Master Semester II project Report: 
Real-Time Rectification Hardware in Dual Camera Disparity Estimation System

Student: 
Halima Najibi

Supervisors: 
Yusuf Leblebici - Professor
Abdulkadir Akin - PHD

July 3, 2014
# Contents

1 Motivation .................................................. 2

2 State of the art .............................................. 3
   2.1 Disparity estimation ........................................... 3
   2.2 Camera calibration and rectification ......................... 4
      2.2.1 Camera calibration ....................................... 4
      2.2.2 Stereo image rectification .............................. 5

3 Hardware implementation of rectification algorithm ............ 8
   3.1 Hardware description ....................................... 9
      3.1.1 Index rectification module ............................ 9
      3.1.2 Rectification control module .......................... 12
      3.1.3 Rectification hardware module ......................... 14
      3.1.4 Top level rectification module ......................... 16
   3.2 Hardware size and precision parameters ..................... 18

4 Modelsim Simulation results .................................. 21
   4.1 Simulation results with test data .......................... 21
   4.2 Simulation results for RESET mode ......................... 23

5 REFERENCES .................................................. 25
1 Motivation

Real-time image rectification is one of the most important pre-processing steps in depth estimation. Achieving a perfect physical allignement of the two cameras is an almost impossible task, that is why a rectification hardware is an essential part of a real-time depth estimation system.

A compressed look-up table based real-time rectification solution was proposed in a previous work of A. Akin [1]. His approach requires negligible amount of hardware, does not need any external memory and can be implemented in low-cost FPGA. The mappings between original image pixel coordinates and rectified image pixel coordinates are pre-computed and stored as compressed look-up tables in ROMs. The hardware decompresses the LUTs and retrieves the color of each pixel of the rectified image from the corresponding pixel from the original image.

The main disadvantage of this realisation is that whenever the camera setup changes, the camera rectification parameters and the pixel mapping between original and rectified pixel coordinates need to be computed again and the whole system needs to be resynthesised.

The new rectification hardware proposed in this work allows to compute the pixel mapping between original and rectified images in real-time and assign the right pixel colors to left and right rectified images from the original ones. The designed graphical user interface (GUI) and software allow to calculate the camera parameters automatically, the user only needs to place a checkerboard in front of the camera. The estimated camera parameters are then written in the FPGA registers and used for image rectification.
2 State of the art

2.1 Disparity estimation

Disparity estimation is an algorithmic step that is applied in a variety of applications such as autonomous navigation, robot and driving systems, 3D geographic information systems, object detection and tracking, medical imaging, computer games, 3D television, stereoscopic video compression, and disparity-based rendering.

The stereo matching process compares the pixels in the left and right images and provides the disparity value corresponding to each pixel. If the cameras could be aligned perfectly parallel and if the lenses were without distortion, the matching pixels would be located in the same row of the right and left images. However, providing the perfect set-up is almost impossible. Camera calibration and real-time image rectification become then a necessary step to guarantee the proper functionality of depth estimation.
2.2 Camera calibration and rectification

2.2.1 Camera calibration

Camera calibration is a necessary step before proceeding with rectification and depth estimation [2]. The task of this step is to determine the parameters of the transformation between an object in 3D space and the 2D image observed by the camera, or in other words estimate the camera projection matrix from object-space to 2D screen space. These parameters can be estimated using visual informations (images of known objects, for example a checkerboard).

There are different techniques for camera calibration: 1D, 2D, 3D, self-calibration. In our case, we use Jean-Yves Bouguet’s algorithm for calibration [3, 4] which is a 2D plane based camera calibration. This technique requires to observe a planar pattern (a checkerboard) at few different orientations and estimate the camera parameters using maximum likelihood methods. His method allows in theory to estimate very well the parameters of the camera only with two images from left and right camera, and cosidering a few different points in each image.

The camera is modeled using the usual pinhole camera model. The parameters that are estimated with camera calibration using the pinhole model may be divided into two categories: intrinsic and extrinsic parameters. Intrinsic parameters are directly related to the physical characteristics of the camera, whereas extrinsic parameters are related to the camera position with regard to the object.

