Yiqing Cai

Team G: The ExcalibR

Teammates:
Huan-Yang Chang
Man-Ning Chen
Siddharth Raina
Sambuddha Sarka

ILR04
Nov. 11, 2016

Individual Progress

Overview

For this stage of project, I was primarily responsible for the photometric calibration (calculating the camera response function). In the last progress review, I was facing serious memory problems with the method for calculating camera response function. Then I moved on to look for other methods and now I am able to deal with large number of full resolution images. Besides, in order to make the code efficient, I translate the Matlab Code to C++ code. We used the emergent vision technology HR-12000 cameras and AEROTECH linear actuator (act as the robot arm) in this stage of work.

Implementation

The camera response is determined as a function per pixel, and we want to figure out the true pixel color irrespective of the exposure time.

Because the algorithm we implemented (referred as method 1) in the last stage would take a lot of memories, so we could only do the calculation based on no more than 20 images and need to down-sample the images when executing the Matlab code. This would end up with rank deficiency problems when solving SVD, and the code took really long time to give out the solution. Besides, the data we got is obviously not smooth enough due to the rank deficiency and memory problems.

In order to solve the problem, I implemented another method (referred as method 2) introduced by the paper " Engel J, Usenko V, Cremers D. A photometrically calibrated benchmark for monocular visual odometry[J]. arXiv preprint arXiv:1607.02555, 2016. " The main idea is to minimize the following Maximum-Likelihood energy formulation:

$$E(U, B') = \sum_i \sum_{\mathbf{x} \in \Omega} \left(U\left(I_i(\mathbf{x})\right) - t_i B'(\mathbf{x}) \right)^2.$$

where U is the inverse response function, I(x) is the pixel value we get, t is the exposure time and B'(x) is the irradiance of a specific pixel.
As a result, we can solve the inverse response function U and the irradiance map B' through specific number of iterations in order to make the error function stable. That leads to the following equations :

$$U(k)^* = \underset{U(k)}{\operatorname{argmin}} E(U, B') = \frac{\sum_{\Omega_k} t_i B'(\mathbf{x})}{|\Omega_k|}$$

$$B'(\mathbf{x})^* = \underset{B'(\mathbf{x})}{\operatorname{argmin}} E(U, B') = \frac{\sum_i t_i U\big(I_i(\mathbf{x})\big)}{\sum_i t_i^2}$$

To initialize the irradiance map B', I took the mean of all the images taken at different exposure time t, I could then took the initialized B' to calculate U, then took the U to calculate B' again. After several iterations, the Maximum-Likelihood energy formulation will become stable at a specific small number, now the U is what we want as the inverse response function.

Firstly I wrote the algorithm on Matlab, but if I took large number of images (for example, 1000 images), the algorithm would take several days to run. So I translated the code from Matlab to C++ to make it more efficient. After some testing and optimization, finally we can get the results within 20 minutes if we use 1000 images and take 20 iterations.

The comparison of method 1 and method 2 is shown in Figure 1.

By theory, more images should give out better results and the inverse response function curve should be smoother. So I tested the algorithms on 50 images and 1000 images, what I got is shown in Figure 2, the curve is definitely smoother.

Another way to solve the problem occurred in method 1 is to randomly select pixels in each images and do the optimized calculation in C++ also. We can not test on full resolution images, but it can also give out the inverse response function. In the following stages we will use the inverse response function got by two methods to calibrate images and see which one is better in practice. The curse is also given in Figure 3.

