RASCAL Tools User Guide

August 1, 1999

Tomoo Mitsunaga

CAVE lab. , Columbia University

Contents

1. About this document

1.1 Purpose

This document is for readers who are trying to do experiments of radiance recovery with **RASCAL** tools (pre-compiled programs). For such readers, this document describes how to use **RASCAL** tools and some tips on using **RASCAL** tools.

§ About RASCAL

RASCAL is a software package for RAdiometric Self CALibration that has been developed at the CAVE lab at Columbia University in the City of New York. This software package includes source codes of C++ and pre-compiled binary programs for MS-Windows. With this software package, you can ...

- Calibrate the radiometric response function of your imaging system without reference objects,
- Make a high dynamic range radiance image from a set of differently exposed images.

The detail of the algorithm is described in the following document.

"Radiometric Self Calibration", T.Mitsunaga and S.K.Nayar, CVPR99, (1999).

1.2 History

Dec. 03 1998:	Created.
Dec. 17 1998:	Updated "How to use / **rrcombine** " to follow recent improvement of **rrcombine.**
Apr. 05 1999:	Updated "How to use / **rrselect** " to follow adding auto-trimming mode to **rrselect**.
Jun. 15 1999:	Updated "2.1 Introduction to radiance recovery ".
August 1, 1999:	Changed the name of this software package to "RASCAL".

2. Outline of radiance recovery

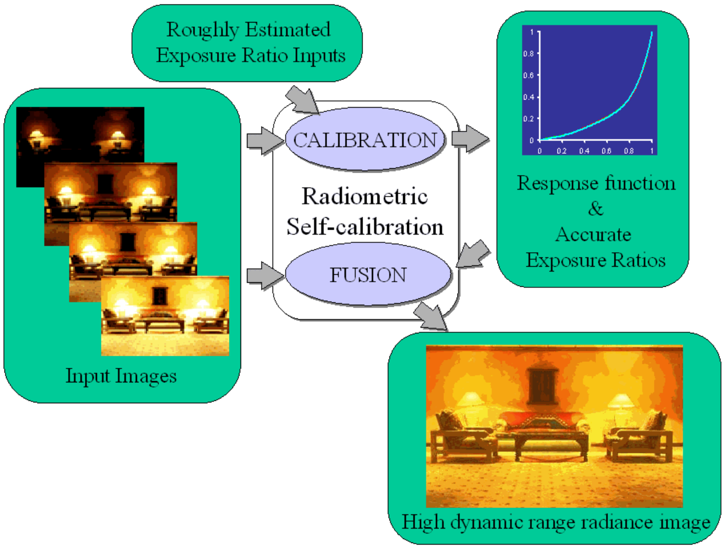
2.1 Introduction to radiance recovery

§ Objective

Many computer vision algorithms are based on the assumption that image brightness is proportional to scene radiance. However, an imaging system has a radiometric response function that is seldom linear. Our objective is to develop methods for computing the response function of an imaging system from multiple images of an arbitrary scene taken using the imaging system. The sensor could be a video camera, photographic camera, a digital still camera, or any other imaging system. In addressing this radiometric calibration problem, we address two sub-problems, namely, computing the response function of the complete imaging system and computing a high dynamic range radiance image from the set of captured images.

§ Radiometric Self-Calibration Algorithm

To achieve the above goals, we have developed a radiometric self-calibration algorithm. The algorithm computes the response function of the imaging system from a plurality of images of an arbitrary scene taken using different exposures. Since, in any imaging system it is difficult to obtain precise values for the exposures (for instance, the aperture setting on a lens only gives an approximate exposure value), we have developed an algorithm that takes rough exposure estimates and the corresponding images to compute the response function of the system, the precise exposure values and a single high dynamic range radiance image of the scene. The basic flow of data in our algorithm is illustrated below.



2.2 What you prepare

§ Differently exposed images

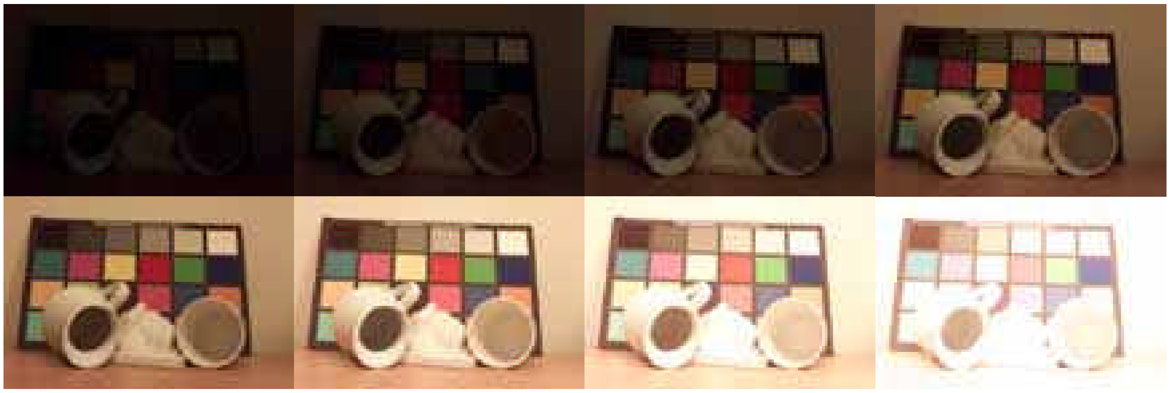
You have to prepare a set of images of a still scene to do radiance recovery with **RASCAL** tools.

The images must be captured with different exposure setting. Exposure can be changed with either shutter speed or aperture. Make sure that the set of images must include both sufficiently dark and sufficiently bright images.

It is also important to minimize noise of images which you want to recover radiance. In general, video cameras have much greater noise than films. So, reducing noise is especially important for video cameras. It is better to apply temporal averaging to a still scene (100 times or more) when you use video images.

RASCAL currently support **BMP** and **PPM** file format as input images. You can also use **PPM16** format which is a 16bit extension of **PPM** format. However, **PPM16** is used only in **RASCAL.**

The following images are an example of differently exposed image set.



§ Rough estimate of exposure ratios

The radiance recovery program also needs information how exposure ratios between images are. Rough estimates of the exposure ratios such as reading F-number of the aperture or shutter speed of the camera are sufficient for accurate radiance recovery by **RASCAL**.

