Rear-Stitched View Panorama: A low-power embedded implementation for smart rear-view mirrors on vehicles

Janice Pan
The University of Texas at Austin
janicepan@utexas.edu

Vikram Appia
Texas Instruments
vikram.appia@ti.com

Jesse Villarreal
Texas Instruments
jesse.villarreal@ti.com

Lucas Weaver
Texas Instruments
l-weaver@ti.com

Do-Kyoung Kwon
Texas Instruments
d-kwon@ti.com

Abstract

Automobiles are currently equipped with a three-mirror system for rear-view visualization. The two side-view mirrors show close the periphery on the left and right sides of the vehicle, and the center rear-view mirror is typically adjusted to allow the driver to see through the vehicle's rear windshield. This three-mirror system, however, imposes safety concerns in requiring drivers to shift their attention and gaze to look in each mirror to obtain a full visualization of the rear-view surroundings, which takes attention off the scene in front of the vehicle. We present an alternative to the three-mirror rear-view system, which we call Rear-Stitched View Panorama (RSVP). The proposed system uses four rear-facing cameras, strategically placed to overcome the traditional blind spot problem, and stitches the feeds from each camera together to generate a single panoramic view, which can display the entire rear surroundings. We project individually captured frames onto a single virtual view using precomputed system calibration parameters. Then we determine optimal seam lines, along which the images are fused together to form the single RSVP view presented to the driver. Furthermore, we highlight techniques that enable efficient embedded implementation of the system and showcase a real-time system utilizing under 2W of power, making it suitable for in-cabin deployment in vehicles.

1. Introduction

A National Motor Vehicle Crash Causation Survey (NMVCCS) conducted between 2005 and 2007 showed that the critical reason, or the last failure in the causal chain of events leading up to the crash, of approximately 94% of traffic crashes can be assigned to a driver [1]. Research in the scientific community as well as advancements in the automotive industry have been improving safety systems. Particularly within the past decade, more researchers and manufacturers have been working toward developing suites of intelligent on-board applications, called Advanced Driver Assistance Systems (ADAS), to provide assistance to drivers and ultimately improve safety in vehicles and on the roadways. ADAS applications may serve the purpose of simply showing clearer and more useful displays, or they may even make quick, intelligent decisions for the driver in safety-critical situations.

A driver's understanding and awareness of their surroundings is integral in keeping them defensive and safe on the road, so an ADAS application that can increase a driver's field-of-view (FoV) can directly and positively influence the safety of driving. Currently, most cars are equipped with three rear-facing mirrors–two on the vehicle's exterior by the driver (left) and front passenger (right) side windows and one in the vehicle's interior, placed front and center and adjusted to capture the scene through the rear window. In order for the driver to gain a full rear-view visualization, they must take their gaze off the road in front and look left, right, and up, which takes critical time and poses a safety hazard.

We present an alternative to the current three-mirror rear-view visualization system that involves a novel camera configuration and a smart panoramic stitching algorithm for creating a single rear-view display, which we refer to as the Rear-Stitched View Panorama (RSVP). The camera set-up (Figure 2) involves single cameras placed on either side of the vehicle, which serve to replace the two external rear-facing mirrors, as well as a stereo pair of cameras placed externally on the vehicle's rear, which serve to capture the view typically seen through the interior rear-view center mirror. The rear stereoscopic vision provides important information about objects in the rear FoV, which we use to fuse the images as seamlessly as possible.

Another key feature of ADAS systems is reliability. In particular, the performance of embedded systems, such as in-cabin electronic mirror-replacement systems, can be adversely impacted by heat produced by the embedded processor. Thus, for real-time implementation of our proposed algorithm, we use the TDA3x SoC [2], which is a low-power heterogeneous SoC containing a dual core of ARM Cortex-M4, a dual core of C66x DSP, and a single core of Embedded Vision Engine (EVE) for vector processing. It also provides an integrated hardware Image Signal Processor (ISP) for image signal conditioning and other hardware accelerators, such as a 2-D spatial warp engine, and a hardware Display Subsystem (DSS) for compositing multiple video and graphic output streams. The combination of such a diverse set of compute cores along with the hardware peripherals makes it suitable for our implementation. We showcase the partitioning of the proposed algorithm on the device and highlight the key optimization steps that enabled us to satisfy real-time performance needs.

