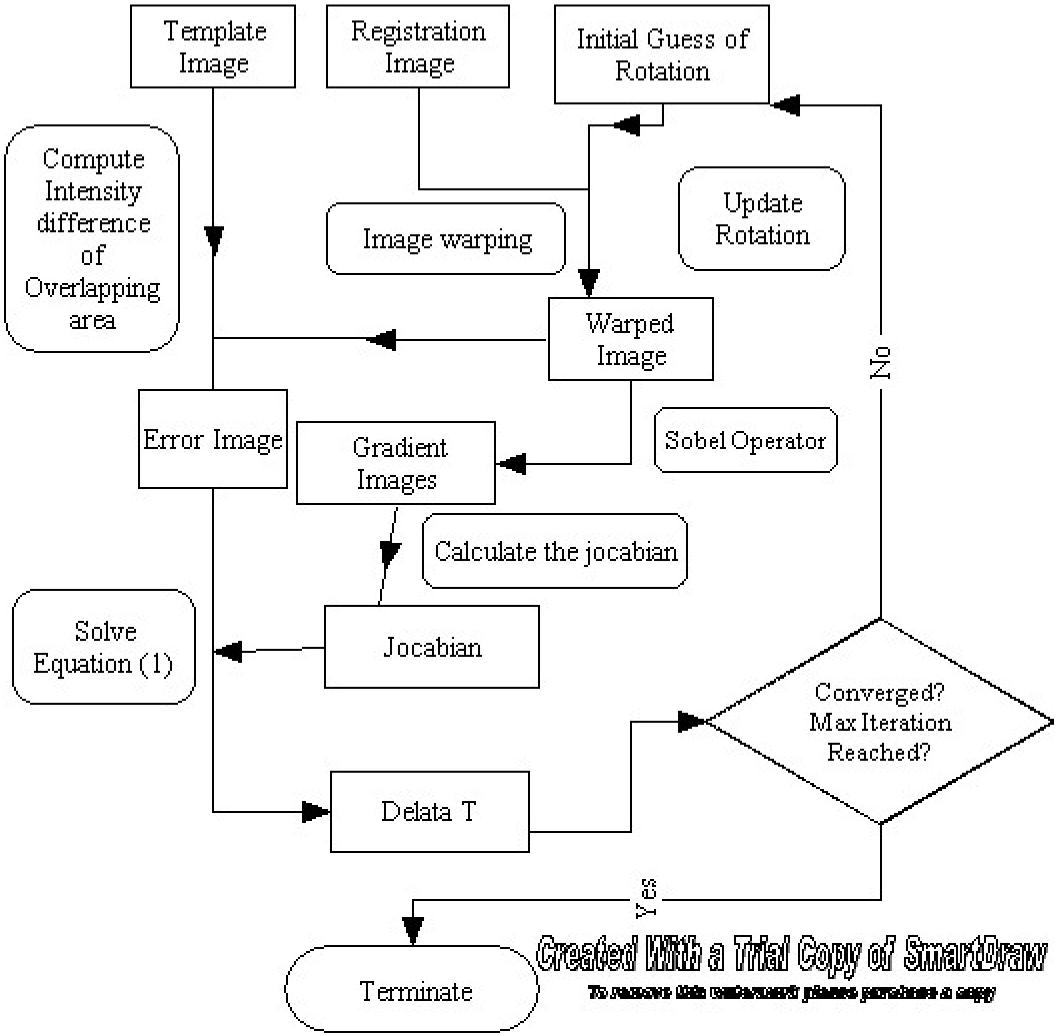


Image Registration

- Minimization of SSE

- Local Minimum Problem
  - Using Fast Fourier Transform to Provide a good initial guess.
  - Using Image pyramid.
  - We use the combination of these two methods.