![Figure 1: Pinhole camera model](image)

- Extrinsic parameters:
  - Orientation (or rotation) $R$, which is one of the most important parameters for rectification. Although it is a $3 \times 3$ matrix, it has 3 unknown parameters which are the angles of rotation around the 3 axes $\hat{e}_x$, $\hat{e}_y$ and $\hat{e}_z$
  - Position (or translation) of the camera $t$, which is a $3 \times 1$ vector
• Intrinsic parameters:
  - The coordinates of the center point \((u_0, v_0)\)
  - The scaling factors \(\alpha\) and \(\beta\), which model the scaling from 3D world to pixel coordinates. These factors correspond in fact to the focal length of the camera \(f_c\) expressed in units of horizontal and vertical pixel. The two components are usually very similar.
  - The skew factor \(\gamma = \alpha \cot(\theta)\), with \(\theta\) being the angle between the two image axes \(\mathbf{u}\) and \(\mathbf{v}\)

The final projection matrix is equal to:

\[
P = A[Rt]
\]

with:

\[
R = R_x(\theta_x) \times R_y(\theta_y) \times R_z(\theta_z)
\]

\[
A = \begin{bmatrix}
\alpha & \gamma & u_0 \\
0 & \beta & v_0 \\
0 & 0 & 1
\end{bmatrix}
\]

In addition to camera projection intrinsic and extrinsic parameters, camera calibration allows to estimate the distortions of which the images are subject, which are the radial and tangential distortions. They are modeled as a \(5 \times 1\) vector \(k\).

In total, there are 16 parameters to estimate using camera calibration, which will be then useful to proceed to the image rectification and calculate the mapping between the pixel coordinates of the rectified and original left and right images. There are 5 intrinsic camera parameters, 6 extrinsic parameters and finally 5 distortion parameters.

2.2.2 Stereo image rectification

Given the stereo rotation and translation matrices between two cameras, stereo rectification aims to mathematically (rather than physically) align them into one viewing plane so that pixel rows between the two cameras are exactly aligned with each other. It also attempts to limit the amount of change that a new projection produces for each of the two images from the two cameras, in order to reduce the distortion and maximize the common viewing area.

Image rectification is achieved by applying two new rotations to the left and right images. These two rotations are provided by the following rotation matrices:

\[
R_{left} = R_{epi\mathbf{r}_l} \\
R_{right} = R_{epi\mathbf{r}_r}
\]
Where:

- $R_{left}$ is the rotation to be applied to the left image to get the left rectified image
- $R_{right}$ is the rotation to be applied to the right image to get the right rectified image
- $R_{epi}$ is the epipolar rotation matrix that row-aligns the images obtained by transforming the original images by $r_l$ and $r_r$
- $r_l$ is the rotation matrix required to make left image coplanar with the right image rotated by matrix $r_r$
- $r_r$ is the rotation matrix required to make right image coplanar with the left image rotated by matrix $r_l$

Matrices $r_l$ and $r_r$ are obtained by applying half a rotation clockwise and counterclockwise, respectively, so that $R = r_l^T r_r$ where $R$ is the rotation between the two cameras.

Matrix $R_{epi}$ can be obtained using the translation matrix $T = [T_x T_y T_z]^T$ and is given as $R_{epi} = [r_1 r_2 r_3]^T$. Matrices $r_1$, $r_2$, $r_3$ are given by the following equations:

$$r_1 = \frac{T}{||T||}$$

$$r_2 = \frac{[-T_y T_z 0]}{\sqrt{T_y^2 + T_z^2}}$$

$$r_3 = r_1 \times r_2$$

The goal of the $R_{epi}$ matrix is to push the epipoles towards infinity, so that epipolar lines become horizontal in the rectified images.
Figure 2: Stereo coordinate system used in algorithm

Figure 3: real situation between two cameras and mathematical alignment we want to achieve
3 Hardware implementation of rectification algorithm

The proposed rectification hardware is a verilog implementation of J.Y Bouguet’s rectification algorithm [3, 4]. However, a few changes were made to the algorithm that allow to reduce the computational complexity in the hardware. The changes that were done include pre-computing certain parameters in the software calibration process, and directly writing them in the registers for the hardware to use them.

