Computationally Efficient Implementation of Video Rectification in an FPGA for Stereo Vision Applications

Buddhika Maldeniya, Dinindu Nawarathna, Kanishka Wijayasekara, Tharindu Wijegoonasekara and Ranga Rodrigo
Department of Electronic and Telecommunication Engineering
University of Moratuwa
Katubedda, Sri Lanka
Email: {060285e, 060326h, 060545d, 060547k, ranga}@ent.mrt.ac.lk

***Abstract*—In order to obtain depth perception in computer vision, it is needed to process pairs of stereo images. This process is computationally challenging to be carried out in real-time, because it requires the search for matches between objects in both images. Such process is significantly simplified if the images are rectified. Stereo image rectification involves a matrix transformation which when done in software will not produce real-time results although it is very demanding. Therefore, the video streaming and matrix transformation are not usually implemented in the same system. Our product is a stereo camera pair which produces a rectified real time image output with a resolution of 320x240 at a frame rate of 15FPS and delivers them via a 100-Ethernet interface. We use a Spartan 3E FPGA for real-time processing within which we implement an image rectification algorithm.**

I. Introduction

In many stereo vision applications the need of a real time rectified image pair input is still a constraint. Though there are FPGA platforms available to fulfill such needs they are unable to meet all requirements of the overall design due to their complexity, lack of real time capability, lack of synchronism, low efficiency and cost. This bottleneck in stereo vision applications created opportunity for us to develop a low cost, high performance and a portable stereo camera pair which provides a synchronized real time rectified video pair as the output utilizing the inherent parallelism feature in FPGAs.

Through the initial survey we found that researchers have successfully implemented several FPGA based architecture implementations for stereo camera pairs with different rectification algorithms such as epipolar rectification [2], compact algorithm rectification [4], stereo rectification and stereo matching methods [1], [2] for specific stereo vision applications. By analyzing these various computational algorithms we came up with a simple integrated approach which simplifies the inherent complex nature of the rectification process which involves several floating point matrix multiplications [6] by replacing the process with a lookup table approach. In our design we try to achieve the maximum usage of the high speed on-chip resources such as block memory and off chip resources such as PSDRAM.

Apart from the main objective which was developing a general purpose stereo rig, we introduce novel computational techniques as a contribution to the body of knowledge such as a new method for camera synchronization, a lookup table approach which eliminates the need of floating point representations, canonical ratification approach and a simple division core which reduces the computational overhead.

II. Methodology

A. Architecture

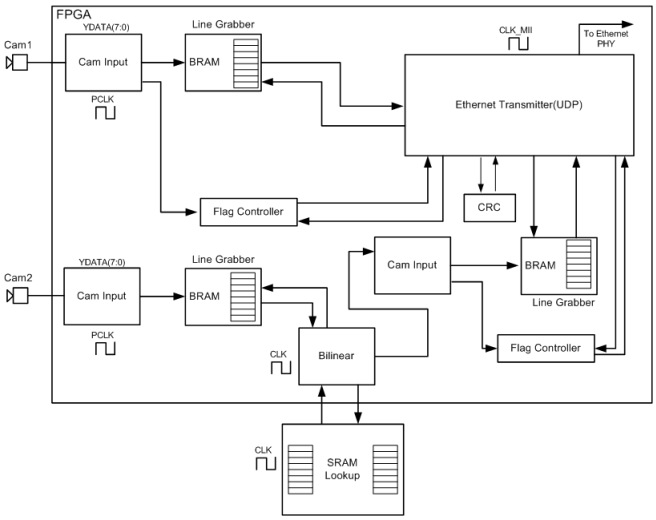

Fig. 1. Architecture of the implemented system. Rectangle blocks represent major modules on the FPGA chip and the SRAM lookup table.

The architecture of this system is shown in the Fig.1 where the input camera data is first stored in a Block RAM line by line in an alternate fashion. During the process the incoming pixel coordinates of one camera is transformed in to the new rectified coordinate plane by acquiring the new coordinates from the lookup table which is stored in the SRAM. Since the cameras are fixed with respect to each other the transformation is a one to one coordination mapping between original and rectified image plane hence can be stored in the lookup table in

the SRAM. The intensity value for the new pixel is interpolated based on the neighboring pixels of the original image using the bilinear module. Other input camera's data is by passed through the FPGA. Both these images are finally sent to the PC via Ethernet.

