White Paper

Generating Panoramic Views by Stitching Multiple Fisheye Images

This white paper discusses an innovative architecture developed by Altera and Manipal Dot Net to generate panoramic views by stitching multiple fisheye images on an FPGA. This architecture provides a complete view of a vehicle's surroundings to the driver and makes it easier to control and navigate the vehicle.

Introduction

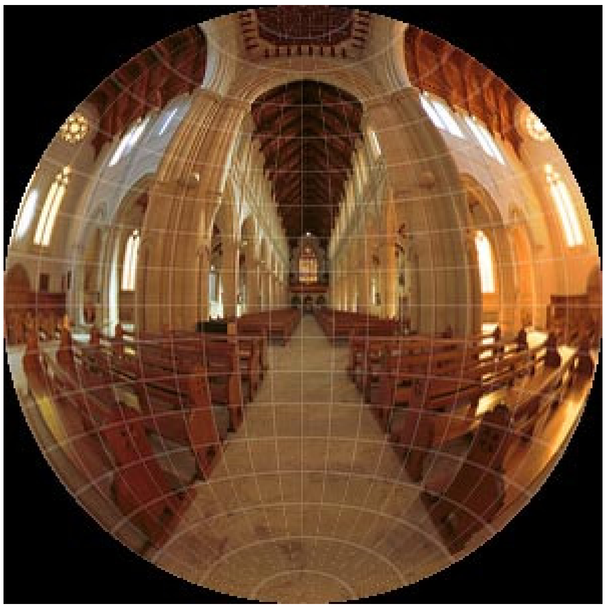
Fisheye cameras are finding an increasing number of applications in automobile imaging systems due to their ultra-wide-angle properties and cost-effectiveness. One such application renders a display of the 360° scene around the vehicle, which is achieved by stitching together multiple images that are captured by an array of fisheye cameras. This provides a complete panoramic view of the vehicle's surroundings to the driver and makes it easier to control and navigate the vehicle.

Since fisheye lenses*(1)* have very large wide-angle views, fewer cameras are needed to generate a panoramic view. However, fisheye images suffer from severe distortion due to the frontal hemispheric scene being mapped onto a flat surface. Because of this distortion, the stitching of multiple fisheye images is a non-trivial task and involves intensive computations and image processing. In general, the problem is difficult to solve, and therefore requires certain simplifying assumptions and approximations to make it feasible. The application assumes that the objects in the scene to be stitched are sufficiently far away from the camera, so that stereographic disparity is negligible.

Panoramic Views From Fisheye Images

Fisheye lenses achieve extremely wide fields of view (FOVs) by foregoing the perspective (rectilinear) mapping common to non-fisheye lenses and opting instead for certain special mappings. An earlier work*(2)* describes different fisheye mappings (e.g., linear-scaled projection mapping) and developed a flexible architecture for correcting fisheye images to perspective versions. The radial distortion caused by fisheye mappings is one in which image magnification decreases with distance from the optical axis. Also known as "barrel distortion," the apparent effect is that of an image (shown in Figure 1) that has been mapped around a sphere. As result, fisheye images do not preserve the most important feature of rectilinear images, which is the mapping of straight lines in the scene onto straight lines in the image*(3)*.