The hardware is fully parametrized and can be adaptable to the image size. All the parameters such as the sizes of the signals and their fixed point precision can be found in the Verilog header file "package.vh". The current chosen parameters have been tested and provide very good results. Indeed, some parameters such as floating point precision for some signals can be increased to guarantee better results, especially in cases where the camera setup is relatively bad. The only problem with increasing the precision is that it slows the hardware and make it heavier.

In the following, each module/sub-module of the rectification hardware will be described, as well as their respective inputs and outputs.
3.1 Hardware description

3.1.1 Index rectification module

The rectification of the row and column indexes of the original image coming from the camera is done in the "rect_index.v" module. This module is a pipelined implementation of the MATLAB algorithm that was previously used to do the rectification. However, some modifications were done to make it adaptable to hardware. More specifically, the inputs of this module are not exactly the same as those in the previous MATLAB codes. Some values that do not depend on the row and column number are precomputed to reduce the complexity of the hardware.

The inputs and outputs of this module are listed in the following:

• Inputs:
  
  - clock signal "clk"
  - reset signal "rst"
  - row and column index numbers "row_di" and "col_di"
  - center coordinates "cx_di" and "cy_di" (x corresponds to the column and y to the row)
  - the P_new signal that contains the 6 first elements of the new projection matrix to apply to the image "P_new_di", where the lower bits correspond the the lower row index, ie:
    
    \[ P_{\text{new}}_{\text{di}} = [P_{\text{new}}(3,2) \ P_{\text{new}}(2,2) \ldots \ P_{\text{new}}(2,1) \ P_{\text{new}}(1,1)] \]
  
  - the precomputed signal "rays_di" such as rays_di = [rays(3) rays(2) rays(1)] where:
    
    \[ \begin{align*}
    \text{rays}(3) &= P_{\text{new}}(9) - P_{\text{new}}(6) - P_{\text{new}}(3) \\
    \text{rays}(2) &= P_{\text{new}}(8) - P_{\text{new}}(5) - P_{\text{new}}(2) \\
    \text{rays}(1) &= P_{\text{new}}(7) - P_{\text{new}}(4) - P_{\text{new}}(1)
    \end{align*} \]
  
  - the focal length in \( \vec{e}_x \) and \( \vec{e}_y \) directions, f_di = [f(2) f(1)]
  
  - the distortion parameters:
    
    \[ k_{di} = [k(5) \ k(4) \ k(3) \ k(2) \ k(1)] \]

• Outputs:
  
  - the four nearest integers to the computed rectified indexes
    
    "Ypos_inv_int_do"
    "Ypos_inv_int_plus_do"
    "Xpos_inv_int_do"
    "Xpos_inv_int_plus_do"
  
  - the fixed point fractional rectified indexes "Ypos_inv_frac_do" and "Xpos_inv_frac_do"
This module includes two other ones "division_pipelined.v" and "apply_distortion.v". The first one is a necessary step in the calculations, and the second one is a separate module that applies the distortions to the new calculated indexes. Obviously, the "apply distortion" module has the most algorithmic complexity, the "pipelined division" adds latency to the hardware. The total latency of the block is 55 clock cycles.

Figure 4: Dataflow diagram of apply_distortion module
Figure 5: Dataflow diagram of rect_index module

Please note that to avoid confusion, same namings of signals are used in the verilog codes and the MATLAB codes.
3.1.2 Rectification control module

The control of the inputs of the rectification module "rect_index" and of the target row and column output of the rectification hardware is done via the control unit "rectification_ctrl".

When the pixels from the camera arrive row by row, the rectification_ctrl waits until a certain number of rows have already been received and stored in the disposable 128 original images BRAMS before starting the rectification. This threshold is set to 64. This threshold indicates that if the difference of row indexes between two identical pixels in left and right image is more than 64 rows (due to an important misalignment or rotation between the two cameras), the rectification hardware will assign wrong data to that pixel because the BRAM block that corresponds to the rectified row will be overwritten.

The rectification_ctrl module also takes into account the delay of the rect_index module, so that at each clock cycle the source pixel coordinates that are calculated are outputted synchronously with the corresponding target pixel coordinates of the rectified image (the source pixel coordinates are the coordinates of the pixel in the original image from which the pixel color data will be taken).