In this paper, we first discuss previous work related to rear-view visualization (Section 2) in ADAS. Then we give an overview of the proposed system (Section 3), explain how the system is calibrated (Section 4), and describe in detail the method for generating the output RSVP image (Section 5). We also discuss the embedded implementation for a real-time ADAS application (Section 6) and conclude with a discussion of limitations, possible modifications, and future work.

2. Related Work

Vision is an integral sensor modality in ADAS and has been used frequently in past research related to driver assistance applications for relaying important information about the environment and the vehicle's surroundings to the driver. Common vision-based ADAS applications focus on detection [3, 4, 5, 6, 7, 8, 9, 10], recognition [11, 12, 13, 14, 15], and tracking [16, 17, 18] of objects, with stereo vision being a popular tool in vision-based approaches [6, 7, 8, 9, 10, 12, 19, 20, 21].

Work has also been done specifically to replace the rear-facing cameras with a single easily-accessible display [22], and further, to combine the feeds from each camera to show a single representative rear-view panoramic image [23, 24]. However, no method integrates stereo cameras to extract depth information. Additionally, while previous work has also been done in finding optimal seam lines and developing seamless stitching methods for mosaics and panoramas [25, 26, 27, 28, 29, 30], most of these methods rely on prior calibration, cost-minimization, or point correspondences. No previous work combines all three or uses raw 3D depth estimates in a scene, going beyond simply using point or feature correspondences, in an intelligent seamless stitching algorithm. Our proposed system learns global alignment through an initial calibration step and then uses stereo estimates to minimize an objective cost function to determine an optimal seam line, along which adjacent image feeds are simply alpha-blended together. We next provide a detailed overview of our system.

3. System Overview

The RSVP display is meant to replace the rear-view mirror system, which is currently an integral feature of most automobiles. Instead of looking in each of the left, right, and center rear-view mirrors, the driver should be able to look at a single display and see *at least* the entire FoV that the trio of mirrors is able to provide. The proposed RSVP system uses four cameras: one each on either side to replace the side external rear-view mirrors and a stereo pair of cameras on the rear to replace the center interior rear-view mirror. Feeds from both side cameras are fused with the feed from the reference (left) stereo camera to generate a seamless panoramic image of the rear surroundings. The right stereo camera is only used for stereo depth computations and is not used for visualization. Figure 2 shows an example configuration of the system.

In order to create this seamless display, the captured feeds are transformed into the same visualization coordinate frame. In other words, the system is calibrated, and the frames are transformed independently so that each image looks as if it was captured with the same *virtual* camera, which in the case of RSVP, may be considered to be located above and in front of the vehicle facing rearwards.

After each image is transformed, they must be combined in such a way that minimizes visual distortion effects and discontinuities in object representation. We refer to the boundaries at which adjacent views are combined as *seams*. We apply a novel seam-finding method that makes use of the available depth information provided by the stereo camera pair on the vehicle's rear. Figure 1 shows the process to create the RSVP view. Our stitching algorithm also includes considerations for temporal sequences of frames, maintaining smooth seam-changing transitions so the output frame sequence neither looks jittery nor contains sudden large seam shifts. This feature is represented by the feedback loop in Fig. 1. We also create a single blending look-up table (LUT), using the computed optimal seam lines, to specify the weights used to fuse images together (in the 'stitching' block in Fig. 1). Each step in the algorithm, from transforming the input frames to designing the blending LUT, was designed for an efficient embedded implementation, as we explain in the following sections.

4. System Calibration

Generating a single panoramic image from three views requires calibration of the system, which enables us to compute each camera's extrinsic parameters, comprising the

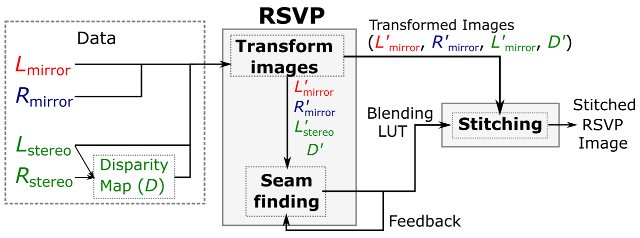


Figure 1: Algorithm flow to create the RSVP image

camera's location and orientation. Using the extrinsic parameters, each view can be transformed, though likely not without distortions, to appear as if it was taken by a virtual camera with arbitrary location and orientation. This virtual camera is what can be considered as the capture device for the generated panoramic image.