Image Registration

- Minimization of SSE

- Images are only Registered with adjacent images.
- From Local Rotation Matrices to Global Rotation Matrices
  - Use only one reference frame $\mathbf{I}_r$
    - $\mathbf{I}_k \sim \mathbf{V}\,\mathbf{R}_k\,\mathbf{V}^{-1}\mathbf{I}_r$
    - $\mathbf{I}_{k+1} \sim \mathbf{V}\,\mathbf{R}_{(k+1)>k}\,\mathbf{V}^{-1}\mathbf{I}_k$
    - $\mathbf{I}_{k+1} \sim \mathbf{V}\,\mathbf{R}_{(k+1)>k}\,\mathbf{R}_k\,\mathbf{V}^{-1}\mathbf{I}_r$

Image Registration

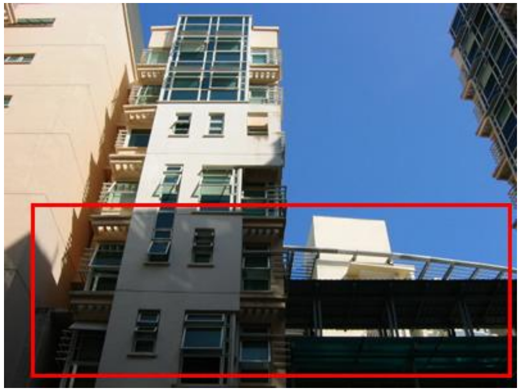
- Minimization of SSE

- Problems and solutions
  - Accumulated error
    - Local Rotation Matrices are not error free.
    - Only the Accumulated error of global rotation matrices of images on the equator are calculated.
    - Distribute evenly to all global rotation Matrices on the equator.
  - Focal length Estimation
    - Closing the gap of images on the equator
  - Registration error due to Intensity changes of overlapping area
    - Due to different lighting condition or exposure level.
    - Using Fast Fourier Transform + Gamma Correction to adjust the intensity of two overlapping images

Image Registration

- Gamma correction

- Problem: Difficult to achieve accurate registration due to obvious intensity changes.

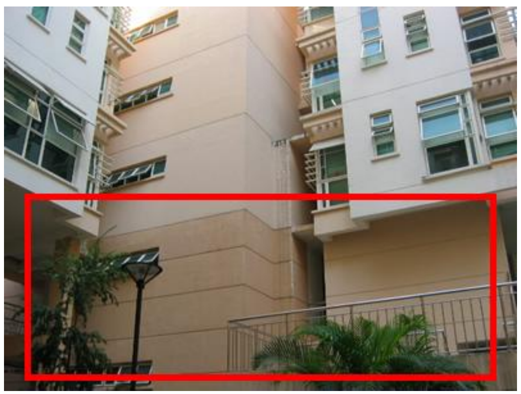


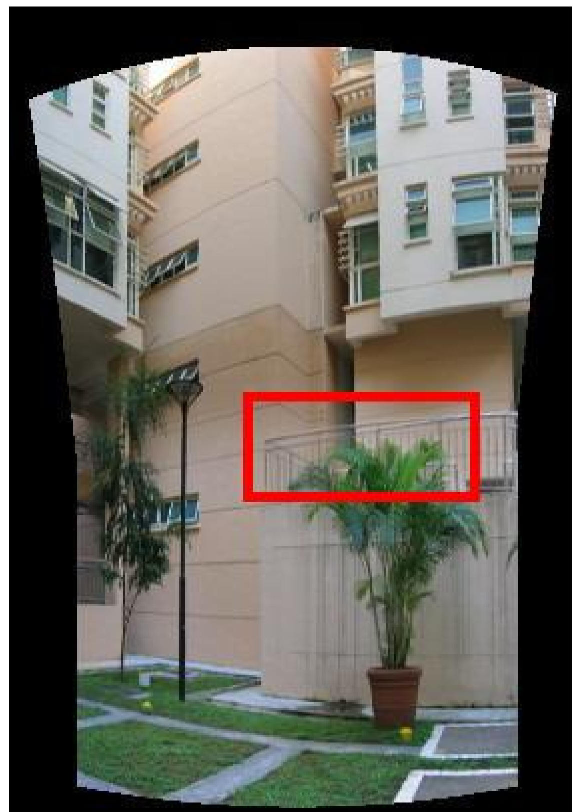
•Relative gamma value: 2.89

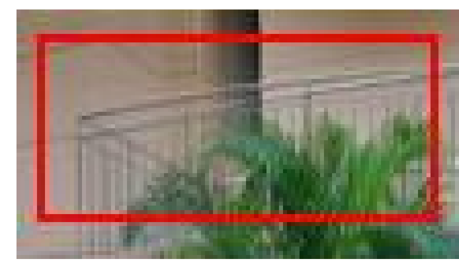
Image Registration

- Gamma Correction

- Example result without gamma correction.