Figure 1. Fisheye Image



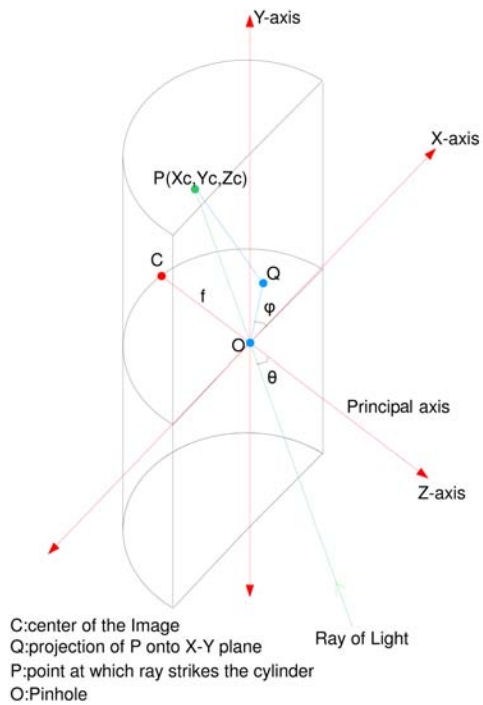
Panoramic views are characterized by very large horizontal FOVs and they typically are used to generate 360° views of scenes such as city skylines. Since fisheye images have large FOVs, they make ideal candidates for generating panoramic views. However, generating panoramic views from fisheye images is not a simple matter of correcting the images to generate corresponding perspective images.

Indeed, perspective images cannot be panoramic because the edges of such images are stretched as a result of the perspective mapping. This stretching, known as perspective distortion, becomes more severe as one gets further away from the principal axis. Generating panoramic views from perspective images therefore requires that several perspective images with small FOVs (to minimize perspective distortion) be stitched together along the horizontal axis. One way to generate panoramic views is to correct the fisheye images to perspective versions and then stitch them. However, since the perspective images themselves need to be of small FOVs, this solution cannot take advantage of the wide-angle properties of fisheye cameras.

An alternate solution is to stitch fisheye images directly to generate a panoramic view. However, this requires a special correction different from the one that corrects a fisheye image to a perspective image. This special correction is motivated by the desirable properties of panoramic views. Even though a panoramic view has a large horizontal FOV, its vertical FOV need not be large. So, in order to generate a panoramic view, the fisheye images must be corrected in such a way that they show little or no perspective distortion in the horizontal direction, while allowing perspective distortion in the vertical direction. A precise formulation of this can be developed by working with the following abstraction.

A fisheye image is formed when rays entering a pinhole camera are incident on a spherical surface of radius equal to the focal length of the camera. On the other hand, a perspective image is formed when the rays are incident on a plane whose distance from the pinhole is equal to the focal length of the camera. To correct the fisheye image so that it shows no perspective distortion in the horizontal direction, the incident surface must be circular in the horizontal direction. Since perspective distortion in the vertical direction is allowable, the incident surface can be planar in the vertical direction. An incident surface that is vertical and cylindrical with a radius equal to the focal length satisfies these requirements. The panoramic view is obtained by "unrolling" the image formed on the cylindrical surface (known as a compositing surface) and stitching it with similar images obtained from other fisheye cameras.

Figure 2. Projecting Onto a Cylindrical Surface to Produce a Panoramic Image



Assuming that the principal axis of the camera is along the *z* axis, the horizontal and vertical directions correspond to the *x* and *y* axes respectively. Let the ray entering the camera make an angle θ with the principal axis and its projection on the *xy* plane make an angle φ with the *x* axis, as shown in Figure 2. On passing through the pinhole, the ray is incident on a cylindrical surface whose radius is equal to the focal length *f* and whose axis lies along the *y* axis. Let (x_c, y_c, z_c) be the coordinates of the point at which the incident ray impinges on the cylinder. Therefore, the point (x_c, y_c, z_c) depends on (f, θ, φ) as shown in Equation (1), Equation (2), and Equation (3).

$$x_c = \frac{f \tan\theta \cos\varphi}{\sqrt{1+\tan^2\theta}} \tag{1}$$

$$y_c = \frac{f}{\sqrt{1+\tan^2\theta \cos^2\varphi}} \tag{2}$$

$$z_c = \frac{f \tan\theta \sin\varphi}{\sqrt{1+\tan^2\theta \cos^2\varphi}} \tag{3}$$

The coordinates of the panoramic (unrolled) image (x_q, y_q) corresponding to (x_c, y_c, z_c) are given by Equation (4) and Equation (5).

