Surround view camera system for ADAS on TI's TDAx SoCs

Texas Instruments

Vikram Appia,
Hemant Hariyani,
Shiju Sivasankaran,
Stanley Liu,
Kedar Chitnis,
Martin Mueller,
Umit Batur,
Gaurav Agarwal
Embedded Processing
Texas Instruments

Introduction

The automotive surround view camera system is an emerging automotive ADAS (Advanced Driver Assistance System) technology that assists the driver in parking the vehicle safely by allowing him/her to see a top-down view of the 360 degree surroundings of the vehicle. Such a system normally consists of four to six wide-angle (fish-eye lens) cameras mounted around the vehicle, each facing a different direction. From these camera inputs, a composite view of the surroundings of the vehicle is synthesized and shown to the driver in real-time during parking. In this paper, we present TI's 360-degree surround view camera solution and its implementation on TI's Automotive ADAS system-on-chip (SoC) processors **TDA2x**, **TDA2Eco** and **TDA3x**.

What is a surround view camera system?

Automotive surround view, also called "around view" or "surround vision monitoring system," is an emerging automotive ADAS technology that provides the driver a 360-degree view of the area surrounding the vehicle. Surround view systems normally consist of four to six fish-eye cameras mounted around the vehicle, for example, one at the front bumper, another at the rear bumper, and one under each side mirror. Figure 1 illustrates a surround view camera system and the fish-eye images captured by each camera.

Figure 1: Illustration of a surround view camera system and the fish-eye images captured by each camera.

How to generate a surround view from four fish-eye camera inputs

A basic surround view camera solution consists of two key algorithm components: geometric alignment and composite view synthesis. Geometric alignment corrects the fish-eye distortion for input video frames and converts them to a common birds-eye perspective. The synthesis algorithm generates the composite surround view after geometric correction. However, to produce a seamlessly stitched surround view output, another key algorithm "photometric alignment" is required. Photometric alignment corrects the brightness and color mismatch between adjacent views to achieve seamless stitching.

Geometric alignment

Geometric alignment, also called calibration, is an essential component of the surround view camera system. This step includes both fish-eye lens distortion correction (LDC) and perspective transformation. For fish-eye distortion correction, we use a radial distortion model and remove fish-eye from original input frames by applying the inverse transformation of the radial distortion function. After LDC, we simultaneously estimate four perspective transformation matrices, one for each camera, to transform four input LDC-corrected frames so that all input views are properly registered with the ground plane. We assume that the world is a 2D flat surface. Our algorithm is a calibration chart-based approach. The content of the chart is designed to facilitate the algorithm accurately and reliably finding and matching features. One particular chart design is shown in Figure 2.

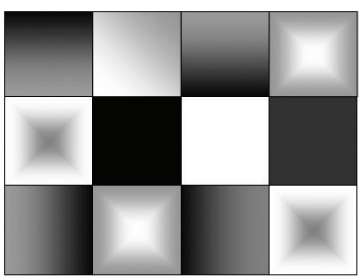

Figure 2: An example geometric calibration chart.

During calibration of the surround view cameras, four calibration charts are placed around the vehicle. Each chart should be placed in the common field of view (FOV) of two adjacent cameras, i.e., every pair of the adjacent cameras should “see” one common chart. After that, a frame from each camera is captured simultaneously.

The first step of the algorithm is to apply LDC correction to each frame. Next, we perform initial perspective transformation to each LDC-corrected frame. The parameters for the initial transformation can be obtained from camera placement specifications or estimated from the frame content itself. We used the latter approach. Next, Harris corner detection is run in the image data in the overlapping area of adjacent views to find regions of interest. We filter raw Harris corner data to locate the strongest corners and then calculate BRIEF descriptor of each corner feature to match corners from two cameras using BRIEF scores. The next step is to find the optimal perspective matrix for each frame that minimizes the distances between matched features. And finally we create a look-up-table (LUT) to encode both LDC and perspective transformation information. After the geometric LUT is obtained, it is saved to memory and used during composite view synthesis to create the final surround view output.

Photometric alignment

Due to different scene illumination, camera auto exposure (AE), and auto white balance (AWB), the color and brightness of the same object captured by different cameras can be quite different. As a result, the stitched composite image can have noticeable photometric difference between two adjacent views (i.e., camera input). The goal of photometric alignment for a surround view system is to match the overall brightness and color of different views such that the composite view appears as if it were taken by a single camera placed above the vehicle. To achieve that, we design a global color and brightness correction function for each view such that the discrepancies in the overlapping regions of adjacent views are minimized.

Assuming proper geometric alignment is already applied to the input frames, the composite surround view is shown in Figure 3 on the following page.

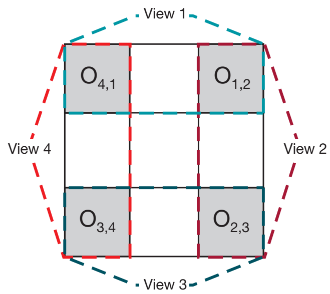

Figure 3: The views and overlapping regions in the composite surround view after geometric alignment. The composite surround view consists of data from all four input frames. $\mathbf{O}_{\{m,n\}}$ *is the overlapping region between view m and view n, and n is the neighboring view of view m in clockwise order, m=1,2,3,4,* **n = (m+1) % 4***. Image data in the overlapping regions are used to compute tone mapping functions for photometric correction.*

The composite surround view consists of data from all four input frames. The overlapping regions are portions of the frames that come from the same physical world but are captured by two adjacent cameras, i.e., $O_{\{m,n\}}$, where m=1, 2, 3, 4, and n = (m+1) % 4. $O_{\{m,n\}}$ refers to the overlapping region between view m and view n, and n is the neighboring view of view m in clockwise order. At each location in $O_{\{m,n\}}$, there are two pixels available, i.e., the image data from view m and its spatial counterpart from view n. For photometric analysis, we used the image data in overlapping regions to estimate a global photometric correction function.

For RGB input data format, we estimate a tone mapping function for each RGB color channel of each input camera by minimizing the total mean square error of the pixel value discrepancies in all the overlapping regions $O_{\{m,n\}}$, where m=1, 2, 3, 4, and n = (m+1) % 4. The pixel value discrepancy is defined as the difference between a pixel value from camera m and that of its spatial counterpart from camera n. To reduce computation, we downsample the overlapping regions by block averaging before computing the errors. The tone mapping functions for all four cameras are jointly optimized for each color channel, but independently optimized for different color channels. To achieve photometric correction, we apply the optimal tone mapping functions to the input frames. For YUV input data format, we first convert the YUV data to RGB data with standard YUV-to-RGB conversion matrix, then estimate the optimal tone mapping functions for the RGB channels, apply tone-mapping correction, and finally get the YUV output by converting the photometric-corrected data from RGB back to YUV.

Surround view synthesis

Synthesis function receives input video streams from four fish-eye cameras and creates a composite surround view. Synthesis creates the stitched output image using the mapping encoded in the geometric LUT. Figure 4 on the following page illustrates the view synthesis process. In overlapping regions of the output frame, where image data from two adjacent input frames are required, each output pixel maps to pixel locations in two input images.