You can calculate the exposure ratio between two images from F-number and shutter speed by using the following equations.

$$(\text{exposure ratio})_{1,2} = (\text{exposure})_1 / (\text{exposure})_2$$

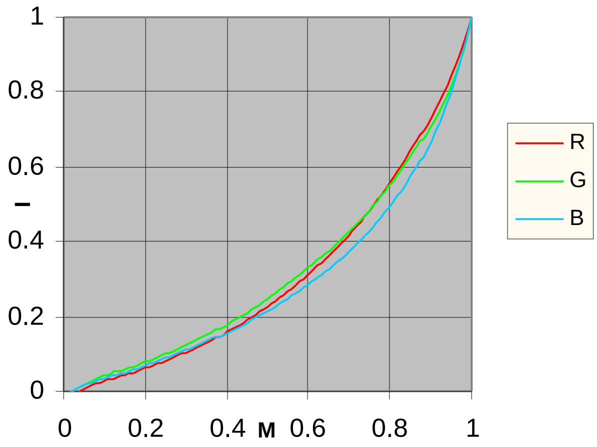
$$(\text{exposure}) = \left(\frac{1}{(\text{F - number})} \right)^2 \cdot (\text{shutter speed})$$

For example, if images are captured with F11, F8, F5.6 and F4 with fixed shutter speed, exposure ratios become 0.53, 0.49 and 0.51. However, in this case, **RASCAL** radiance recovery still works well with more rough estimates like as 0.5, 0.5 and 0.5.

2.3 What you get

§ Radiometric response curve

By using **RASCAL** tools, you obtain the radiometric response curve of the imaging system. The following figure shows an example of radiometric response curve which has been computed by using **RASCAL** tools. M expresses the measured pixel value and I expresses the ideal (calibrated) pixel value. Pixel values are normalized from 0 to 1.

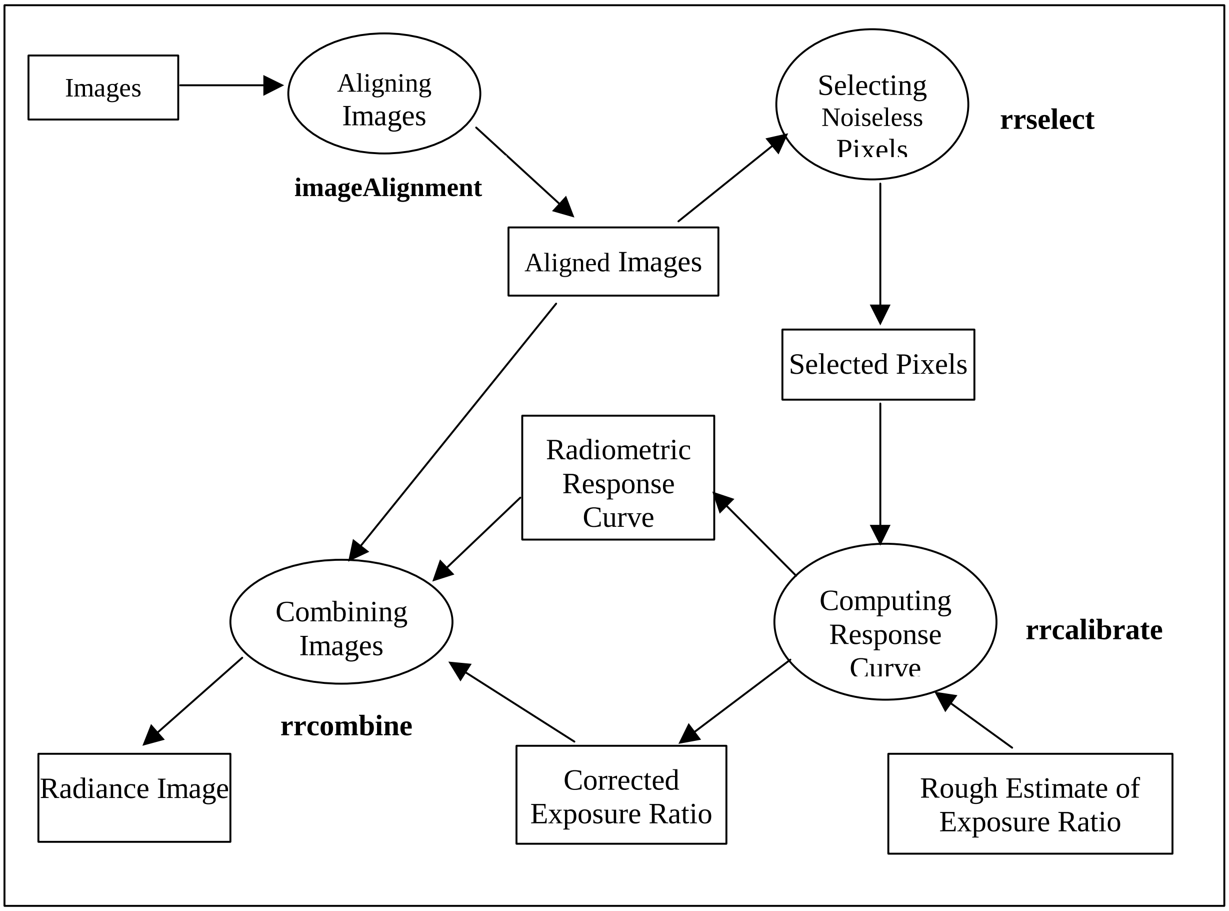


§ High dynamic range radiance image

After obtaining the response curve, you can combine the calibrated images into one. The combined image becomes linear to the scene radiance and its dynamic range may become several-bit greater than the dynamic range of original images.

The following image is an example of high dynamic range image which was computed by using RASCAL tool. (Color balancing was not applied.)

2.4 Outline of process with RASCAL tools



The above figure shows the process flow of radiance recovery with **RASCAL** tools. A Rectangle expresses a stored data and a oval expresses a process. **RASCAL** tools consists of five tools (**rrselect**, **rrcalibrate**, **rrcombine**, **imageAlignment** and **radianceView**). Each process in the above figure corresponds to one of **RASCAL** tools (**rrselect**, **rrcalibrate**, **rrcombine** and **imageAlignment**). The rest tool **-- radianceView** is a viewer to display a high dynamic range radiance image which is finally made by the processes as shown in the above figure.

§ Aligning images -- imageAlignment

It must be guaranteed that all pixels at same position in images capture same scene radiance. Otherwise, radiance recovery computation may not be accurate. Therefore, you maybe need to correct misalignment of images by using **imageAlignment** before computing main process of radiance recovery.

§ Selecting noiseless pixels -- rrselect

It is also important for accurate computing of radiance recovery that image noises are previously minimized. **rrselect** selects and outputs pixels which are evaluated as noiseless pixels.

§ Computing radiometric response curve -- rrcalibrate

The noiseless pixel data which are selected by **rrselect** are used to compute the radiometric response curve of the imaging system by **rrcalibrate**. **rrcalibrate** requires selected pixel data and rough estimates of exposure ratios as input. And it outputs estimated response curve and corrected exposure ratios.