$$x_q = f \tan^{-1}\left(\frac{x_c}{z_c}\right) = f \tan^{-1}(\tan\theta \cos\varphi) \tag{4}$$

$$y_q = y_c = \frac{f \tan\theta \sin\varphi}{\sqrt{1+\tan^2\theta \cos^2\varphi}} \tag{5}$$

Assuming that the camera uses the linear scaled projection for its mapping function, the coordinates of the fisheye image (x_f, y_f) corresponding to (x_c, y_c, z_c) are given by Equation (6) and Equation (7).

$$x_f = f\theta \cos\varphi \tag{6}$$

$$y_f = f\theta \sin\varphi \tag{7}$$

The correction is created by eliminating θ and φ from Equation (4), Equation (5), Equation (6), and Equation (7). Equation (8) and Equation (9) provide the relationship between the coordinates of the panoramic view and the fisheye image.

$$x_f = f\left[\tan^{-1}\left\{\frac{\sqrt{\left(\frac{y_q}{f}\right)^2 + \sin^2\left(\frac{x_q}{f}\right)}}{\cos\left(\frac{x_q}{f}\right)}\right\}\right]\frac{\sin\left(\frac{x_q}{f}\right)}{\sqrt{\left(\frac{y_q}{f}\right)^2 + \sin^2\left(\frac{x_q}{f}\right)}} \tag{8}$$

$$y_f = f\left[\tan^{-1}\left\{\frac{\sqrt{\left(\frac{y_q}{f}\right)^2 + \sin^2\left(\frac{x_q}{f}\right)}}{\cos\left(\frac{x_q}{f}\right)}\right\}\right]\frac{\frac{y_q}{f}}{\sqrt{\left(\frac{y_q}{f}\right)^2 + \sin^2\left(\frac{x_q}{f}\right)}} \tag{9}$$

Stitching Fisheye Images Using a LUT

Equation (8) and Equation (9) allow the designer to map every pixel in the corrected image to a unique pixel in the input image. The corrected fisheye image shows no perspective distortion in the horizontal direction. With a circular fisheye image, these equations are enough to produce a panoramic view with a 180° FOV. However, if the fisheye image is not circular, or if panoramic views with a FOV greater than 180° are needed, then two or more such images must be stitched together.

Stitching the images requires that the images be registered to determine the region of overlap. The images must come from two or more identical cameras whose principal axes all lie in the same horizontal plane. Furthermore, the cameras must be separated by a small distance and rotated with respect to each other. Assuming that all the objects in the scene are sufficiently far away, this situation can be modeled as one in which the different images are captured by the rotation along a vertical axis of a single camera. Under these assumptions, the correction represented by Equation (8) and Equation (9) transforms the images (by mapping them onto a cylindrical surface) into horizontal translations of each other. Thus the problem of image registration is reduced to merely figuring out the horizontal shift that aligns one image with another, after correction.

However, due to errors in camera alignment, in practice it is likely that mere horizontal alignment is insufficient for exact image registration. Therefore, in addition to a horizontal shift, a vertical shift may also be necessary for exact alignment. A simple similarity-based method *(4)* determines the registration parameters (the horizontal and vertical shifts) that align the images with each other.

Once the parameters for registering the corrected fisheye images are known, Equation (8) and Equation (9) directly map every pixel of the output stitched image to one of the multiple input images by suitably translating (x_q, y_q) by the horizontal and vertical shift constants. Clearly, this mapping is independent of the content of the images, but depends only on the characteristics of the cameras (including the FOV and the resolution of the input image), the display (including resolution), and the registration parameters. Therefore, the mapping can be determined through a one-time computation at the start and stored as a look-up table (LUT).

When stitching fisheye images, the computation of the LUT proceeds in a four-step fashion:

1. The LUT for correcting a single image is computed offline (since it does not require the knowledge of the registration parameters) using Equation (8) and Equation (9), and stored in memory.