A precisely designed set of three calibration charts with known dimensions and relative locations are placed coplanar with the ground plane for system calibration. As shown in Figure 2, the system is aligned with the calibration patterns such that one full square pattern is visible in each independent camera capture. Determining the locations of the eight corner points of each chart in the camera images provides sufficient correspondences to known world coordinates to calibrate the cameras, because every corner in each captured image has an associated 2D coordinate in the image plane and a real-world 3D coordinate. Using these correspondences, we estimate the homography from the camera image plane to world coordinates using a direct linear transformation. Furthermore, projecting the homography matrix on an orthogonal sub-space can provide the extrinsic pose of the camera in the world coordinate 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 approaches to improve the estimation. In our particular example we use the Levenberg-Marquardt approach to refine the camera pose estimations [31]. This calibration is performed offline, and no online calibration is required, because the camera positions are fixed.

5. Panorama Generation

After computing the calibration parameters, projections from each image coordinate system to the world coordinate frame can be computed, and vice versa. The captured frames from each camera can then be transformed into the common coordinate frame of a predetermined virtual camera and fused together. In this section, we discuss the image transformation process, how the optimal seam is selected, and how the three views are stitched together. We also present an approach to maintaining smooth transitions across a temporal sequence of captures.

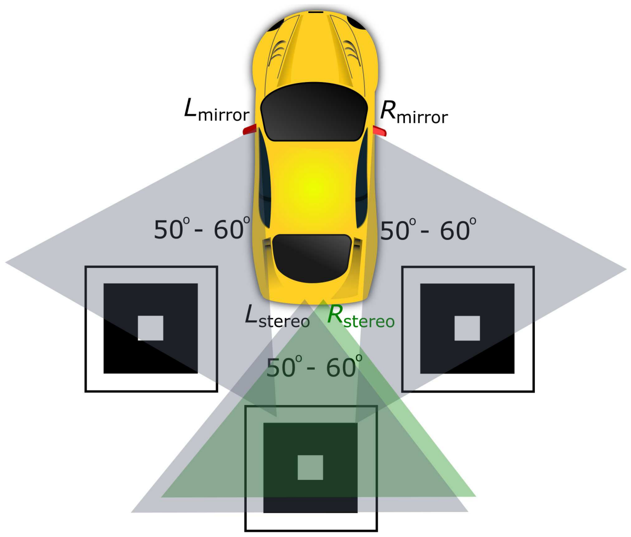


Figure 2: Configuration of system showing the two cameras to replace the side rear-view mirrors (L_{mirror} and R_{mirror}) and the stereo pair on the back of the vehicle (L_{stereo} and R_{stereo}). Three calibration charts are also shown as an example of a pattern that can be used to calibrate the system.

5.1. Image Transformation

To generate a spatially-coherent output RSVP image, each camera's intrinsic and extrinsic parameters are used to transform the originally captured images to the virtual camera view. This process involves first projecting each image pixel onto a flat surface behind the vehicle and then using the predetermined virtual camera parameters to project it into the virtual camera image. We can think of the virtual world around the vehicle as represented by a flat surface located some distance behind the vehicle, onto which the appropriate FoV from each camera feed is projected. Other world representations, e.g., cylindrical surfaces, could also be used.

The right stereo camera is only used to compute depth, so we first obtain the disparity map corresponding to the reference image and then apply the same transformations on the disparity map that were applied to the left stereo image to obtain a virtual camera 'view' of the disparity as well. Figure 3 shows an example set of captured images and the pipeline that generates the output RSVP image. After the images are transformed, the FoV of interest from each image, outlined in red, are combined by stitching along seams, the locations of which we determine with the seam detection algorithm detailed in the following section.

Because we project the RSVP image on a virtual flat surface behind the vehicle, a single projective transform can be computed for each of the three visualization views (L_{mirror}, L_{stereo}, and R_{mirror}). Specifically, for each input from L_{mirror}, L_{stereo}, and R_{mirror}, we use four non-collinear points