In the overlapping regions, we can either blend image data from the two adjacent images or we can make a binary decision to use data from one of the two images. In this paper, we show results using the standard alpha-blending technique. Most of the color and brightness mismatch between adjacent views are removed by the photometric alignment, described in the previous section. Alpha-blending is applied to the photometric-corrected pixels to eliminate any residual seam boundaries and make the seams completely invisible. The alpha-blend weights are pre-stored in another LUT, which we refer to as the blending LUT. Output

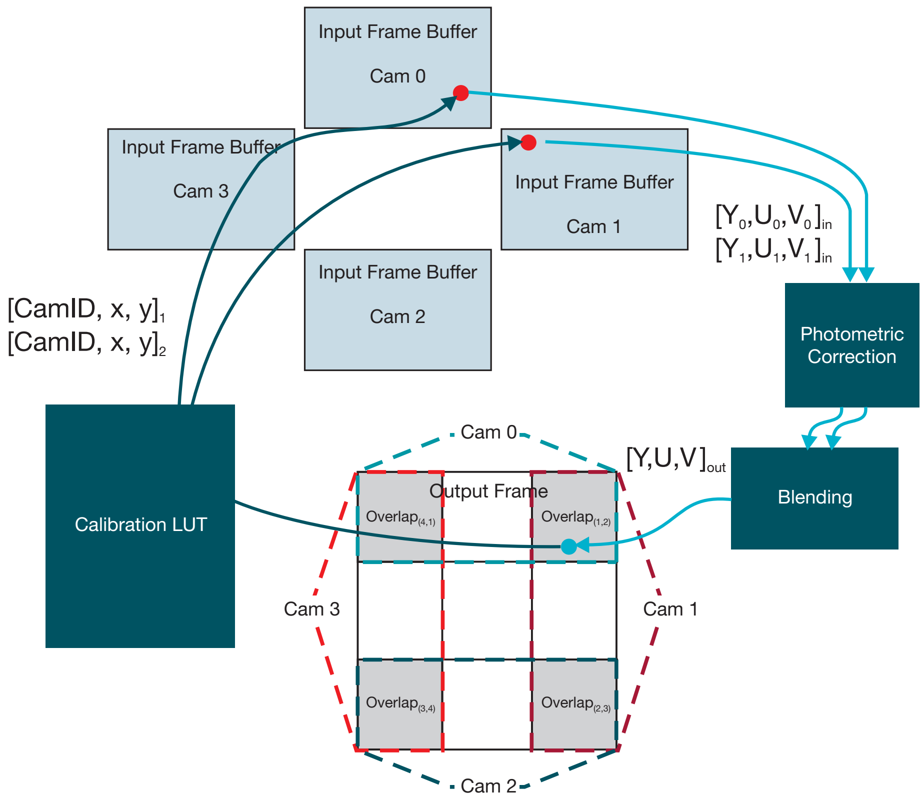

Figure 4: Illustration of the composite view synthesis process: to generate an output pixel: either two pixels (if the output is in the overlapping region) or a single pixel (if output is in non-overlapping region) are fetched from input frames through looking-up the geometric LUT. Each entry in the geometric LUT specifies the camera ID [i.e., index of the input camera(s)] and the coordinates in the input frame for generating the output pixel at the current location. After the input pixels are fetched, we apply photometric correction and blending to these pixels to generate the final output pixel.

pixels are generated by a linear-combination of the corresponding input pixel values weighted by the respective blending weights. In the non-overlapping regions, to generate an output pixel, only one input pixel is fetched based on the geometric LUT. We then apply the proper tone mapping obtained through Photometric Alignment to the input pixel to get the final output pixel value.

Our surround view solution is implemented on the DSP (C66x) core of the TDA2x, TDA2Eco and TDA3x SoCs – TI's offerings for automotive ADAS. As shown in Figure 5, the TDA2x supports high-performance 3D and 2D surround view with analytics and car back box (CarBB). TDA3 supports entry- to high-end 2D surround view with analytics while TDA2Eco supports entry- to mid-level 3D SV with CarBB.

Next we will describe the architecture and optimization of the surround view solution on the C66x DSP to meet real-time performance and memory-bandwidth requirements.

	3D	2D
Performance	TDA2x SoC	TDA2x SoC, TDA3x SoC
Low cost	TDA2Eco SoC	TDA3x SoC

Figure 5: Surround view support throughout the TDAx ADAS SoC portfolio.

Architecture of the surround view solution

The architecture of our surround view solution is designed to meet the performance and memory requirements for an embedded system. The flow diagram of the proposed solution is shown in Figure 6 below. The geometric alignment analysis (block 101) receives input fish-eye camera images and generates the geometric LUT. The output geometric LUT (block 201) depends only on the location of the cameras and does not change significantly after the initial installation. Thus, geometric alignment function (block 101) is called only once by the system framework when the system is powered up. After completion, geometric LUT (block 201) is saved in the memory.

The synthesis function (block 103) runs every frame. It takes four inputs: 1) the fish-eye frames from the four cameras, 2) the geometric LUT (block 201), 3) the photometric LUT (block 203), i.e., the tone-mapping functions, and 4) the blending LUT (block 202). The synthesis function has two outputs: 1) the composite surround view frame and 2) the statistics for photometric function (block 204). Statistics required by photometric function are the block averages of the image data in the overlapping regions of input frames. Ideally, the statistics should be collected by the photometric alignment function (block 102). This requires accessing input frames twice for each output (once for synthesis and once for photometric correction). To reduce memory bandwidth, we collect these statistics in the synthesis function (block 103) for the current frame n, and use the statistics for photometric correction in the consecutive frame (n+1). Such a design limits all pixel-level, computationally intensive operations required in each frame to the synthesis function block. It leads to a one-frame latency, but this has not been an issue for image quality in our tests. Finally, the photometric alignment analysis function (block 102) takes statistics (block 204) as the input, and generates photometric LUTs (block

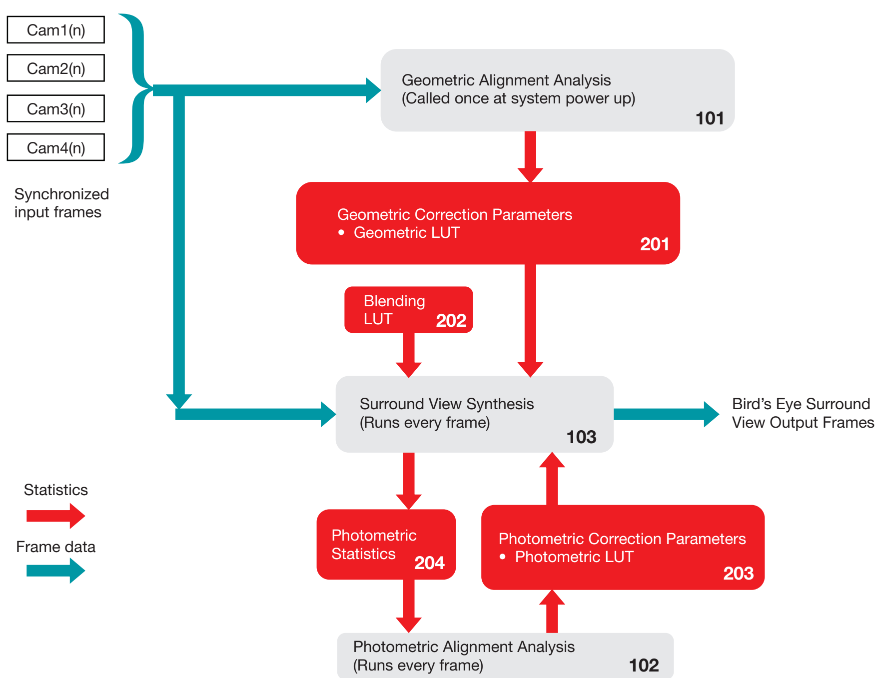

Figure 6: Flow diagram of the surround view solution.