2. At the system startup, images are captured by all the cameras in the system and corrected using the stored LUT.

3. The corrected images are registered and the registration parameters computed.

4. Using the registration parameters and the stored LUT, a final LUT for directly stitching the fisheye images is generated.

Note that although Equation (8) and Equation (9) can produce real numbers for the input pixel locations, the LUT is not required to store floating-point values. This is because a 9-point interpolation method is employed that allows for efficient pixel interpolation and fixed-point representations needed for an FPGA-based implementation.

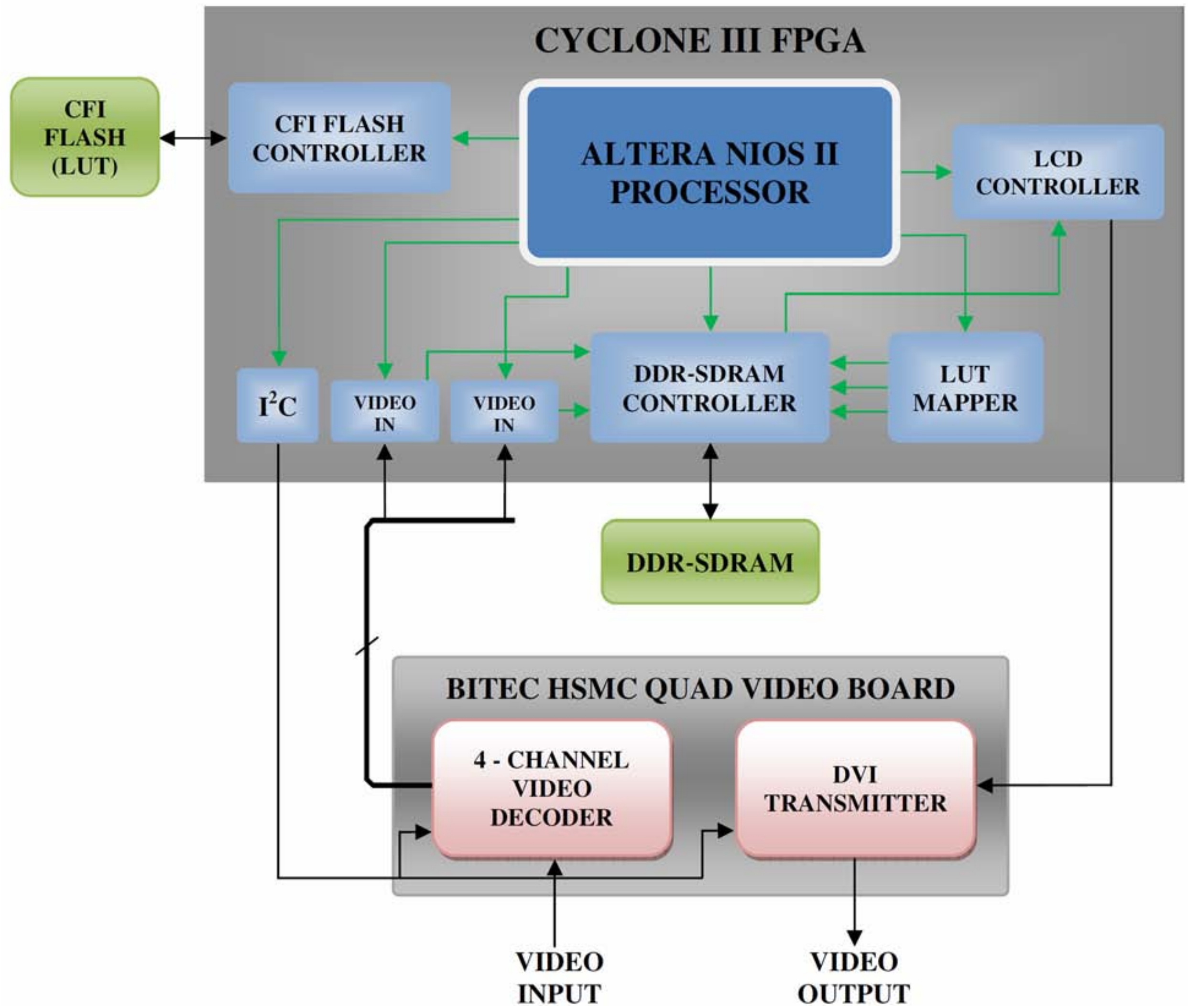
These aspects are described in further detail in Altera's white paper, *A Flexible Architecture for Fisheye Correction in Automotive Rear-View Cameras (2)*.