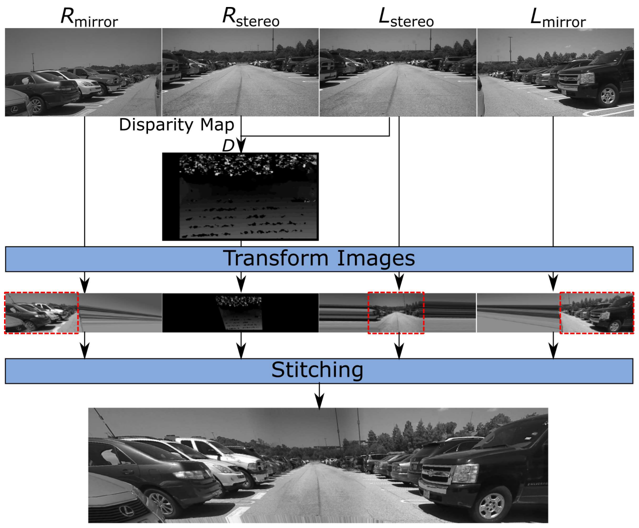


Figure 3: Pipeline of RSVP generation

in the RSVP image $\{p_{ti}\}$, compute the world-coordinate projections of those image points using the virtual camera parameters, and then project those world points back into the input image to get points $\{p_{oi}\}$. We then compute a projective transformation to map $\{p_{oi}\}$ to $\{p_{ti}\}$. We thereby obtain a separate projective transformation matrix for each input, which can efficiently transform input frames as they are continuously being captured.

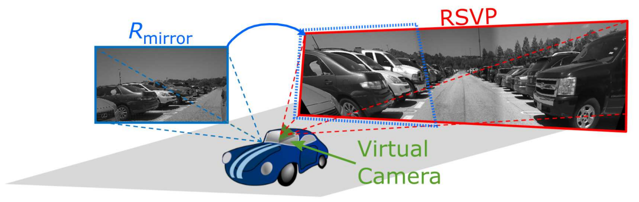


Figure 4: Visualization of projecting the right mirror image onto the RSVP display surface

If the surface onto which the RSVP image is projected changes over frames, the transformations will need to be dynamically computed, which may induce spatial artifacts, as objects in the scene may appear to shift around, causing the representation of relative distances to become confusing. Maintaining a static RSVP projection surface is a safety-critical feature, so the displayed output is always easy for a driver to interpret.

5.2. Seam Detection and Image Stitching

The output RSVP image is a combination of the views from cameras L_{mirror}, L_{stereo}, and R_{mirror}. Camera R_{stereo} is used as part of the stereo system with L_{stereo}, and the depth information extracted from these two views are used to determine the optimal boundaries along which adjacent views are combined, which we refer to as *seam lines*.

Because the RSVP image is essentially a projection onto a flat surface with a predetermined depth, when seams are selected such that they pass through objects *at* the surface distance, those objects appear seamlessly stitched in the output image. In other words, in the process of projecting from each captured image to the virtual surface, a world location that lies on the virtual surface and is captured by adjacent cameras will project back to the same world location, and thus look aligned, in the virtual view. Therefore, we use stereo information to locate points in world space that are close to the projection surface, so that a seam line that runs through their corresponding image points can be computed.

Specifically, we compute disparities for each point on the flat surface, D_W, using its known location relative to the stereo cameras. We then compare this flat surface of disparity values with the transformed disparity map, D', which was computed between L_{stereo} and R_{stereo} and corresponds to 3D locations of the scene captured in the stereo cameras. To perform this comparison, we simply take the magnitude of their difference:

$$D_\Delta = |D' - D_W|. \tag{1}$$

Therefore, for each seam candidate $s = \{(x_i, y_i)\}$, defined by a set of pixels in D_Δ, we define the accumulated matching cost of the seam as $D_\Delta(s)$ and normalize by the number of pixels in s, for which a valid disparity estimate exists, to obtain cost c_s. Specifically, to compute the cost c_s of any candidate seam s, with discretized points $\{(x_1, y_1), ..., (x_{h_{\text{RSVP}}}, y_{h_{\text{RSVP}}})\}$, we use the following equation:

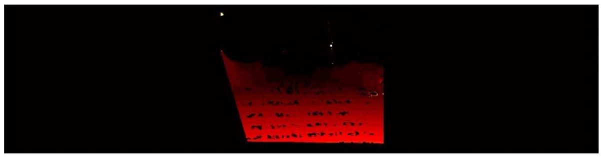
$$c_s = \frac{\sum_{i=1:h_{\text{RSVP}}, D'(x_i,y_i)>0} D_\Delta(x_i, y_i)}{|D'(s) > 0|}, \tag{2}$$