§ Combining images into one -- rrcombine

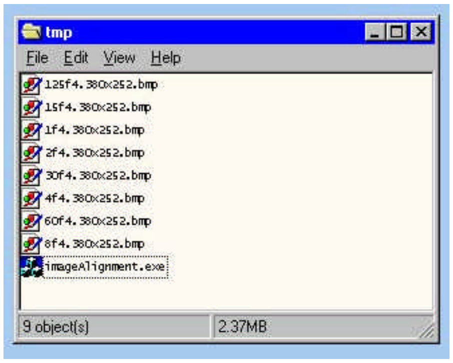
rrcombine combines images into one high dynamic range radiance image. This tool requires response curves, exposure ratios and images. After making a radiance image, you can monitor the radiance image by using **radianceView**.

3. How to use -- step by step

3.1 imageAlignment

§ Preparation

Prepare your images.



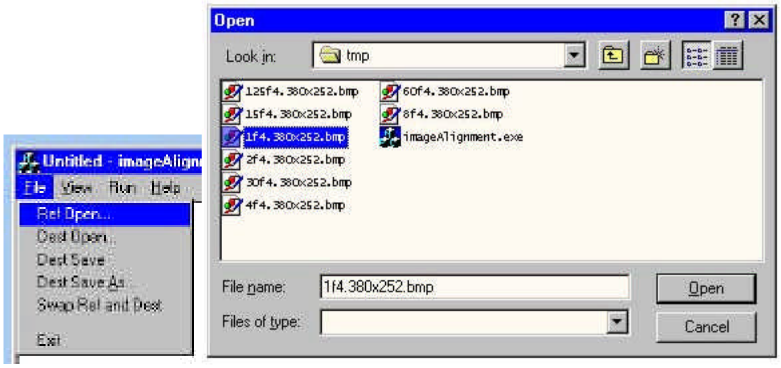
§ Execution

Execute **imageAlignment.exe** by double-clicking the icon.

§ Loading a reference image

Open a reference image -- use menu "**File / Ref Open ...**".

The position of the reference image is used as reference of alignment.



File name: 1f4.380x252.bmp

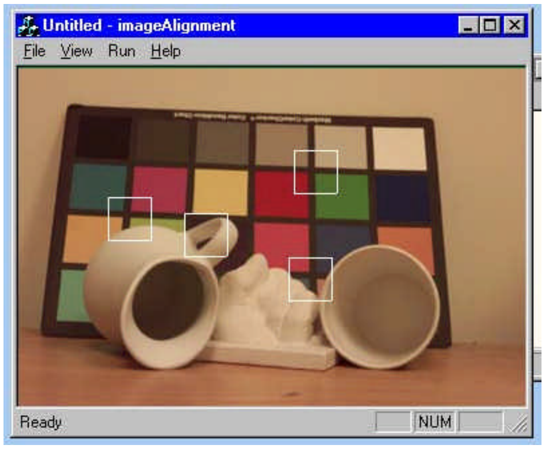
§ Setting search rectangles

Set search rectangles.

to add a new rectangle ... Click Ctrl - left (mouse) button.
to remove a rectangle ... Click Ctrl - right button on the rectangle.

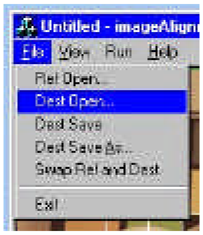
to select a rectangle ... Click left button on the rectangle.
to move a selected rectangle ... Click left button once on the rectangle, then drag it.

imageAlignment uses a template matching based algorithm. So, it is better to place rectangles to where it is easy to find corresponding points. The following figure shows as example of placing rectangles on a reference image.

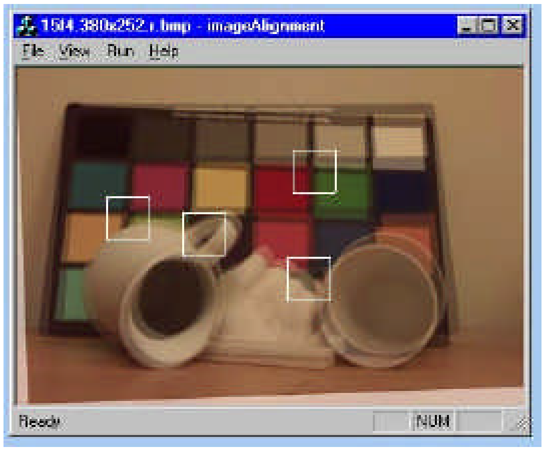


§ Loading a destination image

Open a destination image, which is the image to be aligned to the reference image, -- use menu "**File / Dest Open …**".



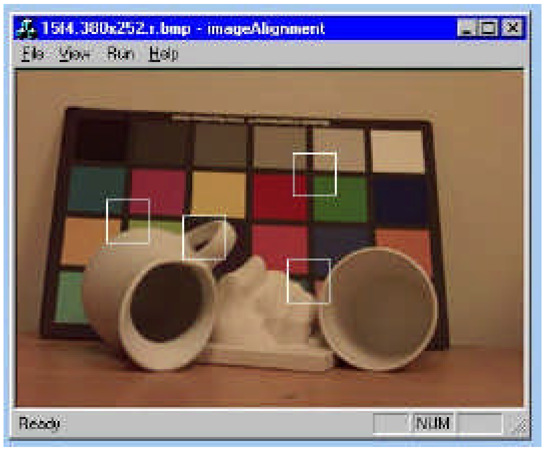
After opening the destination image, the blended image of the reference and the destination image is displayed on the view window. The following figure shows an example of the view after opening a destination image.



§ Executing alignment

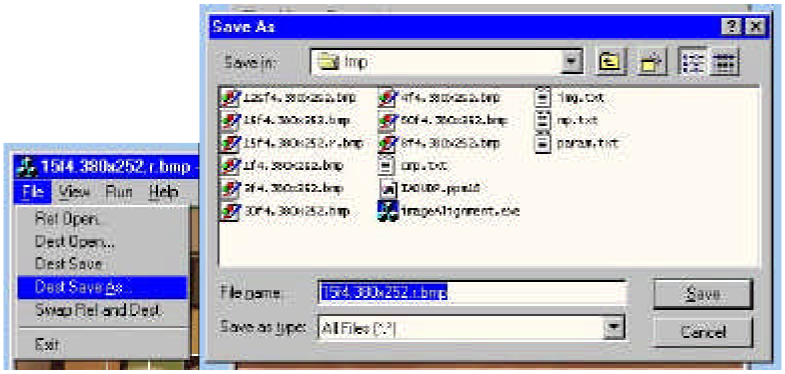
After placing rectangles, do alignment -- use menu "**Run / Alignment**".

The following figure shows the blended view after alignment of the destination image which has been showed in the previous figure.



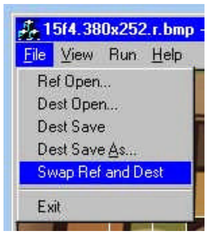
§ Saving a destination image

Save the aligned destination image -- use menu "**File / Dest Save**" or "**File / Dest Save As ...**"



§ Going to the next image pair

For the next reference and destination pair, swap them. -- use menu "**File / Swap Ref and Dest**". Previous destination image becomes new reference image of the next pair.