203), i.e., the tone-mapping functions for each camera. The photometric LUTs map an input value between 0 and 255 to an output value in the same range to compensate for both color and brightness mismatch among the four input frames.

DSP optimization

Due to fish-eye warping, the access pattern for mapping output pixels to input image is not linear as illustrated by the dotted red line in Figures 7 (a) and (b). Thus, the standard linear cache access sequence for the DSP to fetch data from external memory is not optimal for creating the composite surround view frame. In real-time implementation, the DSP fetches successive pixels along the horizontal line from the input image into the internal memory to speed-up access. With pixels following the curvature in the fish-eye image, there are several cache misses. Thus the processor has to wait for the relevant input pixel to be fetched into the internal memory.

To overcome the issue of cache misses, we use a block-based direct memory access (DMA) pattern. To enable this, we divide the output image into several blocks and process each output block independently. Figure 7 shows an example of how the output blocks are mapped to one of the input fish-eye images. For a given output block, the Geometric Alignment Algorithm generates the bounding box that tightly encloses the pixels in the

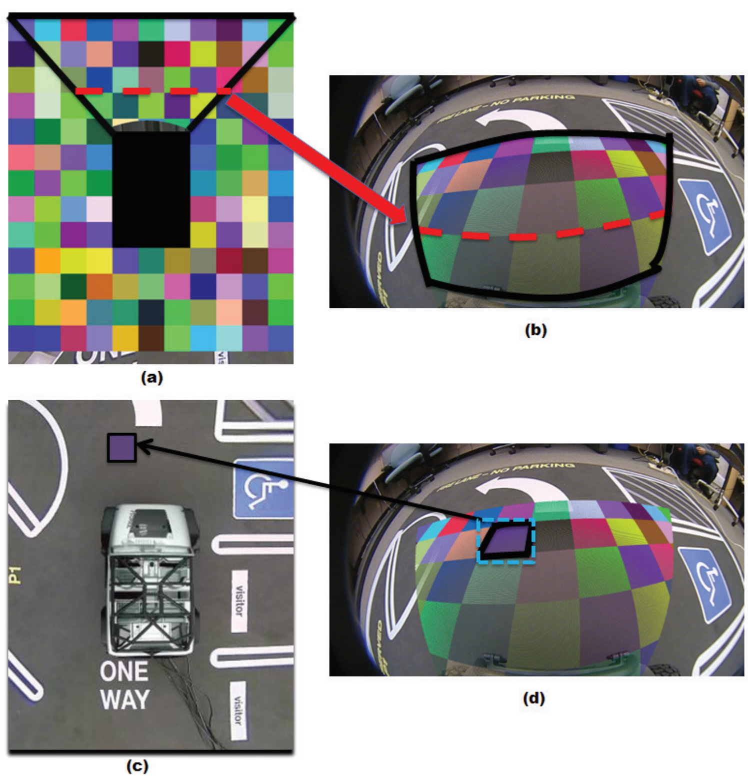

Figure 7: Illustration of DMA blocks mapping between output and input images. (a,b) show the mapping of a portion of the output surround view image to one of the input images, with the corresponding blocks overlaid. The dotted red line indicates the warped access required in the input image to fetch a horizontal line in the output frame. (c) and (d) show the mapping of one DMA block from output image to input image. The output DMA block and its corresponding input pixels are highlighted by black borders. The bounding box for the input pixels, i.e., the input DMA block is highlighted by the dotted cyan lines.

input image required for synthesizing that block and encodes this information in the geometric LUT. Figure 7 illustrates the DMA blocks mapping between output and input images. For each output pixel, the location for input pixel access is stored in the LUT as an offset from the head of the corresponding input block. Since the offset location from the head of the block can be encoded with fewer bytes compared to the offset from the head of the entire input image, we further reduce the size of the LUT, therefore reducing memory bandwidth.

To process a given output block, the photometric LUT and the corresponding block from the blending LUT are also fetched into internal memory along with the corresponding input blocks. Thus, all the data required for processing the entire output block is available in the internal memory. Furthermore, we utilize a ping-pong DMA access pattern. When one block is being processed, the data necessary for the next block is brought into the internal memory simultaneously. This ensures that we minimize the processor idle time in waiting for data to be fetched into the internal memory.

To test our surround view solution, we built a real-time surround view prototype on TDA2x, TDA2Eco and TDA3x SoCs with four cameras, as shown in Figure 8. The hardware block diagram of the system is shown in Figure 9 on the following page. Four 720p live video streams from the surround view cameras are brought into TDA2x, TDA2Eco and TDA3x SoCs through TI's FPD-Link III technology. Our surround view software runs on the C66x DSP, processes these video streams in real-time and produces an 880×1080 birds-eye view of the vehicle at 30fps. The output resolution is chosen to accommodate the display size in our demo. Other output resolutions can be supported as well.

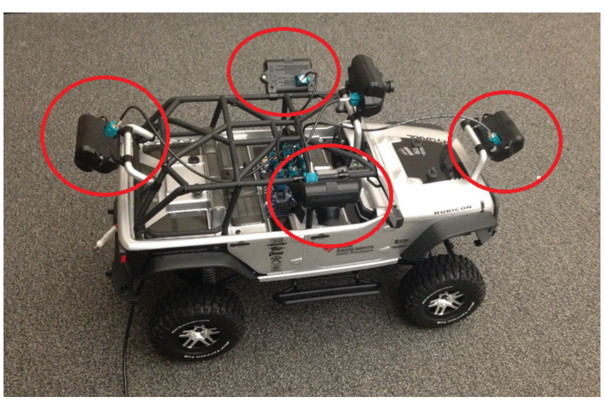

Figure 8: A toy vehicle with surround view cameras. Four fish-eye cameras with 180 degree FOV are mounted around the vehicle, each facing a different direction. The cameras are highlighted with red circles.

TDA2x, TDA2Eco SoC surround view prototype hardware block diagram

FPD-Link III

The **DS90UB913Q** and **DS90UB914Q** chipsets offer an FPD-Link III interface with a high-speed forward channel and a bidirectional control channel for data transmission over a single differential pair. The DS90UB913Q/914Q chipsets incorporate differential signaling on both the high-speed forward channel and bidirectional control channel data paths. The serializer/deserializer pair is targeted for connections between imagers and video processors in an electronic control unit (ECU). These chipsets are designed for driving video data requiring up to 12-bit pixel depth plus two synchronization signals along with bidirectional control channel bus.