III. Hardware

The resource consumption within the FPGA chip is limited to 521 slices, 685 4-input LUTs, DCM clock generators and 80KB of Block RAM. Therefore the need of a high performance FPGA is eliminated making the greatest break-through for a low cost implementation. An external or on-board PSDRAM having a minimum of 2MB capacity is also needed. In our design we have used Digilent Nexys-2 prototyping board which is based on the Xilinx Spartan 3E FPGA which has got all these requirements. For the image sensors we used C3188A CMOS cameras with OV7620 chip. This provides a digital output with configurable word length 8bit or 16bit. This setting and others, such as brightness control, configuring image size, selecting progressive scan option and software master-reset are performed through an I2C bus [5]. Programming the camera chips is done by a 16F628A PIC as illustrated in Fig.2.

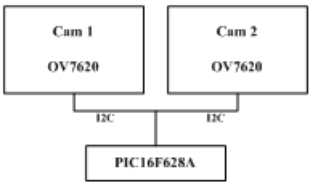

Fig. 2. Configuring camera modules using PIC16F628A through an I2C bus.

To simplify the handling and orientation of the cameras, we made a PCB (printed circuit board) to mount them on. The cameras are placed close to each other with a baseline distance of 70mm which is the average distance between the two eyes of a human, which was identified as an optimum value to obtain rectified videos for a scene with a moderate depth. Camera lenses are having a certain amount of barrel distortion. Despite this distortion, rectification can still be done since the distortion is common to both frames. Therefore additional effort was not exerted on removing the barrel distortion before the process of rectification. After rectification the video data are sent to the PC via UDP protocol through Ethernet.

We used 100BASE-T Ethernet communication for transferring video data from the FPGA to the PC. Our Ethernet module consists of a Media Access Controller (MAC) which we implemented on the FPGA for dealing with the higher level issues and a Physical Layer Interface (PHY) which uses DP83848C chip. The MAC is linked to the PHY by a 4 bit 25 MHz synchronous parallel interface known as a Media Independent Interface (MII).

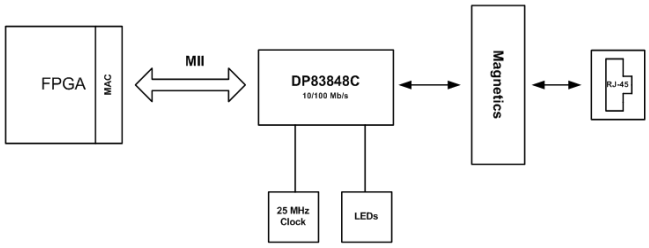

Fig. 3. Ethernet module uses 100BASE-T communication.

IV. Camera Synchronization

Stereo vision needs to find similar features in the two corresponding frames coming from the two cameras. Depending on the disparities found between the features it is possible to predict the depth to each feature in the scene. For a dynamic scene it is essential to trigger the both cameras at the same time so that they are synchronized in the time domain. Otherwise the disparities will not be solely due to the relative positions of the cameras but the objects also may have moved within the time gap between the two captures. This can lead to drastically incorrect results when predicting the depth. This issue has been addressed by accommodating the following two methods.

A. Simultaneous Configuration of OV7620 Image Sensors

The OV7620 camera chips have to be programmed prior to acquiring videos. Usage of two microcontrollers for the two camera modules create additional problems like these two camera modules not getting configured at the same time which in turns affects the synchronization of the data streams. Therefore we use one master-chip which is routed in parallel to both the camera modules so that the both camera chips get configured at the same time.

B. Compensating the Mismatches of the Clock Speeds

A clock speed mismatch is present [3] when operating the two C3188A camera modules independently using their on board crystal oscillators. Although the on board clock is supposed to operate at 27 MHz, there was a slight difference in the clock speeds. The effect of this mismatch was clearly observed by the relative motion of the Vsync clocks coming out from the two camera modules when observed using an oscilloscope which means the frame rates are slightly different.

The crystal oscillators which are placed on C3188A modules can be modeled using an LCR oscillator configuration. In series resonance mode the oscillating frequency f is given by;

$$f = 1/2\pi\sqrt{L_1 C} \tag{1}$$

The problem in hand is the two crystal oscillators having slightly different oscillation frequencies. As a solution we coupled the two crystal oscillators such that they are having a common reference point. Then the oscillator circuit can be modeled as shown in the Fig.4.