Then, open a new destination image. After that, repeat process that has been described so far with the new image pair.

Since **imageAlignment** uses template matching based algorithm, it is better for good result to take two images which have similar brightness each other as a pair of a reference and a destination images. For example, when you have four images im[0], im[1], im[2] and im[3] with brightness increasing (or decreasing) order, we recommend the following way.

(1) im[0] => reference, im[1] => destination

(2) im[1] => reference (Use "**File** / **Swap Ref and Dest**"), im[2] => destination

(3) im[2] => reference (Use "**File** / **Swap Ref and Dest**"), im[3] => destination

3.2 rrselect

§ Preparation

In addition to aligned images which have been made, prepare two text files -- image list file (imglist in the below figure) and parameter file (param in the below figure) -- as inputs to **rrselect.**

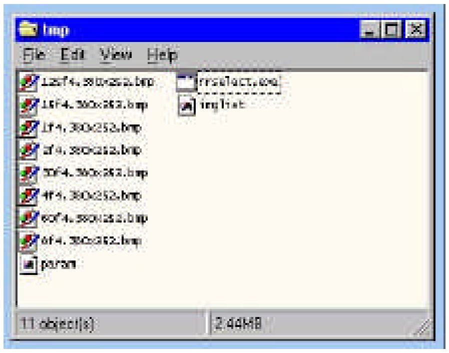


Image list file specifies number of images and their file names. The format of image list file is as followings.

File name of this file.

```
id   imglist     RRFileListData
creator     nobody
numSource   0
numFiles    8
FileName    0    125f4.380x252.bmp
FileName    1    60f4.380x252.bmp
FileName    2    30f4.380x252.bmp
FileName    3    15f4.380x252.bmp
FileName    4    8f4.380x252.bmp
FileName    5    4f4.380x252.bmp
FileName    6    2f4.380x252.bmp
FileName    7    1f4.380x252.bmp
```

Number of images

File names of images. They must be aligned with order from the darkest to the brightest.

Images must be numbered from 0 to (numFiles) -1.

Parameter file defines setting of **rrselect**. The format of parameter file is as followings.

File name of this file.

```
id    param RRSelectParamData
creator     nobody
numSource   0
kernelSize  3
trimRatio   1.000000e+000
saturationLevel  9.900000e-001
noiseLevel 0.100000e-001
maxSamplingNumber     1000
```

rrselect has 5 parameters.
The order of parameter in a parameter file must not be changed.

Parameters of **rrselect** are explained in the followings.

int **kernelSize**

Size of spatial averaging filter. default value is 3 (=3x3).

Increasing **kernelSize** **->** Data selecting becomes severe. But number of selected data decreases.

double **trimRatio**

Ratio of trimming size to the original image size. This trimming is useful to reject side area which may have vegnetting distortion. If trimming ratio is 0.7, the size of trimmed rectangle is 70% of the original one. However, the aspect ratio does not change. default value 1. that means no trimming.

Decreasing **trimRatio** **->** pixels in center area are mainly selected. So, data becomes more safe, but the number of data decreases.

If **trimRatio** is set to 0.0, this means that **auto-trimming mode** is selected. In **auto-trimming mode**, trimming size is automatically adjusted by checking amount of vegnetting effect in the input images. Another difference from manual mode is that the trimming shape is not rectangle but circle. In the current version of **rrselect**, you cannot use both manual rectangular trimming and automatic circular trimming simultaneously.

double **saturationLevel**

Upper clipping level of pixel value. This must be under 1 if you want to reject saturated pixels. default value is 0.990.

Decreasing **saturationLevel** -> Some camera outputs are clipped at a level under 0.990. You must decrease this parameter if you find such clipping. However, too low setting of saturation level might bring lack of data around 1.

double **noiseLevel**

Lower clipping level of pixel value. This must be over 0 if you want to reject blackout and noisy pixels. default value is 0.010.

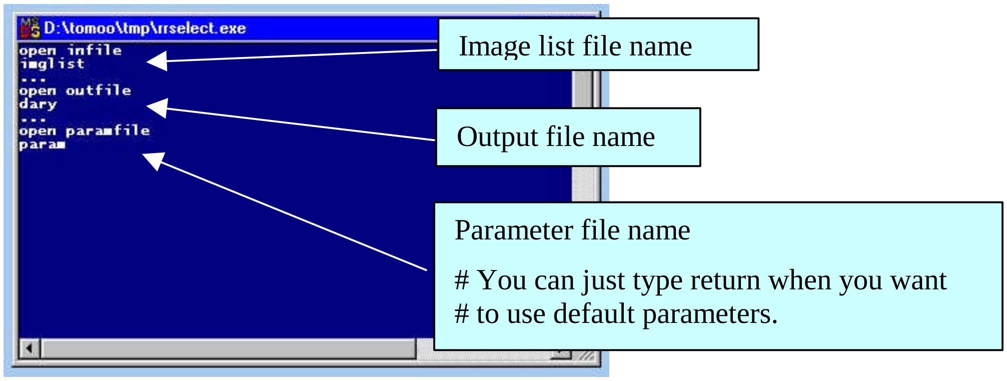
Increasing **noiseLevel ->** Data selecting becomes soft. So, the number of selected data increase. However, selected data becomes noisy.

int **maxSamplingNumber**

Upper limit of number of samples from 1 image pair. This parameter is for the sake of computational cost. default value is 1000.

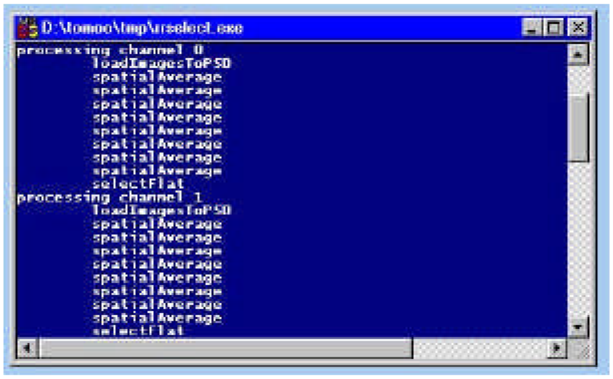
§ Execution

Execute **rrselect.exe** with double-clicking the icon. Then type asked file names.


D:\tomoo\tmp\rrselect.exe
Image list file name
Output file name
Parameter file name
# You can just type return when you want
# to use default parameters.