3D surround view on the TDA2x and TDA2Eco SoCs

The TDA2x and TDA2Eco SoCs have an SGX544 graphics engine that can be used to implement 3D

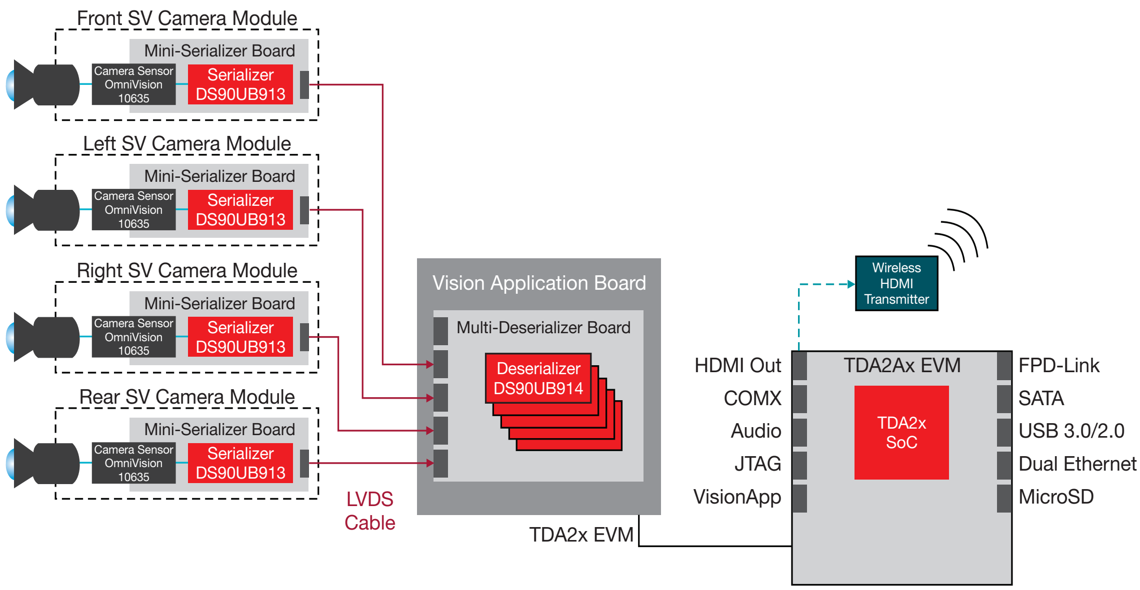

Figure 9: Hardware block diagram of the surround view camera prototype on the TDA2x and TAD2Eco SoCs.

surround view with free camera view point rendering. In this implementation, the surround view synthesis (rendering) function is mapped to SGX544 GPU by converting the warping operation into a graphics mesh. This reduces the processing load on the DSP, and also enables it to render a 3D surround view with a bowl shape from any desired camera view point that can be dynamically changed. Similar to 2D surround view, the 3D surround view has two key components: 1) system calibration and 2) 3D surround view rendering. In the following sections, we will go into detail for these tasks.

System calibration

To generate 3D surround view we need the fully calibrated system of multiple fish-eye cameras. Creating a fully calibrated system involves estimating the intrinsic parameters of each camera as well as the extrinsic pose of each camera with respect to a world co-ordinate system. We use the information provided by the lens manufacturer and the image sensor to estimate the intrinsic parameters for the cameras. In our experience, we find that this is sufficient to get good results. One can also estimate the intrinsic camera parameters for each camera module separately. Extrinsic pose of the cameras varies with each setup (vehicle on which the cameras are mounted). Thus we need to estimate the extrinsic poses of the cameras on each vehicle after the cameras are installed.

For the purpose of extrinsic calibration, we have designed a special chart with patterns of specific size and at fixed distances from each other. The chart shown below is designed to suit the size of the toy jeep we used to develop this algorithm as shown in Figure 10 below. The same pattern can

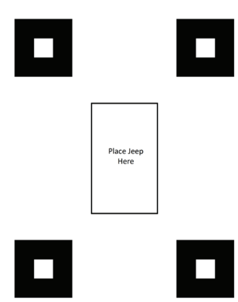

Figure 10: Chart designed for extrinsic camera calibration. Patterns are designed specifically to enable automated corner detection.

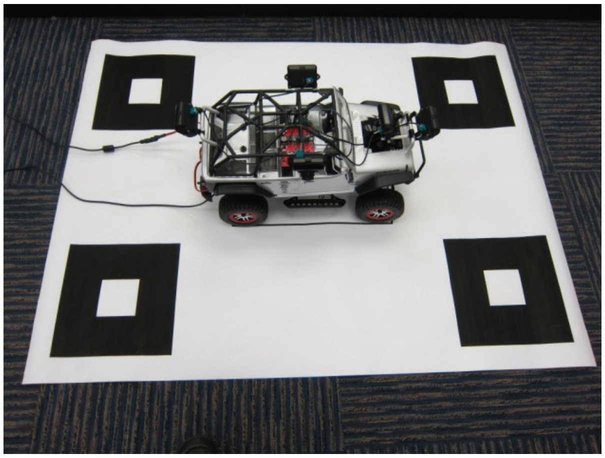

Figure 11: Illustration of placing a toy vehicle surrounded by patterns for calibration.

also be used in real-sized car calibration setup by suitably increasing the size of the patterns.

The distance in physical measurement (in meters) of the corners in this chart is known precisely beforehand. If we can identify these corners in the captured camera images and establish a correspondence with the real world distance of these corners, we can estimate the pose of the camera with respect to the patterns on the chart. To estimate the corner in the camera images we can click the corners manually or have an automatic pattern detection algorithm.

Extrinsic pose estimation

Once the corner points in the camera images are determined we have the necessary correspondences needed to calibrate the cameras. Each corner in the image has an associated 2D co-ordinate in the image plane and a real world 3D co-ordinate in the world system with a pre-defined origin.

Using these correspondences we estimate the homography from camera image plane to world co-ordinates using a direct linear transformation. Further, projecting the homography matrix on an orthogonal sub-space can provide the extrinsic pose of the camera in world co-ordinate system. Since the points on the chart are measured with physical dimensions the same physical interpretation is transitioned in the pose estimate of the cameras. Optionally, one can also use additional non-linear optimization approached to improve the estimation. In our particular example we use the Levenberg-Marquardt approach to refine our estimation of pose.

Since the pose estimation only needs a minimum of four point correspondences, RANSAC-based approaches can be used to improve accuracy. For our system calibration, we use several iterations (in the order of 100) of RANSAC to eliminate outlier correspondences caused by any imperfections in the corner detection procedure (manual or computer-generated errors) and refine our final estimate of extrinsic pose. This approach can also be readily extended to fish-eye cameras by using the fish-eye models to transform the distorted camera co-ordinate into undistorted co-ordinates before applying the pose estimation algorithm.