Design Implementation

An ideal way to implement fisheye image stitching is to use a system that includes the Altera® Cyclone® FPGA series and Altera Nios® II embedded processors. The Nios II architecture is a RISC soft-core architecture implemented entirely in the programmable logic and memory blocks of Altera FPGAs, which is capable of handling a wide range of embedded computing applications, from DSP to system control. The soft IP nature of the Nios II processor lets the system designer specify and generate a custom Nios II core, tailored for the application's specific requirements. Altera's Nios II Embedded Evaluation Kit (NEEK) is used as the development platform.

The FPGA's design architecture is based on a Nios II soft-core embedded processor, a Bitec Quad Video Input module, an I^2C configuration module, the custom LUT Mapper hardware IP, a DDR-SDRAM controller, and a LCD controller, all of which are shown in Figure 3.

Figure 3. System Block Diagram



In this implementation, the Nios II processor is used at system start to read the correction LUT stored in the system flash memory, perform image registration, compute the registration parameters, and uses them to compute a LUT that can directly stitch multiple fisheye images to display a panoramic view. The stitching LUT is made up of 32-bit words (Figure 4) that contain additional information. The interp bits, 30 and 31, indicate the type of interpolation to be performed, while the camera-select bits, 28 and 29, indicate which of up to four cameras is indexed. Bits 27 to 0 indicate which pixel in the image is to be fetched, using a number equal to the product of the row and column number of the pixel.

Figure 4. LUT Entry

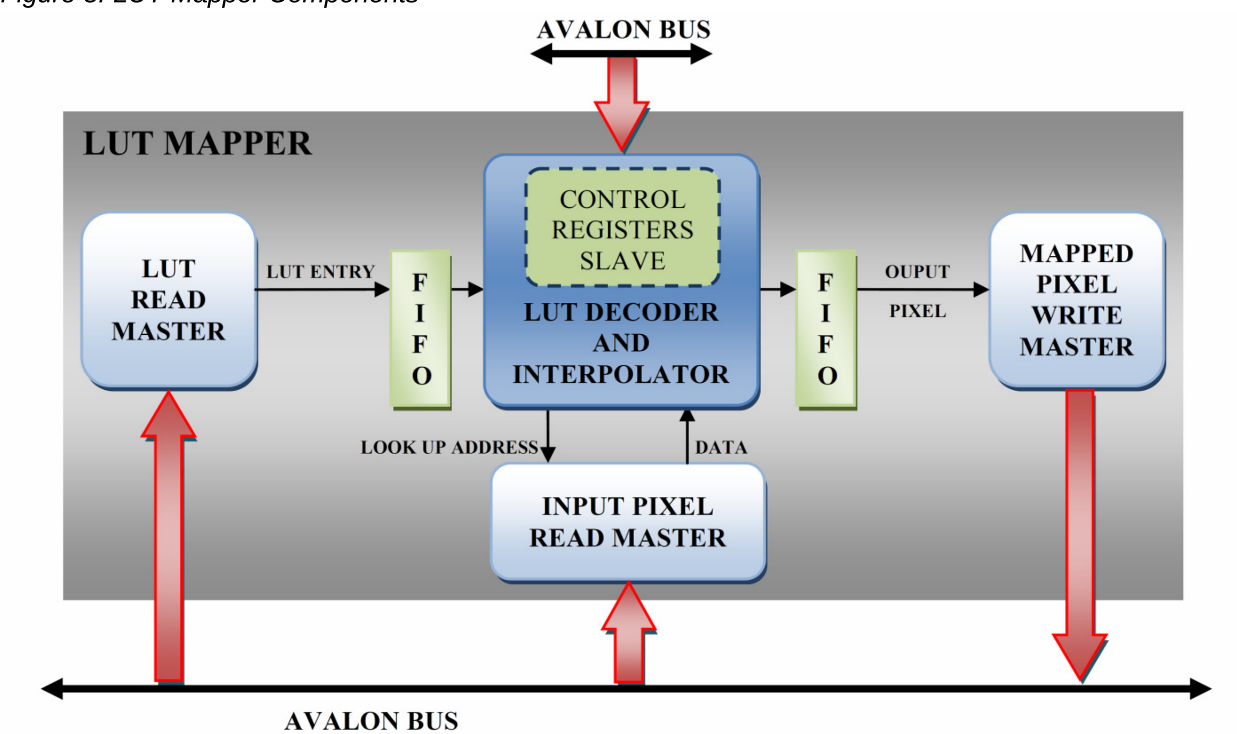
31	30	29	28	27:0
INTERP		CAMERA SELECT		LOCATION OF INPUT IMAGE PIXEL (TOP LEFT) TO BE FETCHED