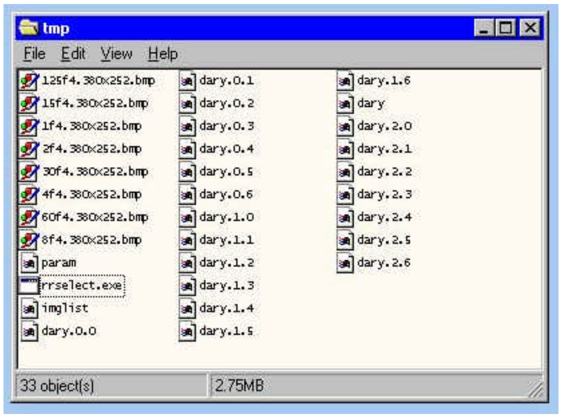
```
open infile
imglist
...
open outfile
dary
...
open paramfile
param
```

Then **rrselect** starts to compute.



```
processing channel 0
         loadImagesToPSD
         spatialAverage
         spatialAverage
         spatialAverage
         spatialAverage
         spatialAverage
         spatialAverage
         spatialAverage
         spatialAverage
         selectFlat
processing channel 1
         loadImagesToPSD
         spatialAverage
         spatialAverage
         spatialAverage
         spatialAverage
         spatialAverage
         spatialAverage
         spatialAverage
         spatialAverage
         selectFlat
```

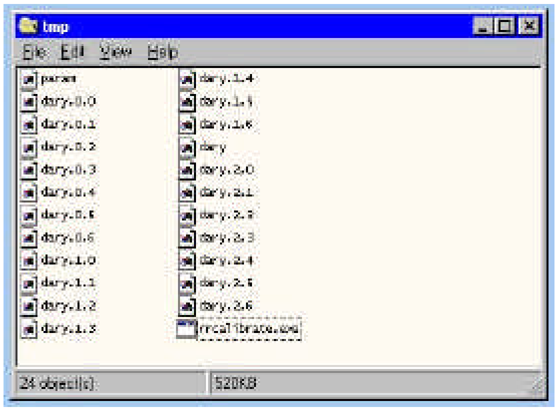
If the number of input images is N, one <outfile> (dary in the following figure) and N-1 <outfile>.c.m (dary.c.m in the following figure) are generated by **rrselect**. Here, c is the number of channel (R, G and B, respectively) and m is number of image pair (m=0 is the pair of image[0] and image[1], and so on). <outfile> contains the filenames of <outfile>.c.m and <outfile>.c.m contain the actual data of selected pixels.



3.3 rrcalibrate

§ Preparation

Input data to **rrcalibrate** are selected pixel data files which **rrselect** outputted (dary and dary.*.* in the following figure) and parameter files for **rrcalibrate** (param in the following figure). Note this parameter file differs to previous parameter file which was for **rrselect**.



Parameter file defines setting of **rrcalibrate**. The format of parameter file is as followings.

File name of this file.

```
id    param RRCalibrateParamData
creator     nobody
numSource   0
maxOrder    5
numExposureRatios     7
exposureRatio    0    5.0e-001
exposureRatio    1    5.0e-001
exposureRatio    2    5.0e-001
exposureRatio    3    5.0e-001
exposureRatio    4    5.0e-001
exposureRatio    5    5.0e-001
exposureRatio    6    5.0e-001
convergenceLevel 1.0e-003
```

rrcalibrate has 4 kinds of parameters.

The order of parameters in a parameter file must not be changed.

Parameters of **rrcalibrate** are explained in the followings.

int **maxOrder**

Upper limit of the order of polynomials which approximates the radiometric response curve of a camera system. **rrcalibrate** finds a best-fit order of polynomial from 1 to **maxOrder**. Default value is 5.

double **exposureRatio**[(number of images) - 1]

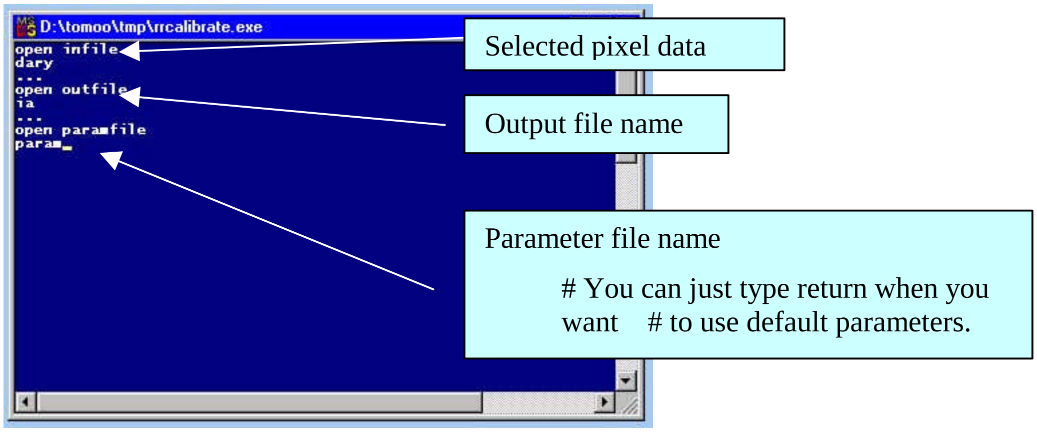
Initial rough estimates of exposure ratios of image pairs. Darker image pair should be assigned to earlier number. How to estimate an exposure ratio has been described in Section 2. Default value is 0.5 for every image pair.

double **convergenceLevel**

The criterion to judge whether computation has converged or not. Default value is 1.e-3.

§ Execution

Execute **rrcalibrate.exe**. And type asked file names.



```
open infile
dary
...
open outfile
ia
...
open paramfile
param_
```

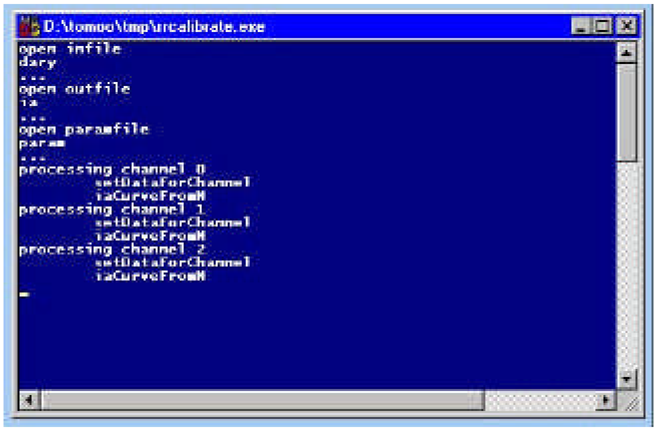
Selected pixel data

Output file name

Parameter file name

# You can just type return when you want # to use default parameters.

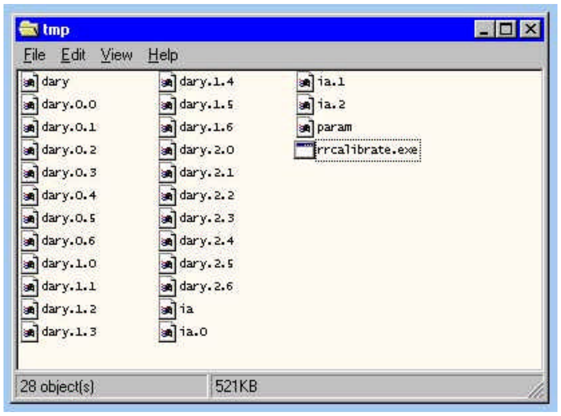
Then **rrcalibrate** starts to compute.



```
open infile
dary
...
open outfile
ia
...
open paramfile
param
...
processing channel 0
          setDataforChannel
          iaCurveFromH
processing channel 1
          setDataforChannel
          iaCurveFromH
processing channel 2
          setDataforChannel
          iaCurveFromH

_
```