Figure 6: chart flow of rectification_ctrl to follow in order to properly assign the row and column to send to the rect_index module
The inputs and outputs of this module are listed in the following:

- **Inputs:**
  - clock signal "clk"
  - reset signal "rst"
  - row and column index numbers coming from the camera "row_di" and "col_di"

- **Outputs:**
  - the row and column pixel coordinates to send to the rectification hardware rect_index "row_r_i_do" and "col_r_i_do"
  - the delayed row and column pixel coordinates of the rectified image "row_r_do" and "col_r_do", so that the output and the input of the rect_index module are synchronized
### 3.1.3 Rectification hardware module

The rectification for each image is done by the final rectification hardware (there will be as many of this module as the number of cameras, in our case two cameras). The inputs and outputs of this module are described in the following:

- **Inputs:**
  - clock signal "clk"
  - reset signal "rst"
  - camera pixel row and column indexes "row_di" and "col_di"
  - camera pixel YCbCr data "YCbCr_di"
  - center coordinates "cx_di" and "cy_di" (x corresponds to the column and y to the row)
  - the P_new signal that contains the 6 first elements of the new projection matrix to apply to the image "P_new_di", where the lower bits correspond the the lower row index, ie:
    \[ P_{\text{new}}_{di} = [P_{\text{new}}(3,2) \ P_{\text{new}}(2,2) \ ... \ P_{\text{new}}(2,1) \ P_{\text{new}}(1,1)] \]
  - the precomputed signal "rays_di" such as \( \text{rays}_di = [\text{rays}(3) \ \text{rays}(2) \ \text{rays}(1)] \) where:
    \[ \text{rays}(3) = \text{P}_{\text{new}}(9) - \text{P}_{\text{new}}(6) - \text{P}_{\text{new}}(3) \]
    \[ \text{rays}(2) = \text{P}_{\text{new}}(8) - \text{P}_{\text{new}}(5) - \text{P}_{\text{new}}(2) \]
    \[ \text{rays}(1) = \text{P}_{\text{new}}(7) - \text{P}_{\text{new}}(4) - \text{P}_{\text{new}}(1) \]
  - the focal length in \( \vec{e}_x \) and \( \vec{e}_y \) directions, \( f_{di} = [f(2) \ f(1)] \)
  - the distortion parameters:
    \[ k_{di} = [k(5) \ k(4) \ k(3) \ k(2) \ k(1)] \]

- **Outputs:**
  - rectified image pixel row and column indexes "row_r" and "col_r"
  - rectified image pixel YCbCr data "dina_r"
The rectification hardware receives the data coming from the camera as well as the rectification parameters that are calculated in the software and stored in the registers of the FPGA. The hardware writes directly the incoming camera data in the 128 disposable BRAMS (each BRAM represent one row of the original image and we can store 128 rows at a time). The rectification control module takes care of giving the right row and column number as input to the rect_index module, as well as synchronizing its inputs with its outputs. After the rectified source coordinates are calculated, the data from the four neighboring pixels are read from the BRAMS (true dual port BRAMS are used to have the possibility to read two addresses in the same time, because we need to read two consecutive columns at a time). After the pixel values are read, we use the 3 first bits of the fractional part of the rectified indexes to interpolate between the four values and assign the interpolated colour to the rectified image pixel.
3.1.4 Top level rectification module

The top level rectification module receives the pixel data from the right and left cameras as well as the rectification configuration signals. The "configuration_writer" module receives the configuration parameters and writes the camera parameter values one by one if the write operation is enabled. The two left and right rectification hardware modules receive the data from the cameras and the cameras parameters from the "configuration_writer" and proceed the rectification. An additional control module sends the "start of lines" and "end of frame" signals according to the rectification outputs.

![Figure 8: Position (in blue) of the top level rectification hardware in the DE system](image)
Figure 9: Block diagram of top_level_rectification module
3.2 Hardware size and precision parameters

As mentioned previously in this report, the rectification hardware is fully parametrized in terms of signal sizes and precisions. A very high accuracy can be achieved if the sizes of the signals are big enough and the precision is high, but in relatively good situation (good camera alignment) the results are very good even if the precision and the sizes of each signal are reduced, in order to achieve high speed and reduce hardware resource usage.