Figure 12 on the following page shows a line representation of a fully calibrated system consisting four cameras. Once we have individually calibrated each of the four cameras, they can be placed on the same world co-ordinate system provided we use a common calibration chart. The design of the chart shown in Figure 9 is suitable for this approach.

3D surround view output generation

Once we have the system of cameras calibrated, we can render the world around the vehicle by mapping

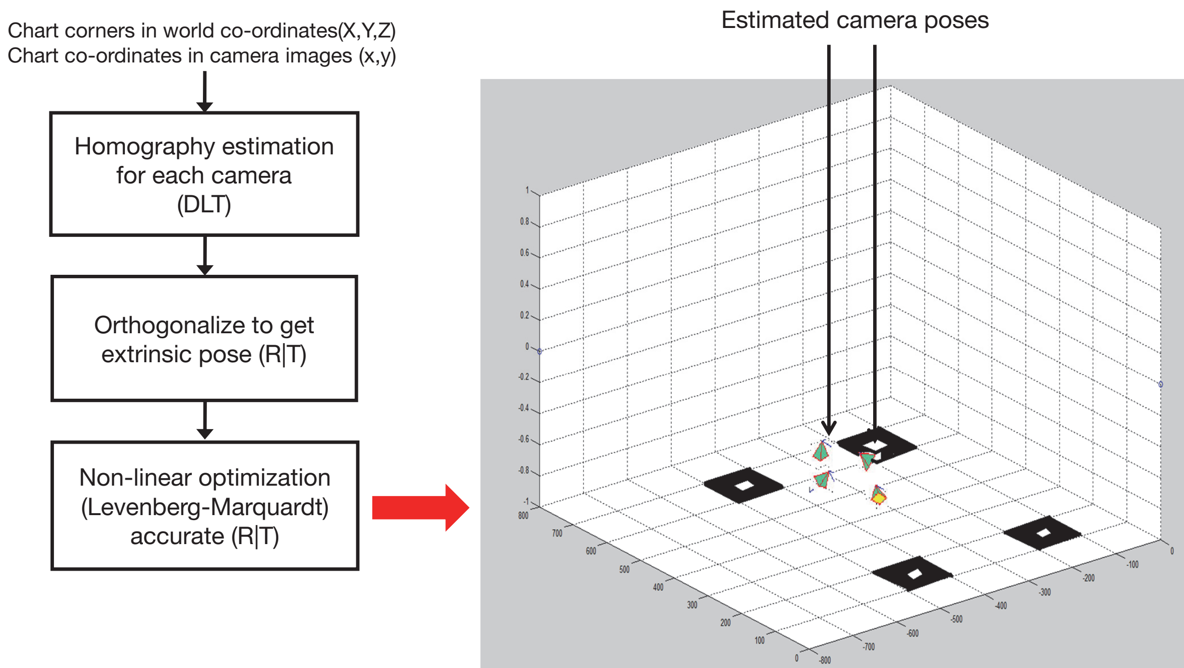

Figure 12: Flow diagram for camera pose estimation using detected chart points and visual illustration of the fully calibrated camera system with a representation of the calibration patterns.

the input images to the 3D world. To enable this, we need to know the depth of the scene around the vehicle. But since the cameras cannot provide depth information, we first need to make an assumption about the world around the vehicle. The 2D surround view example shown in the earlier section assumes that the world around the vehicle is flat and maps texture from the camera images on a flat ground plane. For 3D surround view, we assume that the world around the vehicle can be represented by a bowl. The regions near the vehicle lie on the flat region of the bowl and the regions farther away from the vehicle lie on the elevated surface of the bowl. The shape and parameters of the bowl can be varied to best suit the given use case.

Figure 13 on the following page shows a visual interpretation of the 3D Surround view generation. Once the shape of the bowl is pre-defined, we can map every location on the bowl to a location on the input image with the use of the extrinsic pose of the camera and intrinsic camera parameters. The green region highlighted in the output image falls on the flat region of the bowl and the red regions lie on the elevated surface of the bowl. When we use a fish-eye lens, we also need to take the fish-eye lens model into consideration before applying the camera intrinsic parameters.

Depending on the configuration of the cameras every location in the output bowl can be mapped to one or two adjacent camera. By choosing appropriate blending regions, one can seamlessly stitch information mapped from two adjacent cameras to create a seamlessly blended rendering of the surroundings.

3D mesh table generation

Once the system of cameras are calibrated, we create a LUT mapping the surround view output to the input camera images. We follow the same

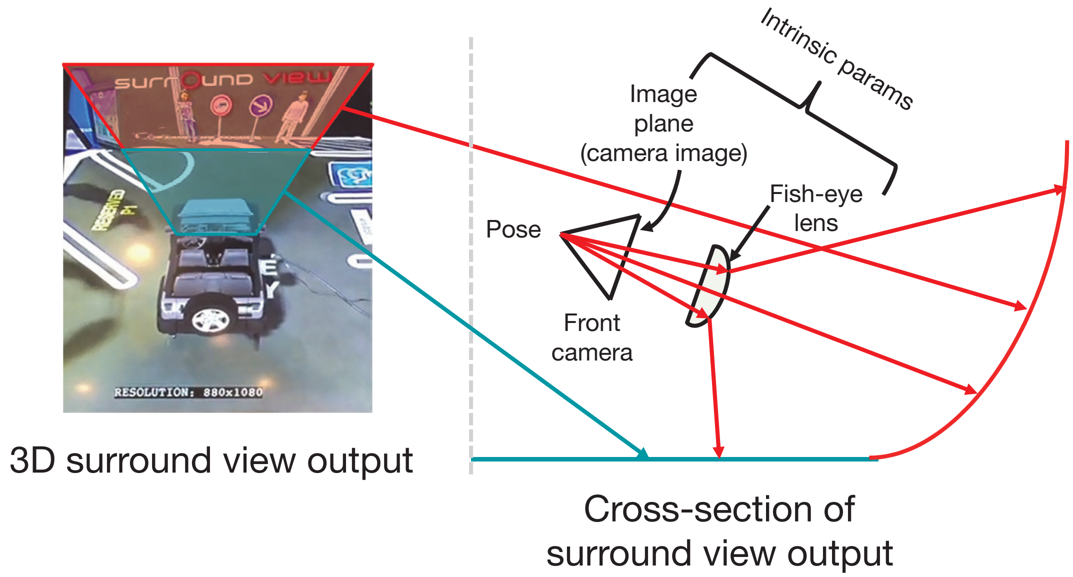

Figure 13: (a) 3D surround view output representation, (b) A cross-section view of the rendered output bowl, which illustrated the mapping of texture and the necessary transformations required.

assumption made in 2D surround view that the cameras positions are fixed after calibration. The mesh table will encode information about the shape of the 3D world around the vehicle as well as texture mapping. This will enable the graphics processor to generate output rendering from various camera viewpoints.