rrcalibrate generates one response curve list file which is named as <outfile> (ia in the following figure) and three response curve files which are named as outfile>.c (ia.c in the following figure) are. Here, c is number of channel (R,G and B, respectively). A response curve list file contains file names of response curve files and each response curve file contains the actual data of estimated response curve function and exposure ratios of each channel.



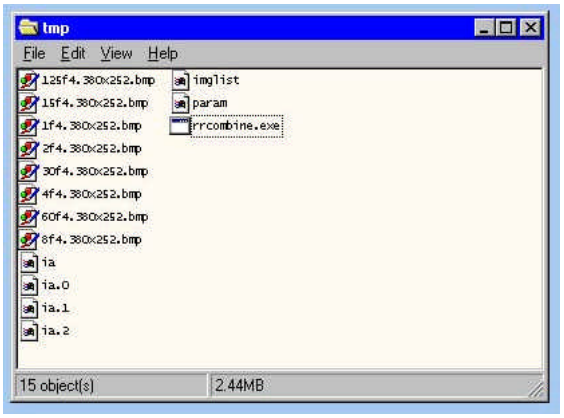
dary
dary.0.0
dary.0.1
dary.0.2
dary.0.3
dary.0.4
dary.0.5
dary.0.6
dary.1.0
dary.1.1
dary.1.2
dary.1.3
dary.1.4
dary.1.5
dary.1.6
dary.2.0
dary.2.1
dary.2.2
dary.2.3
dary.2.4
dary.2.5
dary.2.6
ia
ia.0
ia.1
ia.2
param
rrcalibrate.exe

28 object(s)
521KB

3.4 rrcombine

§ Preparation

rrcombine requires image files, an image list file, response curve files, a response curve list file and a parameter file. The image list file is same one which you prepared as an input to **rrselect**. The response curve files and the response curve list file are outputs by **rrcalibrate**. The parameter file differs to previous parameter files which were for **rrselect** or **rrcalibrate**.



Parameter file defines setting of **rrcombine**. The format of parameter file is as followings.

File name of this file.

```
id    param RRCombineParamData
creator     nobody
numSource   0
saturationLevel  0.99
noiseLevel 0.01
checkIVarianceCriterion     0.0
trimRatio   1.
bestMLuminance   0.7
selectNeutralColorCriterion 0.02
```

rrcombine has 6 kinds of parameters.

The order of parameter in a parameter file must not be changed.

Parameters of **rrcalibrate** are explained in the followings.

double **saturationLevel**

Upper clipping level of pixel value. This must be under 1 if you want to reject saturated pixels. default value is 0.990.

double **noiseLevel**

Lower clipping level of pixel value. This must be over 0 if you want to reject black-out and noisy pixels. default value is 0.010. If you have significant noise at dark area., try to increase this parameter (try before changing **checkIVarianceCriterion** parameter).

double **checkIVarianceCriterion**

Threshold value of variance of ideal pixel value. When **rrcombine** computes combining images with weighting summation, one of two kinds of weighting scheme is chosen. Variance of ideal pixel values is used as index for the choosing weighting scheme. Increasing this parameter is effective when noise at dark area is significant. Decreasing this parameter softens noise at edge area but noise at dark area may become significant. Default value is 0.0. This means that only one scheme is usually used.

double **trimRatio**

Ratio of trimming size to the original image size. Use this parameter when images have non-image area at frame. **rrcombine** ignores pixel values in outside area trimmed by this parameter to compute a correct radiance image. If **trimRatio** is 0.7, the size of trimmed rectangle is 70% of the original one. Default value is 1, that means no trimming.

double **bestMLuminance**

Luminace value to be selected for color balancing. **rrcombine** adjust balance of R,G and B channel of the output radiance image so that the radiance image has same color to original images. For the adjustment, colors which have luminance near **bestMLuminance** are selected as reference colors from original images. Increasing this parameter forces to select colors from brighter (more exposed) images. This parameter must be from 0 to 1. Default value is 0.7.

double **selectNeutralColorCriterion**

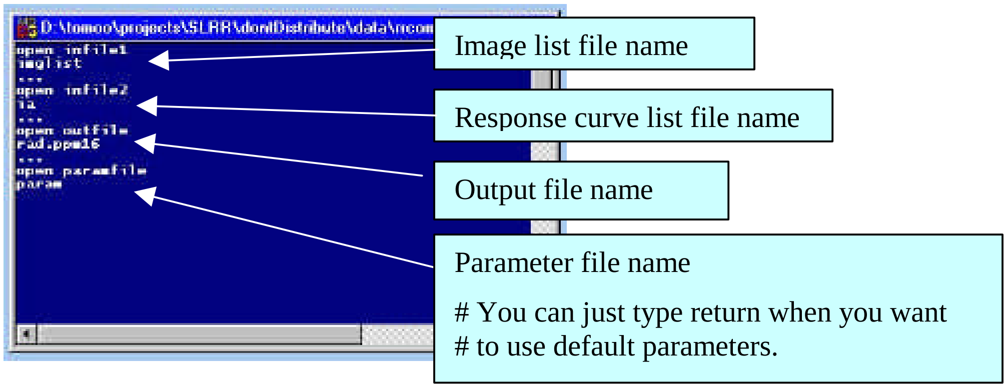
Selectivity criterion of neutral colors for color balancing. Color balancing by **rrcombine** can be adjusted so that it prefers to balance neutral colors. **selectNeutralColorCriterion** specifies the range of neutral color. When the following equations are satisfied, the color (R,G,B) is a neutral color.

| R - mean(R,G,B) | / mean(R,G,B) < **selectNeutralColorCriterion**,
| G - mean(R,G,B) | / mean(R,G,B) < **selectNeutralColorCriterion**,
| B - mean(R,G,B) | / mean(R,G,B) < **selectNeutralColorCriterion**

Decreasing **selectNeutralColorCriterion** lets the color balancing prefer more neutral color.Default value is 0.02.