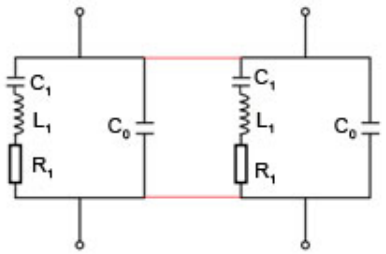

Fig. 4. Equivalent circuit for the connected oscillators.

In this arrangement the new series resonance frequency *f'* is given by;

$$f' = \frac{1}{2\pi\sqrt{(L_1/2)2C_1}} = \frac{1}{2\pi\sqrt{L_1C_1}} \tag{2}$$

It can be noted that the oscillating frequency of the modified oscillator circuit is the same as that of the original oscillator circuit. Even though there was a slight difference in the frequencies of oscillation, say f and $f+\Delta f$, still the resultant frequency will be a value between f and $f+\Delta f$ letting the system operate at a common clock frequency. Therefore this modification does not cause any change in the clock frequency but it provides the advantage of both clocks having a common reference. This way we can have the system clocks of the two camera modules perfectly synchronized with identical frequencies.

V. Image Rectification

Rectification of an image is transforming the coordinate spaces of both images until they get their epi-polar lines parallel to the baseline between the two cameras.The process of rectification has two main steps:

1) Calculation of the required transformation.
2) Application of the transformation to the images.

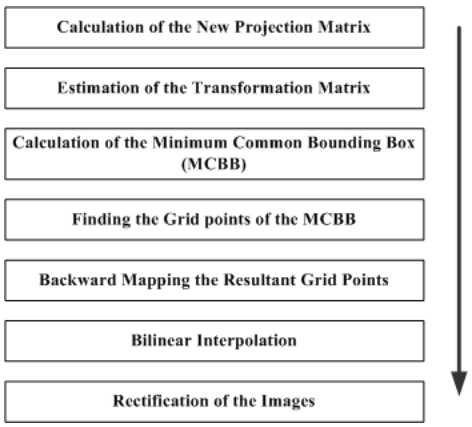

Fig. 5. Rectification Process.

During this process, calculation of calibration matrix is done using MATLAB camera calibration Tool box and referring to compact algorithm by A. Fusiello [4]. One to one relationship between original image coordinates and rectified image coordinates are derived as explained below.

A. Camera Model

Let us consider a pinhole camera model[2] with the optical center C and the retinal plane R as shown in Fig.6.

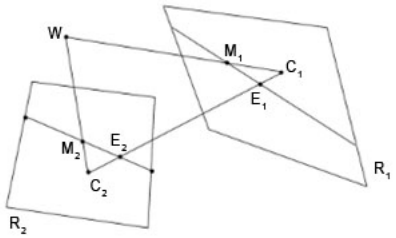

Fig. 6. Epi-polar model.

A point in the 3D space W is projected on to the image plane of the model by connecting the points C and W, naming the intersection of the lines as M in the image plane. The orthogonal line to the image plane (or the retinal plane) that is going through the optical center C is called the optical axis and its intersection with R is the principal point. The distance between C and R is the focal length f. Let us represent the world coordinate points as,

$$w = \begin{bmatrix} x & y & z \end{bmatrix}^T \tag{3}$$

While representing the image plane (pixel) coordinates in,

$$I = \begin{bmatrix} u & v \end{bmatrix}^T \tag{4}$$

Subsequently the linear transformation relation of world coordinates to the image plane coordinates can be represented by the homogenous coordinates system,

$$\hat{W} = \begin{bmatrix} x & y & z & 1 \end{bmatrix}^T \text{ and } \hat{I} = \begin{bmatrix} u & v & 1 \end{bmatrix}^T$$

The perspective projection matrix can be defined as,

$$\hat{I} \cong \hat{E}.\hat{W} \tag{5}$$

where $\cong$ means that the equation is valid up to a scalar factor. Therefore the camera is modeled by its Perspective Projection Matrix (PPM) $\hat{E}$, which is decomposed using the QR factorization method.

$$\hat{E} = A[R|t] \tag{6}$$

The matrix A depends only on the camera intrinsic parameters. It contains the parameters such as the focal length of the camera, skew factors, principal point co-ordinates etc. This matrix is obtained using the calibration process of the stereo rig.