Relative gamma value: 1.627

Image Registration

- Gamma Correction

- Solution: Adjust image intensity by gamma correction before final registration.

- Steps:
  - Use FFT to approximate translation between the 2 images. (e.g. img1(template) and img2)
  - Find relative gamma value base on their average intensity in the overlapping area.
    - Gamma = log (avgI2)/log (avgI1)
  - Apply gamma correction to Img2.
    - I2' = I2 ^ (1/gamma)
  - Repeat until gamma within threshold. (0.99 to 1.01)

Image Registration
- Gamma Correction

- Example result after gamma correction:

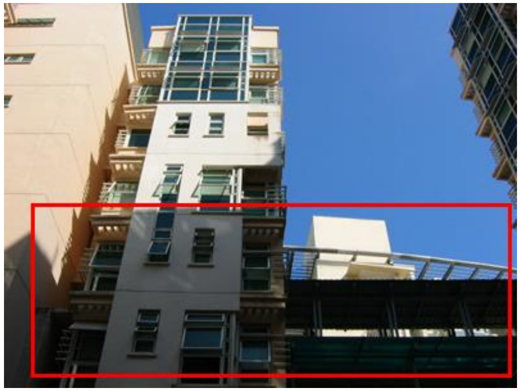


Image Registration

- Gamma Correction

- Example results after gamma correction:

Before:

After:

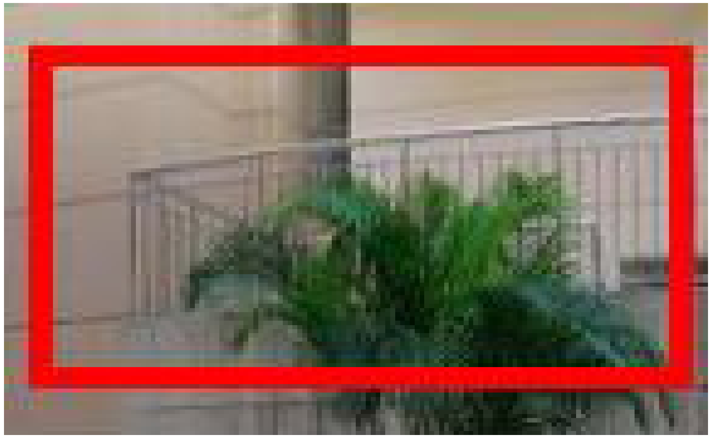


Image Integration

- Gamma Distribution

- Which image set should we choose for final output texture?
  - Original images
  - Images being gamma corrected

- There are obvious intensity changes between original images. This may cause sudden change of brightness in the final output.

- We would like to solve this problem first before blending.

Image Integration

- Gamma Distribution

- Sudden changes in brightness in final texture if *original images* are used for final texture generation
- This is not desired.

Image Integration

- Gamma Distribution

- Some part of the final texture becomes pale if *gamma corrected images* are used.

- This is also not desired.

Image Integration
- Gamma Distribution

- We would like the intensity changes to be distributed averagely over the final texture while the coloring effect won't be affected too much.

- Solution: gamma distribution
- Basic idea:

  For each image, find a gamma value to minimize its relative gamma value to all the neighbors. The process is repeated iteratively until a stable state is reached.

Image Integration
- Gamma Distribution

- Steps:
  - There is an absolute gamma value for each image
  - For each img i:
    - Find its relative gamma values to all its neighbors. (e.g. G1, G2 … Gn)
    - Find the geometrical mean of these values
      - G0 = (G1*G2*…*Gn)^ (1/n)
    - Update img I's absolute gamma value and record G0
    - Repeat until all G0 within threshold. (1.25 ~ 0.8).
  - Use accumulated G0 of each image to do the final gamma correction.