§ Execution

Execute **rrcombine.exe**. And type asked file names.



```
open infile1
imglist
...
open infile2
ia
...
open outfile
rad.ppm16
...
open paramfile
param
```

Image list file name

Response curve list file name

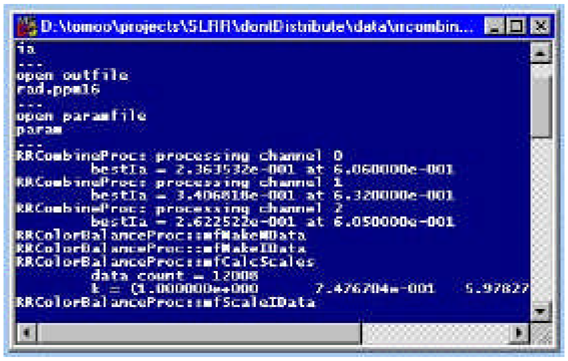
Output file name

Parameter file name

# You can just type return when you want
# to use default parameters.

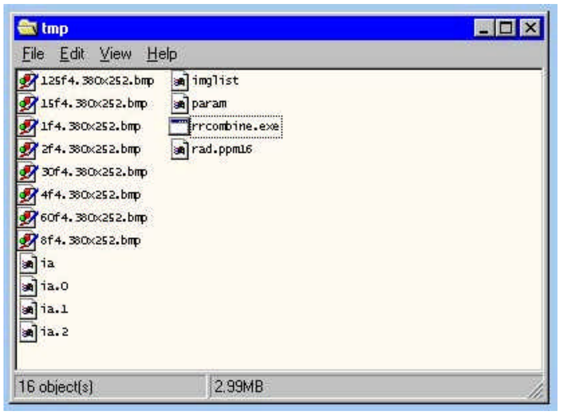
rrcombine decides the image file format of output image with the extension of the file name which you specified. Currently, **rrcombine** recognizes "**.bmp**", "**.ppm**" and "**.ppm16**" as available extension. Since a combined result which **rrcombine** outputs has greater dynamic range than the dynamic ranges of original images, **PPM16** format which is a 16bit image format is recommended.

Then **rrcombine** starts to compute.



```
ia
---
open outfile
rad.ppm16
---
open paramfile
param
---
RRCombineProc: processing channel 0
          bestIa = 2.363532e-001 at 6.060000e-001
RRCombineProc: processing channel 1
          bestIa = 3.406818e-001 at 6.320000e-001
RRCombineProc: processing channel 2
          bestIa = 2.622522e-001 at 6.050000e-001
RRColorBalanceProc::mfMakeMData
RRColorBalanceProc::mfMakeIData
RRColorBalanceProc::mfCalcScales
          data count = 12008
          k = (1.000000e+000        7.476704e-001    5.97827
RRColorBalanceProc::mfScaleIData
```

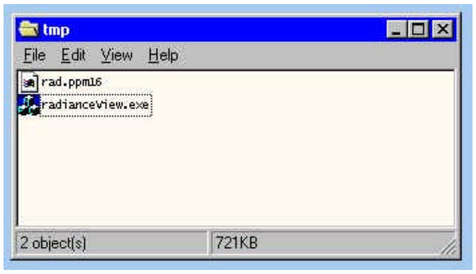
Finally, **rrcombine** outputs one high dynamic range radiance image file (rad.ppm16 in the following figure).



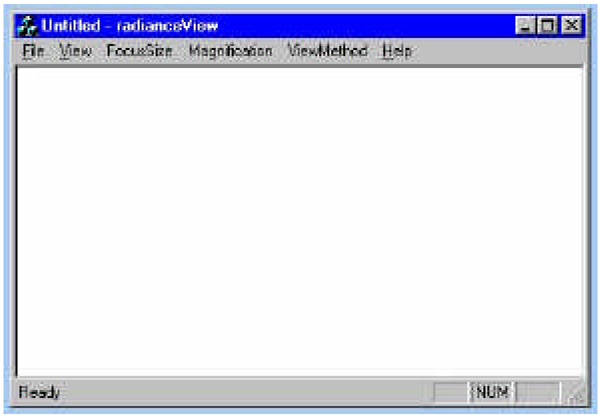
3.5 radianceView

§ Execution

You can display a PPM16 image with **radianceView**. Double-click the icon to execute **radianceView**.

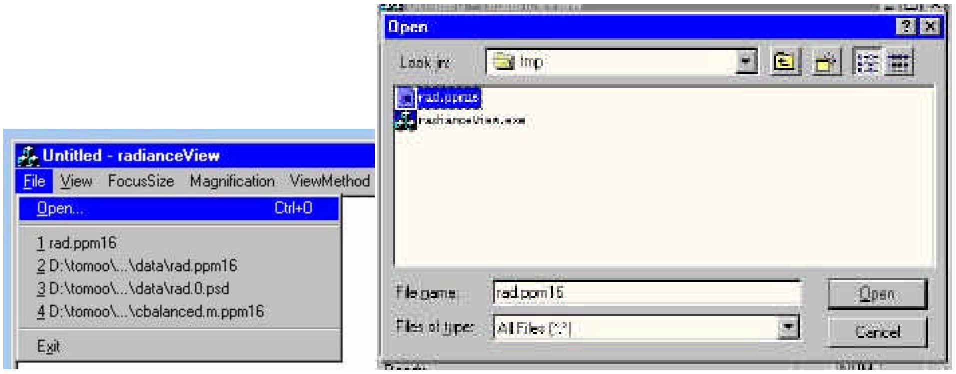


The following figure shows starting view of **radianceView**.



§ Loading a radiance image

Use menu **"File / Open..."** to open an image. Then select an image file which you want display.

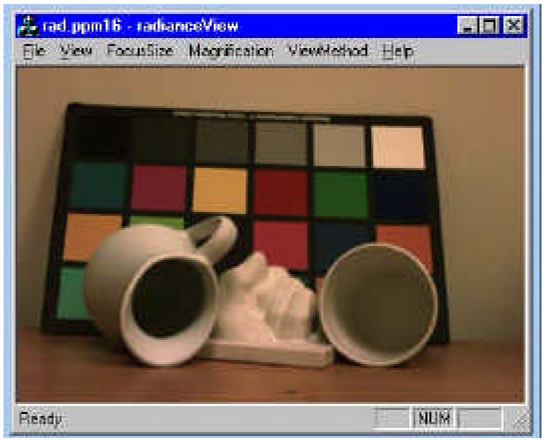

Untitled - radianceView
File View FocusSize Magnification ViewMethod
Open... Ctrl+O
1 rad.ppm16
2 D:\tomoo\...\data\rad.ppm16
3 D:\tomoo\...\data\rad.0.psd
4 D:\tomoo\...\cbalanced.m.ppm16
Exit
Open
rad.ppm16
radianceView.exe
Open
Cancel

Look in: tmp

File name: rad.ppm16

Files of type: All Files (*.*)

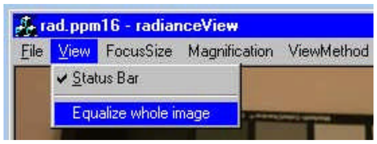
Then selected image is displayed.