B. Formation of the Camera Matrices

Since we initially calibrate the stereo rig before the rectification process we have the two perspective projection matrices of left and right cameras $\hat{E}_{l1}$ and $\hat{E}_{r1}$ respectively. The idea behind rectification is to define two new PPMs $\hat{E}_{l2}$ and $\hat{E}_{r2}$ obtained by rotating the old ones around their optical centers until focal planes becomes coplanar, thereby containing the baseline. This ensures that the epipoles are at infinity and the epipolar lines are parallel to each other. In order for the epipolar lines to become parallel the baseline should be parallel to both X axes in the two cameras.

In abstract, positions of the new PPMs are the same as the old cameras, whereas the new orientation (same for both cameras) differs from the old ones by suitable rotations (Intrinsic parameters are the same). Therefore, the two resulting PPMs will differ only in their optical centers, and they can be thought of as a single camera translated along the X axis of its reference system.

The two new PPMs obtained after rotation and translation will be,

$$\hat{E}_{l2} = A[R| - Rc_l] and\ \hat{E}_{r2} = A[R| - Rc_r]$$

Since the cameras do not change, the intrinsic parameters of the cameras remain the same. The points c_l and c_r can be taken through the old projection matrices. The matrix R, which gives the camera's orientation, is the same for both PPMs. It will be specified by means of its row vectors.

$$R = \begin{bmatrix} {r_1}^T \\ {r_2}^T \\ {r_3}^T \end{bmatrix} \tag{7}$$

C. Canonical Rectification

For the rectification to become successful, we have to transform the two image planes of the old camera pair to its corresponding new camera pair image planes. After the rectification the pixels (integer-coordinate positions) of the rectified image correspond, in general, to non-integer positions on the original image plane. Therefore, the gray levels of the rectified image are computed by bilinear interpolation.

Normal rectification performs domain transformations on both left and right images using their transformation matrices T_l and T_r. The transformation matrices T_l and T_r are formed in such a way that they transform the coordinate spaces of both images until they get their epi-polar lines parallel to the baseline between the two cameras. The homogeneous image coordinate transform is as follows.

$$\begin{bmatrix} X_1 \\ Y_1 \\ \lambda_1 \end{bmatrix} = T_l \begin{bmatrix} x_1 \\ y_2 \\ 1 \end{bmatrix} \qquad \begin{bmatrix} X_2 \\ Y_2 \\ \lambda_2 \end{bmatrix} = T_r \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix}$$

This process has the inherited disadvantage of having to perform transformational operations on both images. This leads to problems especially when it comes to sequential operations like memory access.

Therefore the rectification algorithm was modified such that it keeps the left image as the reference and transforms only the right image until it gets rectified with the static left image. In order to accomplish this requirement the above transformation is followed by an additional transformation by the inverse of T_l.

$$\begin{bmatrix} X_1 \\ Y_1 \\ \lambda_1 \end{bmatrix} = T_l^{-1} T_l \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} \tag{8}$$

$$\begin{bmatrix} X_2 \\ Y_2 \\ \lambda_2 \end{bmatrix} = T_l^{-1} T_r \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} \tag{9}$$

In this case the left image does not go under any transformation while $T_l^{-1}T_r$ acts as the transformation matrix for the right image. This concept is borrowed from Canonical Rectification concept. The main advantage is the left video can be directly passed through the FPGA while transformation is applied only upon the right video which saves a lot of time by reducing the number of sequential operation and reduces the resource consumption within the FPGA .

D. Lookup Table Approach

As stated under image rectification it can be noted that the transformation involves a lot of floating point matrix multiplications. This is much difficult to implement in an FPGA since a floating point representation is needed. On the other hand performing all the transformational calculations online is not essential since the transformation matrices are always fixed. Therefore a lookup table is used to store the coordinates of the rectified image planes. When storing the coordinated a 10bit word length is used for the whole part of the coordinate while the fractional part is represented as a whole number after truncating it to the nearest first decimal place. At the end of the calculation a division is performed to obtain the real result. This way it was possible to eliminate the floating point representations and online calculations.

To store the lookup table on-board PSDRAM was used. Block RAM was used as line grabber to store the incoming data from the cameras until they are put in UDP packets and transmitted to the PC.

E. Bilinear Interpolation

Intensity= $\sum(pixelintensity) \cdot (Wx \cdot Wy)/(Xd \cdot Yd)$
Wx, Wy= weighted values in X and Y direction
Xd, Yd= distance between consecutive pixels

When calculating the interpolated intensity value for the back transformed pixel coordinate as shown in Fig.7 we input X and Y coordinates with their factional parts and four pixel intensity values. These intensity values are inversely weighted with the fractional parts corresponding to the location of the neighbouring pixels. This result is then assigned to the respective X and Y coordinates which is obtained from the lookup table. We limited the fractional part of the sub coordinate level in to one decimal place to reduce computational complexity and storage requirements. Instead of using a Xilinx coregen IP core for floating point operation, we consider values as integers and did the algebra and the result is divided by 100. Since verilog does not support division, fractional multiplication (41/4096 which is almost equal to 1/100) is used for that which is simple and higher in efficiency in interpolating. Division by 4096 is done by shifting12 bits to the right.