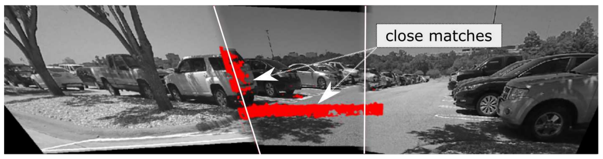
where $|D'(s) > 0|$ is the number of points in s with valid disparity estimates, and h_{RSVP} is the height of the output panorama in pixels in the transformed domain. Each seam line candidate is specified by h_{RSVP} points (one per row). Our output display is 1920×480, so $h_{\text{RSVP}} = 480$ in our implementation. The minimum-cost seam, $s_{\text{min},W}$, that stitches any two views together is the linear path with the lowest cost that lies within the overlapping region between the views:

$$s_{\text{min},W} = \underset{s}{\operatorname{argmin}}\, c_s. \tag{3}$$

Figure 5 shows an example of the transformed disparity map and the points in 3D it helped identify as being the appropriate points along which to combine adjacent views as seamlessly as possible.

In cases where there are no significant objects at the depth of the specified flat surface, there will be insufficient matches between projection surface disparities and the transformed disparity image. The seam line on the right in Fig. 5b, for example, was chosen somewhat arbitrarily, because the ground plane dominated the set of 3D points at the virtual surface

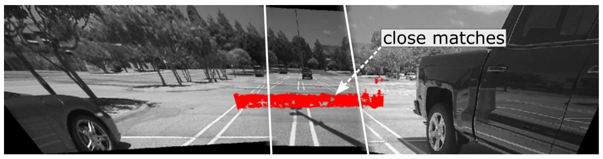

(a)



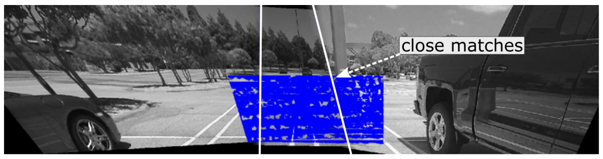
(b)

Figure 5: Example showing how disparities can be used to identify seam lines. (a) Transformed disparity map; (b) Low-cost pixels with min-cost seam lines.

location, so all seam candidates between the right two image feeds were equally poor choices. In such instances, we instead attempt to stitch along the ground plane, thereby finding a seam that avoids objects in the scene. The same method for determining $s_{\text{min},W}$ can be applied to find $s_{\text{min},G}$, the min-cost seam along the ground. Instead of matching the transformed disparity map to the flat surface disparities, we match it to ground-plane disparities, which are pre-calculated based on the calibrated stereo camera pair. Figure 6 shows an example scene where there are no significant objects at the surface distance, so picking a seam line according to surface disparities (Fig. 6a) is arbitrary. In these cases, we match ground-plane disparities (Fig. 6b).



(a)



(b)

Figure 6: Selecting seams using (a) virtual surface disparities, (b) ground disparities. In this scene, the seams along the ground plane preserve the pole structure visible in the right image feed in (b), which is not visible in (a).

To select the overall min-cost seam, we select the minimum between the seams computed using matching the flat surface disparities and matching the ground disparities, with optional algorithmic constraints. A bias towards using the surface or ground can be used if one method is preferred over the other. Additionally, when computing matching flat surface disparities, we can impose a constraint to only allow surface pixels with disparity estimates less than some Δ_w to contribute to computing the cost of the seam. Without constraining the valid disparity pixels using Δ_w, seam selection will be biased towards matching along the ground, because the ground is a dominant feature in the majority of driving scenes, so ground-plane matches are generally more prevalent. This constraint can be used to modify the cost computation of a seam candidate s:

$$c_s = \frac{\sum_{i=1:h_{\text{RSVP}},D'(x_i,y_i)<\Delta_\text{w}} D_\Delta(x_i,y_i)}{|0 < D'(s) < \Delta_\text{w}|}, \tag{4}$$

and the overall min-cost seam is simply

$$s_{\text{min}} = \min(s_{\text{min},W}, s_{\text{min},G}), \tag{5}$$

which is computed independently for each RSVP frame.

5.3. Temporal Consistency

We also present a method for temporally smoothing the seam movement, because there may be large jumps and jitter in the stitching boundary, which could be distracting and disorienting to the driver, if new seams are computed independently in each frame. To maintain temporal consistency in an RSVP video sequence, we add a feedback loop to our seam-detection algorithm to consider the seam location in the previous frame, as shown in Fig. 1.