Figure 14 on the following page shows a block diagram for the mesh table generation procedure. The mesh table consists of 3D world co-ordinates for locations in the surrounding of the vehicle and the associated input locations for texture mapping from adjacent cameras viewing the scene for a given location. The output is represented as a bowl, whose height varies as a function of the distance from the center of the vehicle. The collection of the mesh table, which includes output mesh and the associated texture mapping, is passed on to the graphics processor for further rendering. Along with the mesh table, we also generate a blending LUT, which encodes the weights for linear combination of image intensity information received at each location from adjacent cameras. This blending table along with the mesh table is generated once at start-up and stored in memory to be re-used in each from. In the following section we will describe how these tables are used to generate 3D surround view rendering from various viewpoints using the SGX (graphics) core on the embedded device.

GPU rendering

Once the 3D mesh table is generated, it is passed to the OpenGLES part of the application for GPU rendering. GPU used here is Imagination Technologies™ PowerVR™ SGX544 (MP2 or MP1 depending on the device).

The mesh table is read as a set of vertex attributes—vertex coordinates (x, y and z) and texture coordinates for each of the two cameras contributing to a point on 3D surround view. A separate blend table assigns weights to pixel contributions from each camera.

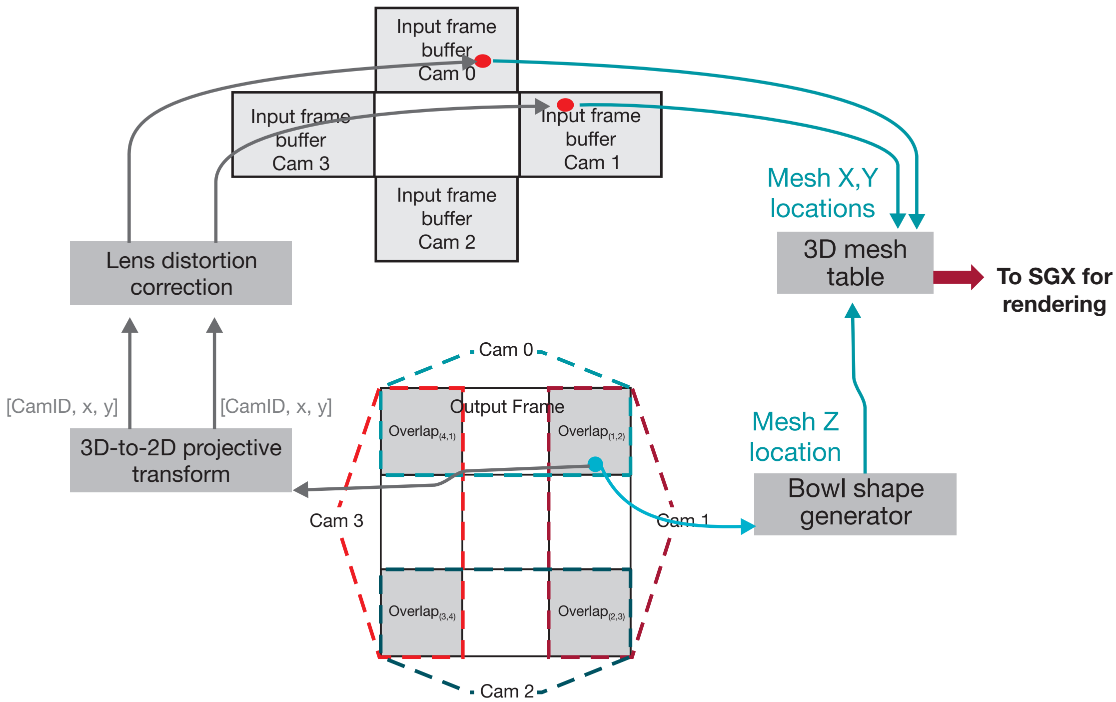

Figure 14: 3D mesh table generation flow diagram. Given a specific output resolution, each output location can be back-mapped to input images via projective and lens transformation to a location in the input image.

Camera images are passed on the GPU using GL_OES_EGL_image_external extension that allows YUV images to be passed on to the GPU as textures.

Car/Jeep model

Car/Jeep model is imported into the OpenGLES application using PowerVR SDK Tools from Imagination Technologies. The model is exported to POD format using PowerVR tools and PowerVR SDK tools provide libraries to import this model in OpenGLES applications. For more information, refer to PowerVR SDK examples and documentation: **http://community.imgtec.com/developers/powervr/graphics-sdk/**

Figure 16 on the following page shows the result of integrating the car model in surround view.

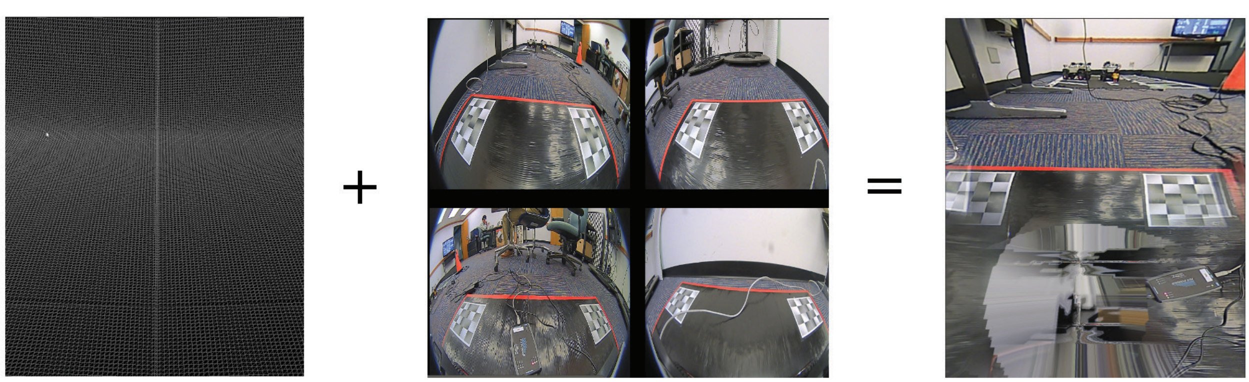

Figure 15: Mesh along with input texture images shown along with final rendered output. Although there are four cameras, each pixel on the final rendered output is generated using only two camera images.

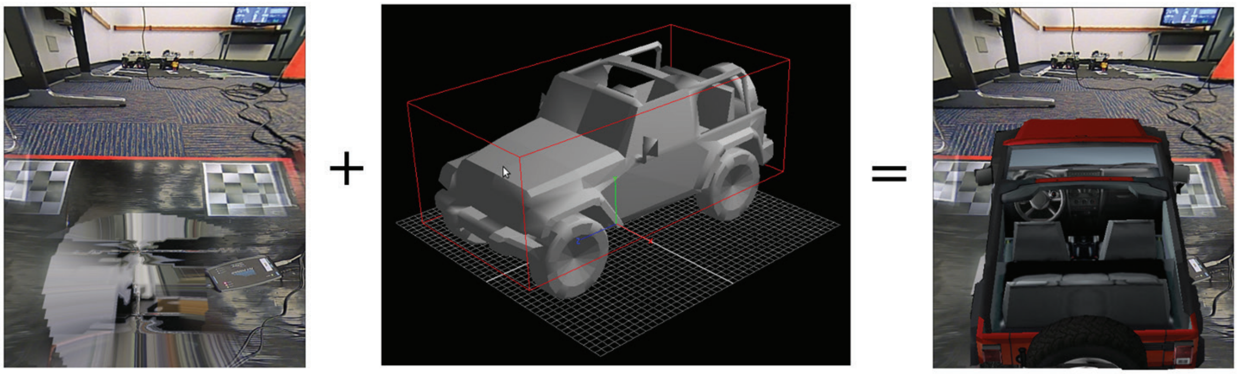

Figure 16: Adding vehicle POD model to surround view application.