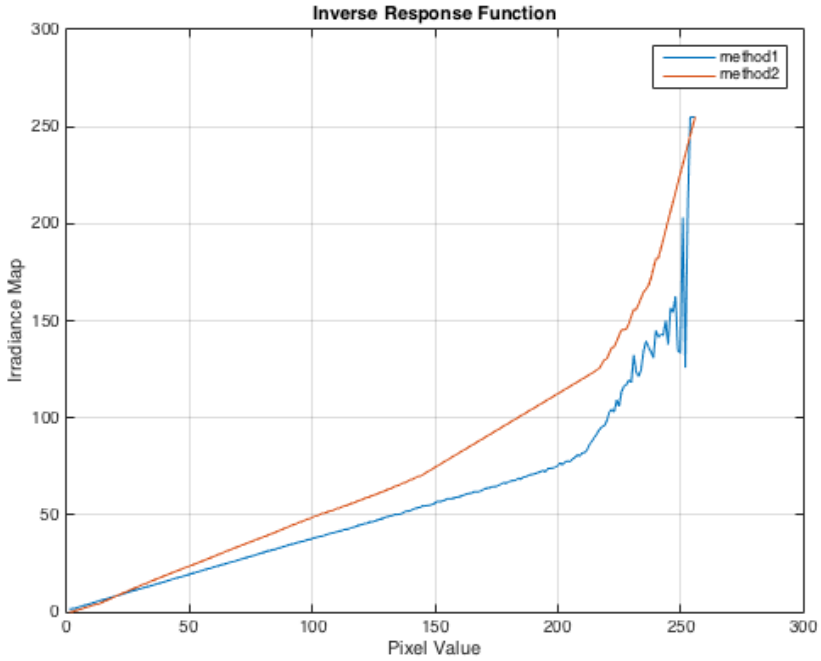

Figure 1. Comparison of 2 methods on 1000 images

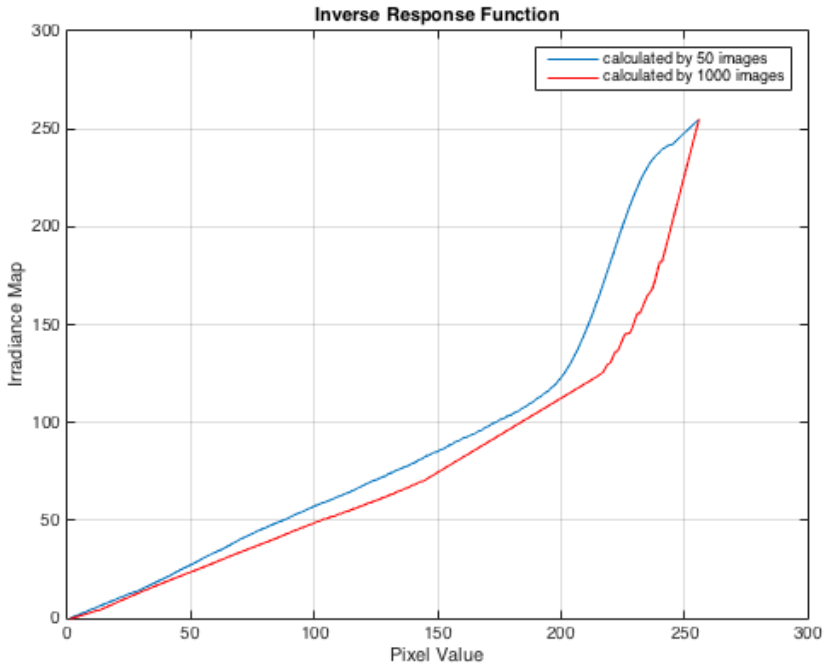

Figure 2. Comparison of method 2 on 50 images and 1000 images

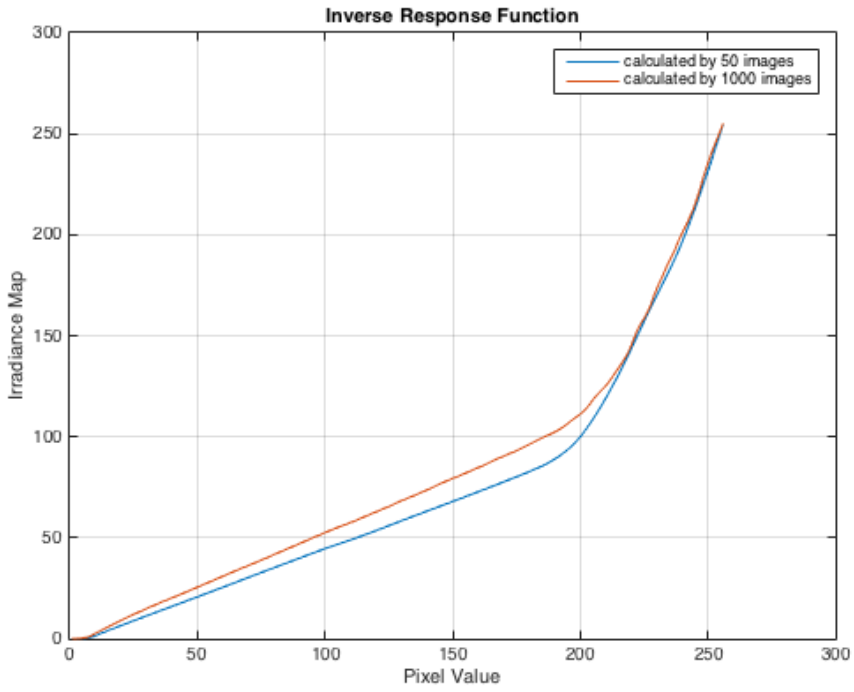

Figure 3. The optimized method 1 on 50 images and 1000 images

Challenges

The main challenges I faced during this task were:

1. The photometric calibration process is based on large amount of calculation, so it is not suitable to use Matlab and we need to use C++ and OpenCV, which is also good for building the pipeline and integration of all different parts of the whole projects.

2. When gathering image data for testing, it is hard to obtain 100% static scenes of different exposure time because of the wobbling of camera and change of environment light. That will ends up with rubbish pixel values of the image data and might lead to the calculation mistakes. In order to solve the problem, we might need to use a light box and a stable tripod to fix the camera.

Teamwork

Work undertaken by each team member is as follows (see Table 1):

Member	Tasks
Huan-Yang Chang	Implement geometric code and confirm the way of evaluation
Man-Ning Chen	Optimize FPN code and conduct photometric calibration experiments
Yiqing Cai	Test and Implement different methods and generate the inverse CRF curve
Sambuddha Sardar	Robot arm trajectory generation and velocity profiling
Siddharth Raina	Implement FPN correction and photometric calibration for geometric input

Table 1. Team co-work

Our team worked with great coordination during execution of the second stage of this project. We communicated during the entire task and solved problems together. Sam was working on the Robot (AEROTECH - 3 Prismatic Joint) Robot arm trajectory generation and velocity profiling, Peter was working on implementing the geometric calibration code and confirming the evaluation method, Mandy was working on translating the FPN code to C++ and also part of the photometric calibration experiments, I was working on testing and implementing different methods of photometric calibration and collecting static images with large dynamic range, Sid is working on implementing the FPN correction and photometric calibration on images for geometric input. We faced many difficulties but we worked them out eventually as a team.

Future Plans

From now on, my task will be focused on building the calibration pipeline and connecting photometric calibration part with FPN correction and geometric calibration. We need to discuss and agree on a fixed format to store the calibration parameters and a good way to integrate all the functions and codes in C++. Also, we should integrate the multi-camera image capturing procedure with the whole calibration pipeline, I will be working on the light box to collect better data and generating CRF for multi-cameras.