With the goal to minimize both unnecessary jitter as well as large jumps in seam locations between frames, we define minimum and maximum distance thresholds, p_{min} and p_{max}, for seam movement. If no points in s_{min} are at least p_{min} pixels away from the seam in the previous frame (s_{t-1}), we do not update the seam, i.e., $s_t = s_{t-1}$.

However, if any points in s_{min} are more than p_{max} pixels away from s_{t-1}, then s_{min} is scaled such that the largest pointwise displacement between s_{min} and s_{t-1} is p_{max}. Specifically, if the min-cost seam is s_{min} at a time t, and we consider the points comprising each seam:

$$\begin{aligned} s_{\text{min}} &= \{(x_{1,\text{min}}, y_{1,\text{min}}), ..., (x_{h_{\text{RSVP}},\text{min}}, y_{h_{\text{RSVP}},\text{min}})\}, \\ s_{t-1} &= \{(x_{1,t-1}, y_{1,t-1}), ..., (x_{h_{\text{RSVP}},t-1}, y_{h_{\text{RSVP}},t-1})\}, \\ s_t &= \{(x_{1,t}, y_{1,t}), ..., (x_{h_{\text{RSVP}},t}, y_{h_{\text{RSVP}},t})\}, \end{aligned} \tag{6}$$

with $h_{\text{RSVP}} = 480$, then the x-coordinates of s_t can be computed by scaling s_{min} as follows:

$$\begin{aligned} \Delta_{x1} &= |x_{1,\text{min}} - x_{1,t-1}|, \\ \Delta_{xh_{\text{RSVP}}} &= |x_{h_{\text{RSVP}},\text{min}} - x_{h_{\text{RSVP}},t-1}|, \\ r &= \frac{\max(\Delta_{x1}, \Delta_{xh_{\text{RSVP}}})}{p_{\text{max}}}, \\ x_{i,t} &= x_{i,t-1} + \frac{x_{i,\text{min}} - x_{i,t-1}}{r} \quad \forall\, i = [1, 480]. \end{aligned} \tag{7}$$

Figure 7 shows examples of how these displacement thresholds can affect the way the new seam is chosen. When all the points of the new min-cost seam line, s_{min}, are within p_{min} pixels from the previous seam, s_{t-1}, then $s_t = s_{t-1}$ and remains unchanged (Fig. 7a). When s_{min} is computed to be far away from s_{t-1}, then instead of instantly jumping to the new location in frame t, s_t is computed such that over a sequence of frames, it gradually moves toward s_{min}.

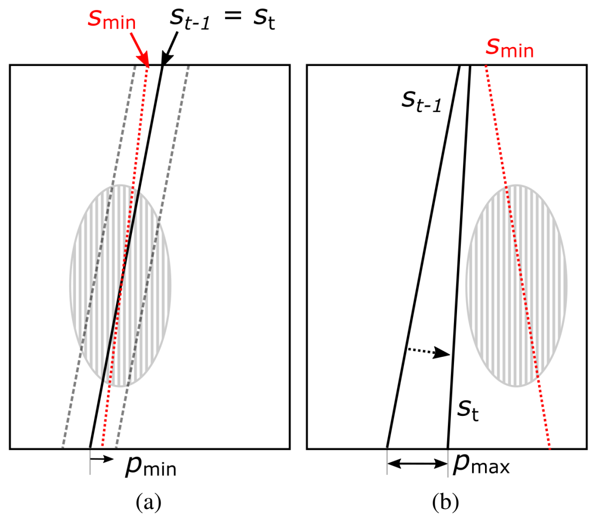


Figure 7: (a) To avoid jitter, the minimum movement threshold is used. (b) To avoid a large jump to a distant seam line, the new s_{min} is scaled such that s_t moves *toward* the min-cost seam.

We considered using an alternate approach that computes a weighted mean between a newly-computed s_{min} and the previous s_{t-1}, i.e.,

$$s_t = s_{\text{min}} + \alpha s_{t-1}, \tag{8}$$

where α is chosen to bias the system towards either picking the min-cost seam or maintaining more temporal consistency. However, this method does not allow the seam location to truly reach an optimal min-cost seam that is farther away from s_{t-1}, as it will converge to somewhere in between. By using our algorithm for updating s_t by scaling the seam location based on p_{max} and allowing for displacements of p_{max}, we avoid this problem.