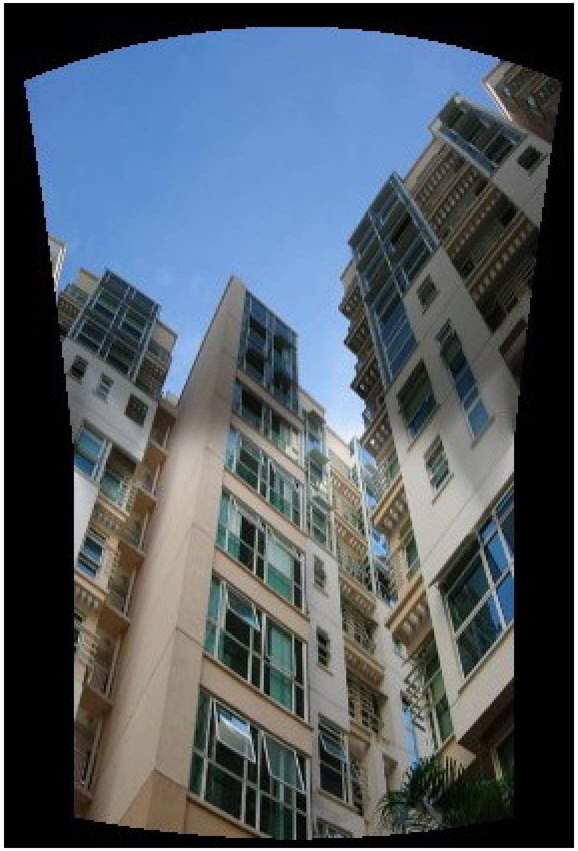
Image Integration - Gamma Distribution

- Alternative adjusting function:
  - $G0 = (G_{max} * G_{min})$ ^(1/2)
  - $G0 = (G_{max} * G_{min})$ ^(1/4)
    - ¼ used instead of ½ to avoid over-adjustment of gamma

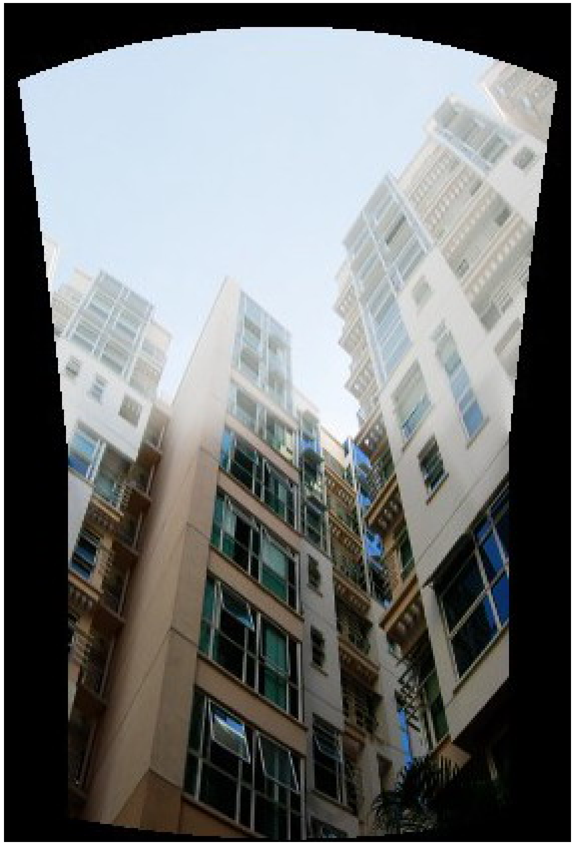
  - We find G0 used produce best effect although the difference is not very significant.