Based on the entries in the stitching LUT, the output pixels are recomputed (using the 9-point interpolation method) and then displayed. The memory normally used for storing the input images is DDR-SRAM. While 9-point interpolation is not a compute-intensive task, the time needed to access the DDR-SRAM to retrieve the input pixels for interpolation is excessive. This is because the Nios II processor is not optimized for DMA-type read-write operations. In addition, some of the inherent redundancies in memory access for particular LUT entries cannot be leveraged by the Nios II processor. Therefore, a custom hardware IP block was designed to implement the LUT mapping and 9-point interpolation scheme.

The LUT Mapper

Using less than 1500 logic elements (LEs) and only three M9K memory blocks on an Altera Cyclone III FPGA, the LUT Mapper design (Figure 5) includes two Avalon® Memory-Mapped (MM) read masters, one Avalon-MM write master, and one Avalon-MM slave. For each pixel written out of the write master, one LUT entry is read through one of the read masters. The other read master reads the pixels of the input images. The slave configures the control registers, which define the operation of the LUT Mapper. The custom-designed LUT Mapper SOC component is similar to a DMA controller in that it offloads the processor from block-memory copy operations, and does not do any processing on the data it copies.

Figure 5. LUT Mapper Components



The LUT Read Master

The LUT read master complies with the Avalon-MM Read Master protocol. Its main objective is to read the LUT entries sequentially and then write them to a FIFO buffer without performing any processing. This approach buffers data from the Avalon bus to the LUT decoder by employing pipelining to maximize throughput.

The LUT Decoder and Interpolator

The LUT decoder and interpolator module reads the LUT entries from the FIFO buffer and then decodes them. For example, suppose a maximum of four camera inputs are to be stitched. The pointers to the base address of the frame buffers in memory correspond to each of the cameras and are represented as `frame_cam[camera]` where `camera` = 1, 2, 3, or 4. Let the data read out of the FIFO buffer be represented as `lut_data`. Then the LUT is stored in such a way that

```
    interp = lut_data[31:30]
    camera = lut_data[29:28]
    offset = lut_data[27:0]
Let address = (frame_cam + offset)
Let the line width of each camera image be line_width.
```

Once the LUT entry is read out of the FIFO buffer, the output pixel corresponding to this LUT entry is computed as shown in Table 1.