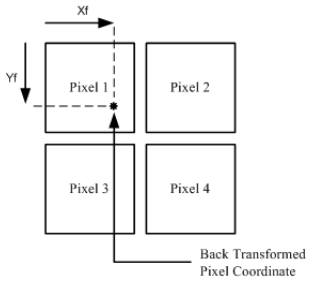

Fig. 7. Bilinear interpolation.

When the above stated modules are put together it forms the rectification module. Note that the lookup table in here is to store the coordinates obtained by canonical rectification hence the videos will be canonically rectified.

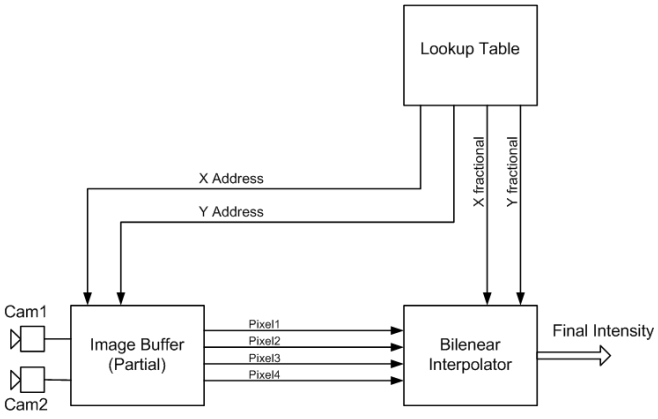

Fig. 8. Rectification module.

VI. Results and Discussion

The major milestones of synchronizing the cameras, rectification of videos, canonical rectification and the obtaining the depth map are presented.

A. Results of Synchronizing the Data Streams

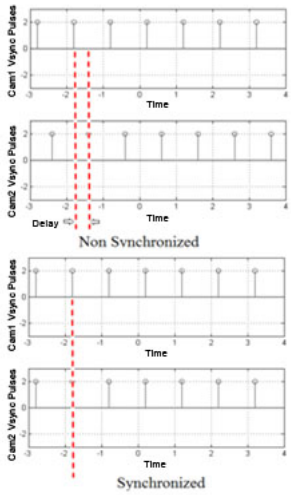

Fig. 9. Non-synchronized and synchronized vsync pulses.

Figure 9 shows the oscilloscope view of the vsync pulses coming from the two camera modules. In the first diagram a time shift between the two corresponding vsync pulses can be observed because the camera modules are not synchronized. The second diagram shows the vsync pulses obtained after synchronizing.

B. Results of Rectification

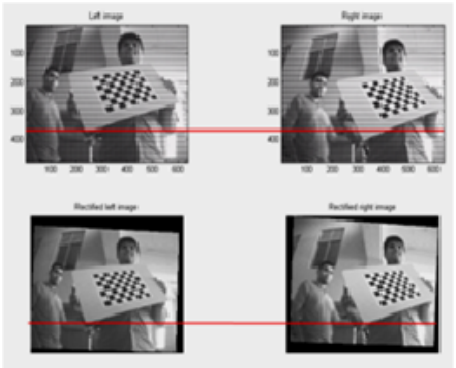

Fig. 10. Rectification done on both left and right images.

The Fig.10 illustrates that the implemented modules are capable of rectifying two videos in real-time minimizing the vertical disparity between them. By looking at the bottom most point of the checker board this can be observed.

C. Results of Canonical Rectification

Figure 11 is the result obtained by canonical rectification. In here the transformation is done on the right image while keeping the left image as the reference coordinate plane. Canonical rectification reduces the computation complexity and saves resources in the FPGA.