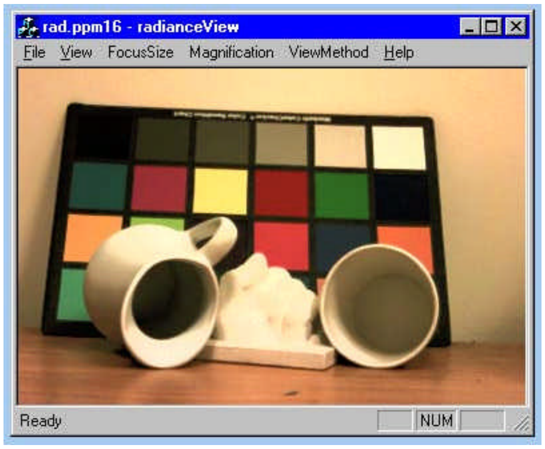
§ Enhancing details

Radiance images usually dark to see directly. **radianceView** has detail enhancing functions to view details.

To view the whole image with histogram equalization, use menu **"View / Equalize whole image"** .

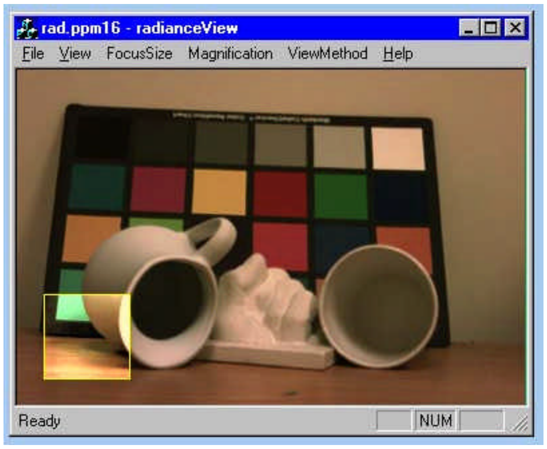


The following figure is the result of histogram equalization. If you want return to normal view, re-use same menu.

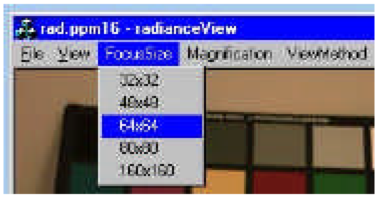


§ Enhancing detail of a local area

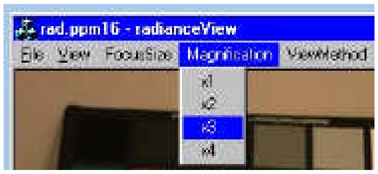
To view details local areas, press **left button** on where you want view. Then you will see a rectangle window which display enhanced detail of the local area.



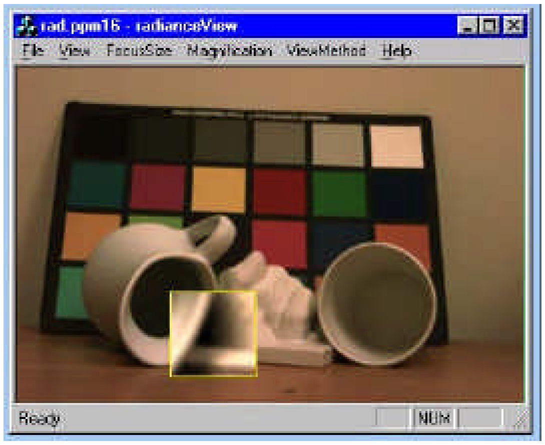
To change the size of the window, use menu **"FocusSize"**. Select one size which you want see with.



To view with magnification, use menu **"Magnification"**. Select one magnification scaling which you want see with.



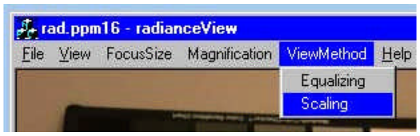
The following figure is an example of magnified view.



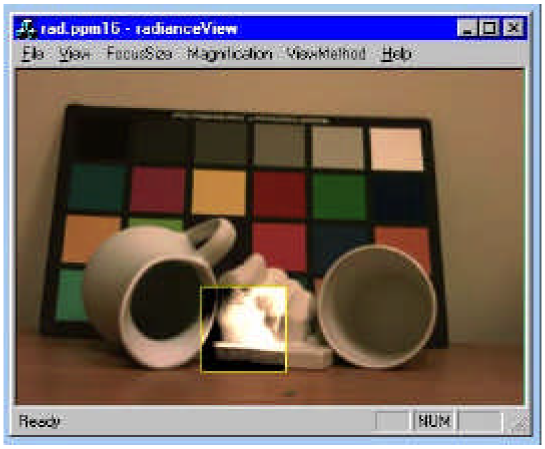
To change detail-enhancing method, use menu **"ViewMethod"**. Select **"Equalizing"** or **"Scaling"** which you want.

- **Equalizing** : Histogram equalizing -- Strong enhancing, better to recognize the details.
- **Scaling** : Monotonic scaling -- Linear, more natural than histogram equalizing.

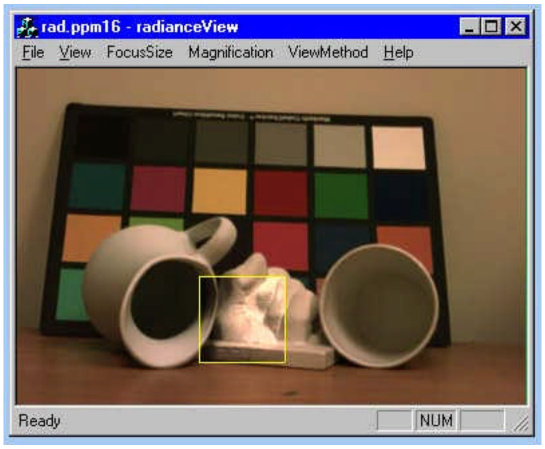
The default method is **Equalizing**.



The following figure is an example of **Scaling**.



The following figure is an example of **Equalizing**.



4. Problems

4.1 Contact

Please feel free to direct any questions and bug reports to tomoo@cs.columbia.edu or nayar@cs.columbia.edu . And we appreciate very much it if you let us know how your experiments with RASCAL are going.

4.2 Currently known problems

§ General

Very slow when programs treats large images.

§ imageAlignment

Ripples at interpolating -- Sinc function with 4x4 neighbor pixels are currently used. Some better interpolating method is needed.

Sise of search area is fixed to 32x32. This might be cause of misalignment.

§ rrselect

None.

§ rrcalibrate

Sometimes computing does not converge. Most cases of those can be improved by changing some parameters of **rrselect** .

§ rrcombine

None.

§ RadianceView

None.