TDA2x, TDA2Eco and TDA3x SoCs – TI's offerings for automotive ADAS

TI's TDA2x, TDA2Eco and TDA3x SoCs are a highly optimized and scalable family of devices designed to meet the requirements of leading ADAS. TDA2x, TDA2Eco and TDA3x SoCs enable broad ADAS applications by integrating an optimal mix of performance, low power and ADAS vision analytics processing to facilitate a more autonomous and collision-free driving experience. TDA2x, TDA2Eco and TDA3x SoCs enable applications including front camera, park assist, surround view and sensor fusion on a single architecture. Front camera applications include high-beam assist, lane-keep assist, adaptive cruise control, traffic sign recognition, pedestrian / object detection and collision avoidance. Park-assist applications include intelligent 2D and 3D surround view and rear collision warning and detection. The TDA2x, TDA2Eco and TDA3x SoCs are also capable of the fusion of radar and camera sensor data, allowing for a more robust ADAS decision-making process in the automobile.

Embedded software framework

The surround view solution was implemented and integrated on the TDA2x, TDA2Eco and TDA3x SoC using the **Vision SDK** software framework from TI (see Figure 17 on the following page). Vision SDK is a multi-processor software development platform for TI's family of ADAS SoCs. The software framework allows users to create different ADAS application data flows involving video capture, video pre-processing, video analytics algorithms and video display. Vision SDK is based on a framework named "Links and Chains" framework, and the user API to this framework is called "Link API".

In the surround view use case, the TI Vision SDK shows how to capture video data from multiple cameras, synchronize them, and then "stitch" them together by passing the frames through a multi-stage surround view stitching algorithm. The algorithm is implemented on a DSP and is spread across two DSPs as shown in Figure 18 below.

The TI Vision SDK allows easy integration of the algorithm via algorithm link. It allows the user to test different system partitioning scenarios like 1× DSP vs. 2× DSP without having to rewrite code. It allows algorithm integration without worrying about SoC

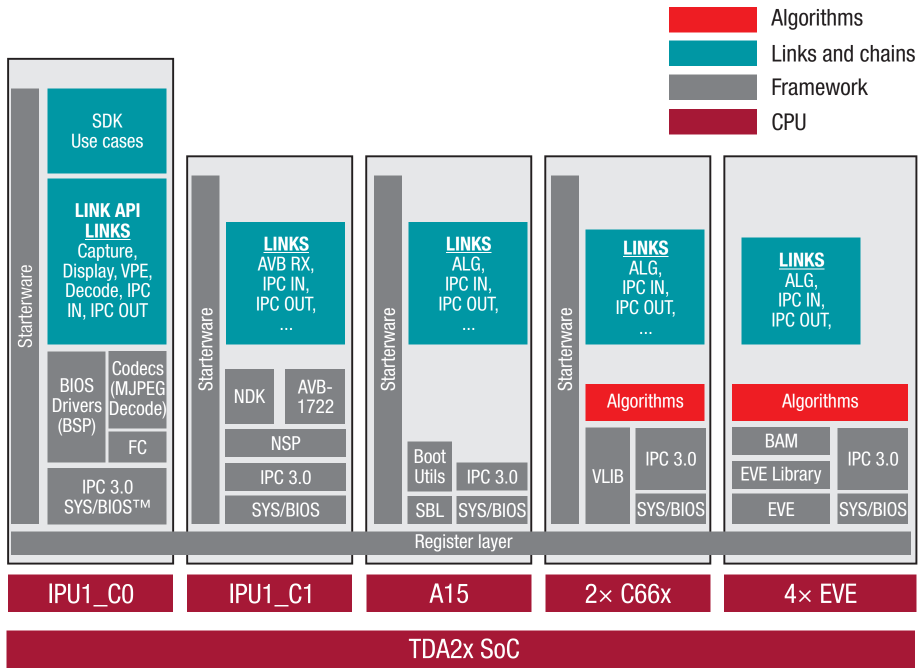

Figure 17: TI Vision SDK stack on TDA2x SoC. TDA3x SoC has a reduced number of DSP cores and Embedded Vision Engines (EVE) and excludes the ARM Cortex-A15. The TDA2Eco has a reduced number of DSP and ARM Cortex-A15 cores and excludes EVE.

details like the number of CPUs in the system, i.e., the algorithm is integrated as if it is getting frames from a task on the same processor. Using the IPC mechanism, the actual frame exchange happens between multiple CPUs.

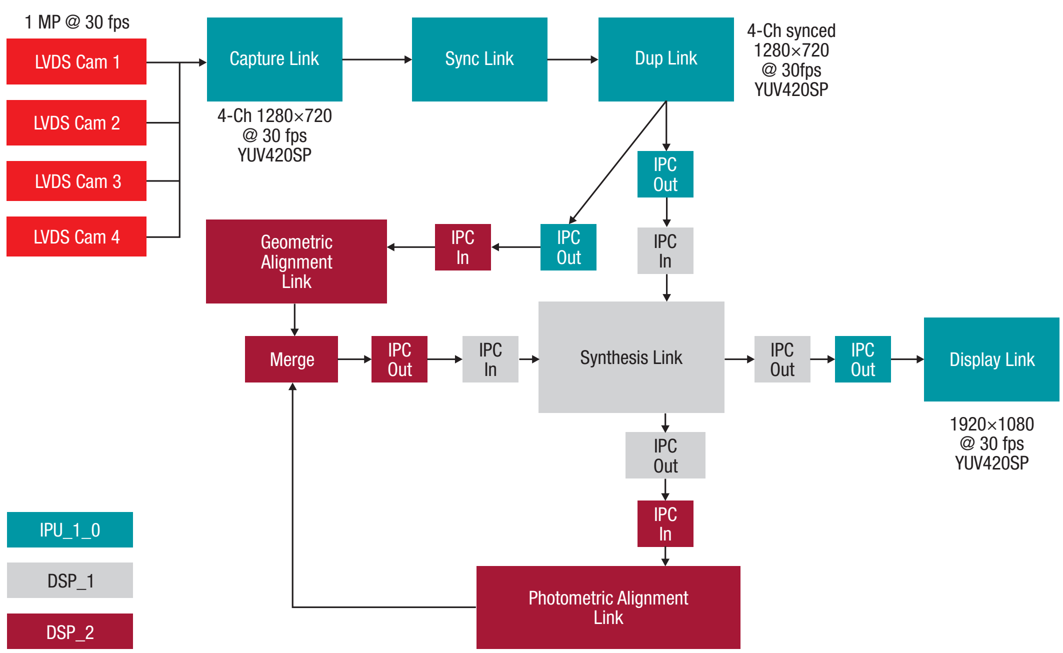

Figure 18: Surround view use-case data flow using Vision SDK.