Image Integration
- Gamma Distribution

Original Images used



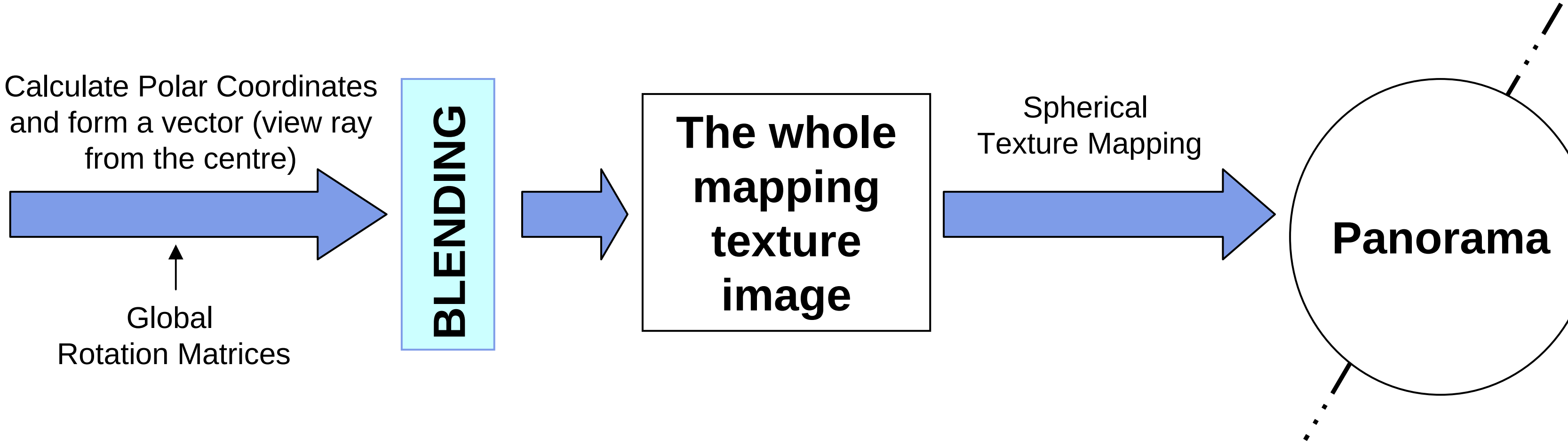
Gamma corrected Images used



With Gamma Distribution

Image Viewing

- Generating The Whole Mapping Texture Image
- Realizing Spherical Texture Mapping by Using OpenGL



Generating The Whole Mapping Texture Image

- Step 1: For each pixel (i, j) in the output mapping image, calculate the polar coordinates (theta, phi)

1. Normalize coordinates

   x = 2 * i / width - 1
   y = 2 * j / height - 1

   (x,y) each ranging from -1 to 1

2. Derive polar coordinates

   theta = x * pi
   phi = y * (pi / 2)

   theta $\in$ [-pi, pi]
   phi $\in$ [-pi/2, pi/2]

Generating The Whole Mapping Texture Image (Cont.)

- Step 2: Compute corresponding 3D position vector (view ray from the centre) point on unit sphere

$$p = \begin{pmatrix} \cos(phi) * \cos(theta) \\ \sin(phi) \\ \cos(phi) * \sin(theta) \end{pmatrix}$$

- Step 3: for each p, determine its mapping into each image k using $p' = VR_k p$;

  Where p' is 2D point in the image,

  R_k is global rotation matrices

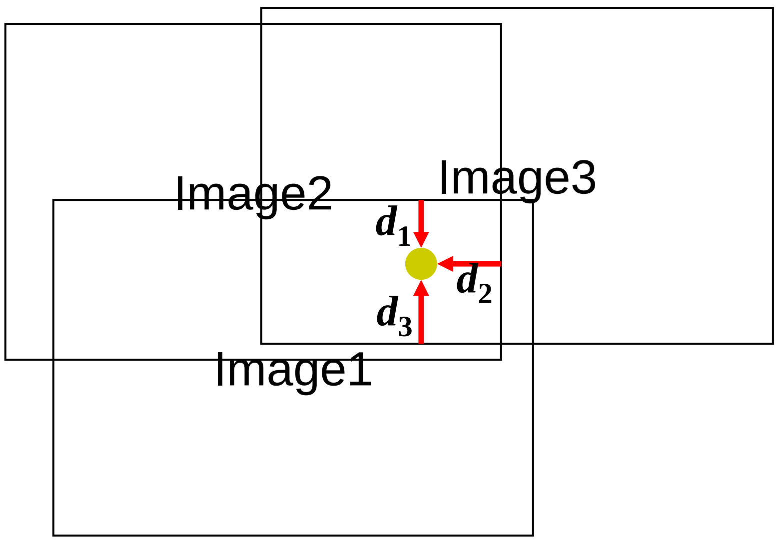
$$V = \begin{pmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Generating The Whole Mapping Texture Image (Cont.)