In order to make a decision on the size/precision parameters, Several MATLAB simulation were made to determine the limit for each signal at which the results are no longer acceptable. For that purpose, a new MATLAB code was written. This new MATLAB code works similarly to the hardware. The inputs have the same binary precision as the hardware, and all the intermediate values that are calculated in the algorithm are rounded to have the same precision as the corresponding signals in the hardware.

The integer part size of the inputs is set according to their maximum possible values (for example the new projection matrix can never have an element equal to 100). For the intermediate signals, the maximum value that are reached in the MATLAB simulations were calculated, giving information about the maximum number of bits that is needed. 4 more bits are added then to guarantee functionality in different cases, possibly worse.

One of the most important parameters according to MATLAB tests is the precision of the new projection matrix. In case of 8 bits precision, the rectification completely fails, in case of 10 to 16 bits, the rectification has poor performance. The precision of the centre point coordinates, on the other hand, has very little effect since it is just added in the last step of the rectification. It is also important that the intermediate results have also enough precision not to accumulate big errors after each step.

In the following are the MATLAB simulation results using test data for different cases:

![Figure 10: Left and right rectified image - full precision](image)
3 Hardware implementation of rectification algorithm

Figure 11: Left and right rectified image - P_new 26bits precision

Figure 12: Left and right rectified image - P_new 16bits precision

Figure 13: Left and right rectified image - P_new 12bits precision
The following sizes and precisions are used for the final implementation:

<table>
<thead>
<tr>
<th>signal</th>
<th># bits</th>
<th># bits of precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>P_new</td>
<td>28</td>
<td>26</td>
</tr>
<tr>
<td>rays</td>
<td>28</td>
<td>26</td>
</tr>
<tr>
<td>cx/cy</td>
<td>16</td>
<td>6</td>
</tr>
<tr>
<td>k</td>
<td>12</td>
<td>10</td>
</tr>
<tr>
<td>f</td>
<td>16</td>
<td>2</td>
</tr>
<tr>
<td>intermediate signals</td>
<td>-</td>
<td>16</td>
</tr>
</tbody>
</table>
4 Modelsim Simulation results

4.1 Simulation results with test data

In this section, the MODELSIM simulation results are presented of the rectification hardware included in the disparity estimation system. A proper testbench is used that verifies the functionality of the hardware using real image data and simulating the input of two real cameras. The simulation results of the rectification hardware (rectified row and column numbers) are the same as those of the modified (Hardware-like) MATLAB code, and are accurate compared to the original matlab results.

![Screenshot of MODELSIM simulation waveforms](image1)

Figure 14: Screenshot of MODELSIM simulation waveforms

![Screenshot of MODELSIM simulation waveforms - ZOOMED](image2)

Figure 15: Screenshot of MODELSIM simulation waveforms - ZOOMED
Figure 16: Left and right rectified image - HARDWARE results

Figure 17: Left and right rectified image - MATLAB results
4.2 Simulation results for RESET mode

In this section, the reset mode is tested. RESET mode is very important as it will provide the first images (that should ideally be unchanged) which the software will use to estimate the camera parameters. The simulation results of the modified (Hardware-like) MATLAB code are the same as those of the hardware.

In the RESET mode, the following parameters are used to keep the original images unchanged:

$$KK = \begin{bmatrix} f(1) & 0 & c_x \\ 0 & f(1) & c_y \\ 0 & 0 & 1 \end{bmatrix}.$$

$$R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

$$P_{new} = R \cdot KK^{-1}.$$

$$f(1) = f(2) = 1024 \text{ (or any other value, preferably a power of 2)}.$$

$$K(i) = 0 \forall i \in [1, 5].$$

$$c_x = \frac{\#cols - 1}{2}.$$

$$c_y = \frac{\#rows - 1}{2}.$$
Figure 18: Left rectified image - RESET mode

Figure 19: Right rectified image - RESET mode
5 REFERENCES

References