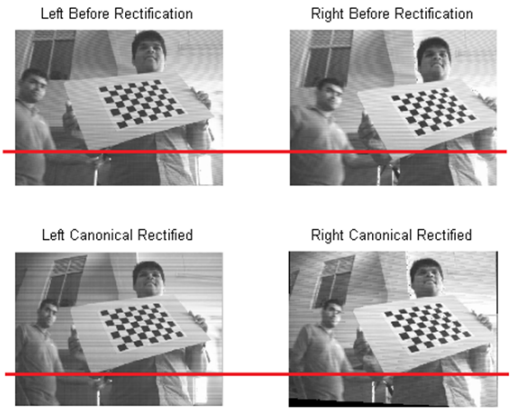

Fig. 11. Canonical rectification.Note that the left image has not been subjected to transformation.

D. Results of Creating a Depth Map

Although the main purpose of the camera pair is to rectify the two videos, a depth map was also created in software in order to demonstrate the usage of the stereo camera pair. Figure 12 and 13 show snap shots of the two rectified videos and the corresponding depth maps. The closer objects are represented in dark colours while the intensity increases with distance to the object.

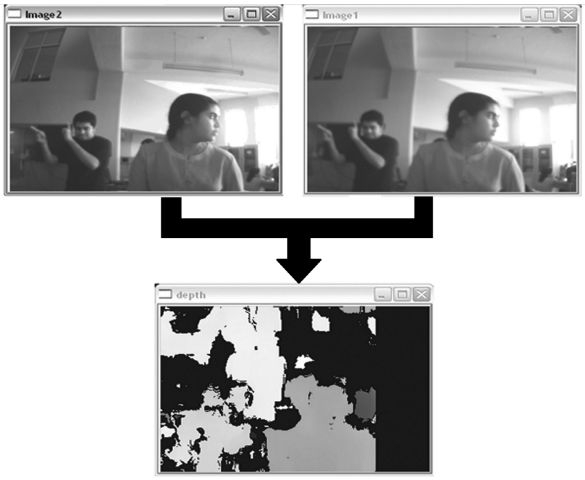

Fig. 12. Depth map.

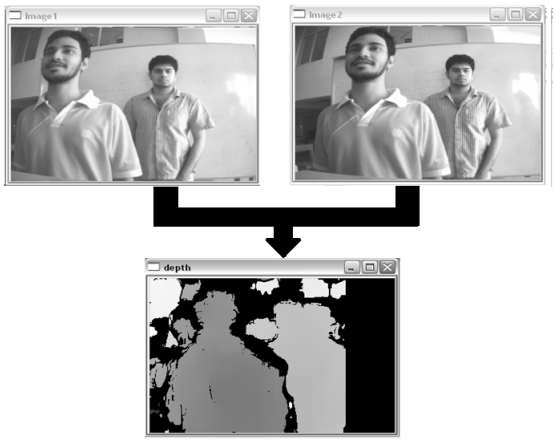

Fig. 13. Depth map presents a clear segmentation.

VII. Conclusion

Performing the rectification process within an FPGA is well suited for real-time stereo applications since the inherent parallelism in hardware can increase the performance by a large factor, hence meeting the real-time requirement. Because of the unique approach we take in rectifying the videos the resource consumption within the FPGA is kept at a minimum allowing any low performance FPGA to be used in the system. Although this implementation provides a video pair with a resolution of 320x240 at a frame rate of 15 FPS, the same system can be replicated as an enhanced design to output larger frame sizes at a higher frame rate provided that the ethernet communication can handle the data rate. Going for Gigabit Ethernet would be a possible option in future enhancements.

In summary, this paper proposes an implementation of a stereo camera pair which outputs a synchronized, rectified video pair to be used in stereo vision applications. The implementation details and the results are also presented.

References

[1] D. K. Masrani, W. J. MacLean, "A Real-Time Large Disparity Range Stereo-System using FPGAs," icvs, Fourth IEEE International Conference on Computer Vision Systems (ICVS'06), 2006.

[2] A. Fusiello. (2000, March) Epipolar rectification. [Online].Available:http://profs.sci.univr.it/˜fusiello/rectif_cvol/rectif_cvol.html.

[3] J. G. P. Rodrigues, *"Implementação de rectificação de imagens estéreo num sistema embutido baseado em FPGA"* , Junho 2009.

[4] A. Fusiello, E. Trucco, and A. Verri.*"A compact algorithm for rectification of stereo pairs"*. Machine Vision and Applications, 12(1):16-22, 2000.

[5] R. L. Baer."CMOS Imaging System and Method."U.S.Patent 7 112 774 B2, Sep. 26, 2006.

[6] R. I. Hartley, and A. Zisserman, *"Multiple View Geometry in Computer Vision"*, second edition. Cambridge University Press, 2004, pp.65-66.