Table 1. Output Pixel Computation—9-point Interpolation Method

Interp	Interpolation Method	Output Pixel Value
0	No interpolation, uses “real” pixel	`[pixel_at (address)]`
1	2-pixel vertical averaging	`[pixel_at (address) + pixel_at (address + line_width)] ÷ 2`
2	2-pixel horizontal averaging	`[pixel_at (address) + pixel_at (address + 1)] ÷ 2`
3	4-pixel horizonal and vertical averaging	`[pixel_at (address) + pixel_at (address + 1) + pixel_at (address + line_width) + pixel_at (address + line_width + 1)] ÷ 4`

The pointer used within the function `pixel_at( )` is assumed to point to a 16-bit data type, since the data is in the RGB565 format (16-bit pixels). This module thus computes the output pixel value and writes it to a FIFO buffer.

The Input-Pixel Read Master

This module complies with the Avalon-MM Read Master protocol and supports pipelining. The LUT decoder and interpolator module directs this module to fetch the pixels it needs for its computations. The number of pixels required to be fetched depends on the interp bits. This module effectively reads more data than the LUT read master (for cases when interp > 0), so it is advantageous to have the arbiter’s priority of this module greater than that of the LUT read master.

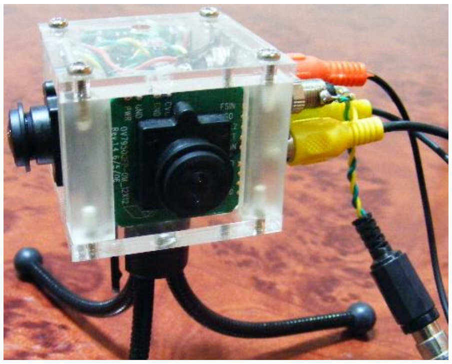
The Mapped-Pixel Write Master

This module complies with the Avalon-MM Write Master protocol. The mapped-pixel write master reads the pixel from the FIFO buffer (where it was placed by the LUT decoder and interpolator) and then writes it to the display memory in a sequential manner.

Camera Set-Up

In this example, shown in Figure 6, OmniVision OV7950 cameras are used to generate the input video streams. Each fisheye camera has a FOV of 120° and is placed with the principal axes at an angle of 90°.

Figure 6. Camera Set-Up



Results

The system described in Figure 6 captures the video feeds coming from two fisheye cameras. After a one-time registration procedure to determine the registration parameters, the resultant image sequences are stitched using the LUT Mapper, and the stitched image is displayed on a LCD screen at a resolution of 1024×768. Using the Nearest Neighbor interpolation method and without any merging algorithm, the system runs at approximately 15 frames per second with further improvement in the speed expected. This process takes the input frames shown in Figure 7 and stitches them together to create the panoramic image shown in Figure 8.

Figure 7. Input Frames From Two Cameras

Figure 8. Stitched Output Frame

Conclusions

Using FPGAs and soft-core embedded processor technology, Altera and MDN have developed a novel architecture for stitching fisheye images to generate panoramic views. This architecture is flexible, scalable, and makes efficient use of the FPGA's resources. Because the architecture's Nios II embedded processor is versatile and powerful enough to take on additional functions, this technology is ideal for applications in which 360° views are needed, such as automotive panoramic cameras.

References

1. "Fisheye lens," Wikipedia:
   http://en.wikipedia.org/wiki/Fisheye_lens
2. *A Flexible Architecture for Fisheye Correction in Automotive Rear-View Cameras*:
   www.altera.com/literature/wp/wp-01073-flexible-architecture-fisheye-correction-automotive-rear-view-cameras.pdf
3. *Transformations and Projections in Computer Graphics*, David Salomon, Springer, 2006.
4. *Image Stitching - Comparisons and New Techniques*, Technical Report, University of Auckland, Oct 1998:
   http://citr.auckland.ac.nz/techreports/1998/CITR-TR-30.pdf

Acknowledgements

- Judd Heape, Sr. Technical Marketing Manager, Industrial and Automotive Business Unit, Altera Corporation
- Dave Elliott, Sr. Marketing Manager, Industrial and Automotive Business Unit, Altera Corporation
- The MDN SOPC Team

101 Innovation Drive
San Jose, CA 95134
www.altera.com