- Step 4: Blending

  Use simple heuristic to get good result:

  - Every pixel is weighted with the distance to the closest image boundary to the n^th power



$$I_{final}(p) = \frac{d_1^{\boldsymbol{n}} I_1(p') + d_2^{\boldsymbol{n}} I_2(p') + d_3^{\boldsymbol{n}} I_3(p')}{d_1^{\boldsymbol{n}} + d_2^{\boldsymbol{n}} + d_3^{\boldsymbol{n}}}$$

where n = 3

Note: If many pictures cover the same pixel, we only consider 3 of them which have higher weight.

without blending

averaging overlapped intensity

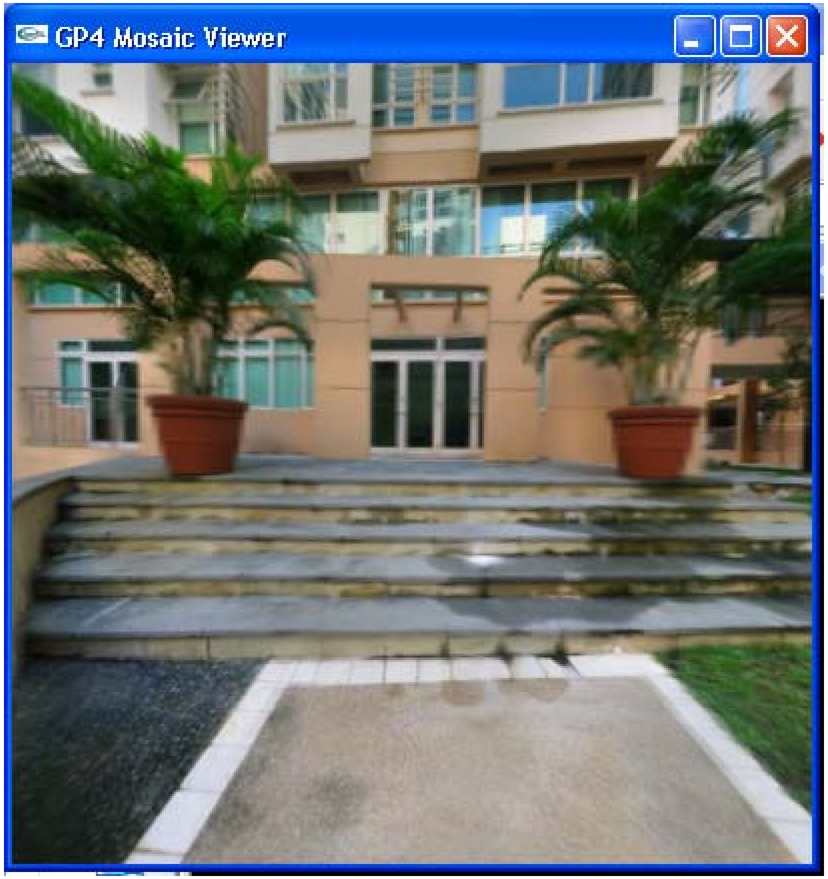
with specific blending

Realizing Spherical Texture Mapping by Using OpenGL

- Create one sphere
- Load the output image
- Performing texture mapping by using openGL functions.
- Design some simple user-friendly interface

Result From 68 images

Result



Future improvement

- Better Error Distribution
- Better blending funcion

Reference

- R. Szeliski... : Creating full view panoramic Image Mosaic and Environment map.
- R. Szeliski... : Video Mosaics for virtual environment.
- M. G. Gonzalez... : Improved Video Mosaic Construction by Accumulated Alignment Error Distribution
- Paul Bourke: http://astronomy.swin.edu.au/~pbourke/projection/spheretexture/