The Vision SDK framework allows users to extend the use case and add their own algorithms on top of the surround view solution; for example, integrating ultrasonic-sensors with surround view to implement a park-assist application.

Results

Our 2D surround view solution is implemented on C66x DSP on TDA2x, TDA2Eco and TDA3x SoCs. Our system consists of four fish-eye cameras, each having a 180-degree FOV and 720p (1280×720) resolution. The four cameras are mounted on a toy vehicle as shown in Figure 7. From these four fish-eye videos, a composite surround view is synthesized at 30 fps. The composite surround view has a dimension of 880×1080 pixels. The 880×1080 pixel output resolution was chosen to match our display constraints; other output resolutions can be achieved in a similar manner.

The geometric alignment algorithm runs on one DSP C66x (600 MHz @ TDA2x, TDA2Eco), but is called only once at system powering up. It takes about 5 seconds to finish and consumes the entire DSP during this time. The synthesis algorithm (with de-warping and blending operation) runs every frame on a second C66x (600 MHz) core. It takes about 75 percent loading of the 600-MHz DSP for 880×1080 output resolution. The DSP loading for synthesis is a function of the stitched output resolution. The photometric alignment algorithm, which uses image statistics collected during synthesis, runs every frame on the first DSP utilizing 3 percent of its capacity. The composite surround view with varying algorithm complexities are shown in Figure 16: (a), without geometric alignment and photometric alignment; (b), with geometric alignment, but without photometric alignment, and finally, (c), the output with our proposed geometric and photometric alignment algorithms. The proposed surround view solution produces a seamlessly stitched composite view as if it were taken by a camera above the car. A video of the live performance of our real-time surround view prototype can be found at **ti.com/ces-sv-video**

The 3D surround view rendering is implemented on SGX core on the TDA2x, TDA2Eco SoC. It receives four fish-eye cameras images with 720p (1280×720) resolution. From these four fish-eye videos, a composite surround view is synthesized

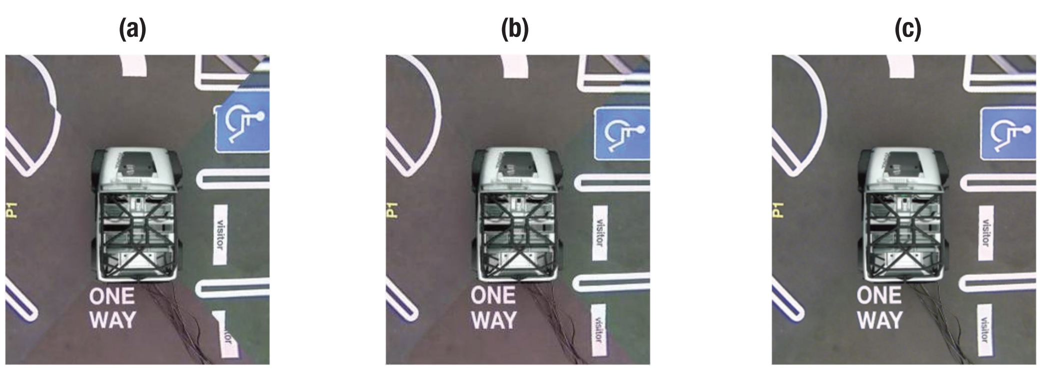

Figure 19: The composite surround view synthesized from four fish-eye frames shown in Figure 1: (a), without proper geometric alignment or photometric alignment; (b), with proper geometric alignment, but without photometric alignment, and (c), with our proposed geometric and photometric alignment algorithms. Without geometric alignment, misalignment at view boundaries is very noticeable. Without photometric alignment, the stitched surround view output suffers color and brightness inconsistency at stitching boundaries. In (c), we achieve a high-quality seamlessly stitched result.

at 30 fps from various virtual viewpoints around the jeep. The composite 3D surround view has a dimension of 880×1080 pixels. The 880×1080 pixel output resolution was chosen to match our display constraints; other output resolutions can be achieved in a similar manner.

The system calibration and mesh table generation algorithms run on one C66x DSP (600 MHz @ TDA2x, TDA2Eco), but is called only once at system powering up. The output from these algorithms are stored in memory and accessed at 30 fps to render output in SGX. Figure 20 shows the rendered output of the 3D surround view from one such free viewpoint. Any view-point in the entire 360 degree surroundings of the car can be similarly generated.

Figure 20: Rendered 3D surround view output of a virtual camera viewpoint of the vehicle from behind.

Conclusions

In this paper, we presented a complete real-time surround view solution on TDA2x, TDA2Eco and TDA3x SoCs for ADAS applications. We describe two different versions of surround view systems: 1) 2D (top-down) surround view and 2) 3D surround view with output rendering from various virtual camera positions around the vehicle. We presented the flow, architecture and the optimization of the entire solution to achieve high-quality stitched HD video output at 30 fps for both 2D as well as 3D surround view systems. We also presented the design of the real-time prototype leveraging TI's FPD-Link III technology.

References

[1] B. Zhang, V. Appia, I. Pekkucuksen, A. Z. Batur, P. Shastry, S. Liu, S. Sivasankaran, K. Chitnis, and Y. Liu, "A surround view camera solution for embedded systems", in *Proc. the 10th IEEE Embedded Vision Workshop* (Held in conjunction with IEEE CVPR 2014), Columbus, Ohio, June 2014

[2] **http://www.ti.com/lit/ml/sprt681/sprt681.pdf**

[3] **http://www.ti.com/lit/ds/symlink/ds90ub913q-q1.pdf**

[4] **http://www.ti.com/lit/pdf/SPRY260**

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as “components”) are sold subject to TI’s terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI’s terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

TI assumes no liability for applications assistance or the design of Buyers’ products. Buyers are responsible for their products and applications using TI components. To minimize the risks associated with Buyers’ products and applications, Buyers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of significant portions of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI components or services with statements different from or beyond the parameters stated by TI for that component or service voids all express and any implied warranties for the associated TI component or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI’s goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or “enhanced plastic” are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have ***not*** been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

Products		**Applications**	
Audio	www.ti.com/audio	Automotive and Transportation	www.ti.com/automotive
Amplifiers	amplifier.ti.com	Communications and Telecom	www.ti.com/communications
Data Converters	dataconverter.ti.com	Computers and Peripherals	www.ti.com/computers
DLP® Products	www.dlp.com	Consumer Electronics	www.ti.com/consumer-apps
DSP	dsp.ti.com	Energy and Lighting	www.ti.com/energy
Clocks and Timers	www.ti.com/clocks	Industrial	www.ti.com/industrial
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Security	www.ti.com/security
Power Mgmt	power.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Applications Processors	www.ti.com/omap	**TI E2E Community**	e2e.ti.com
Wireless Connectivity	www.ti.com/wirelessconnectivity		

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265