5.4. Stitching

Once s_t is computed for each of the two overlapping regions (one between R_{mirror} and L_{stereo} and one between L_{stereo} and L_{mirror}), we compute a blending LUT with weights specifying the alpha-blending coefficients for combining pairs of images together. The alpha values are computed such that they are 0.5 for each point in s_t and linearly decrease from 1 to 0 over a blending width around the seam. We predefine the blending width, but it could be changed dynamically over frames with knowledge about the scene structure in an extension to this algorithm.

We only require a single blending LUT the same size as the output RSVP image, because the left seam (between R_{mirror} and L_{stereo}) always falls in the left half of the RSVP image, and the right seam (between L_{stereo} and L_{mirror}) always falls in the right half of the RSVP image. Thus, we can create the blending LUT under these assumptions and define the weights with respect to the *left* image of each pair that are being combined. An example of a blending LUT is shown in Figure 8, and red lines are overlayed to indicate seam locations. If the LUT is represented by

$$\{W_{ij}, i \in \{1, ..., h_{\text{RSVP}}\}, j \in \{1, ..., w_{\text{RSVP}}\}\},$$

then the stitching block (in Fig. 1) will perform the following computations for the left and right halves of the output RSVP image, respectively:

- For $j = \{1, ..., \frac{w_{\text{RSVP}}}{2}\}$ and $\forall i$,

  $$\text{RSVP}(i,j) = W_{ij} R'_{\text{mirror}}(i,j) + (1 - W_{ij}) L'_{\text{stereo}}(i,j), \tag{9}$$

- For $j = \{\frac{w_{\text{RSVP}}}{2} + 1, ..., w_{\text{RSVP}}\}$ and $\forall i$,

  $$\text{RSVP}(i,j) = W_{ij} L'_{\text{stereo}}(i,j) + (1 - W_{ij}) L'_{\text{mirror}}(i,j). \tag{10}$$

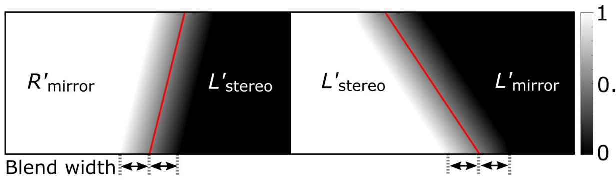


Figure 8: Blending LUT example with seam lines overlayed for visualization. The weight at each point on both seam lines is 0.5.

6. Embedded Implementation

Figure 9 shows the data flow for our embedded implementation on the TDA3x SoC. We used four raw Bayer image sensors with a capture resolution of 1280×720 at 30 fps while the display resolution is 1920×480 at 30 fps. The image sensors are connected via FPD-Link to the SoC [2], where TDA3x's hardware ISP processes the video frames, enabling centralized control of exposure and white balance for all camera images. This produces a seamlessly stitched image output with no color or intensity irregularities at the seam (Figures 10 and 11). These images are then passed to the 2D warp engine for Image Transformation (Section 5.1) and Image Rectification needed for the Stereo Engine [32]. The 2D warp engine is a flexible and efficient back-mapping hardware accelerator for 2D images, which we use for perspective transformations and rectifications of the images and disparity map. By using bilinear interpolation and choosing optimal block sizes, the total compute time for all image transformations and rectifications on the 2D warp engine was minimized to 29.35 ms, which fits within our 30 fps budget.

The image-rectified stereo pair is passed to the Stereo Engine running on the EVE and DSP1, which creates the disparity map needed for the Seam Detection algorithm in 30.8 ms. We use a stereo baseline of approximately 10 cm and standard semi-global matching for our stereo algorithm, but other dense correspondence methods may be used instead. The Seam Detection and Stitching algorithms require the perspective-transformed stereo disparity map in addition to the transformed images as inputs. In order to meet the real-time latency requirements, we use the disparity map from the previous frame ($t-1$) to detect the optimal seam location in the current output frame (t).

Furthermore, as described in Section 5.2, we search for the optimal seam by computing a cost on the disparity map for each candidate seam. Here we use a projection surface depth of 15 meters, measured from the stereo camera baseline, i.e., the rear of the vehicle. To reduce the compute complexity we use a coarse-to-fine search strategy, which identifies the optimal vertical seam location and searches for other linear (non-vertical) seams in the small neighborhood of the identified best vertical seam location. Although this strategy reduces computations, a vertical seam search is not cache friendly for an embedded system. To improve cache efficiency, we transpose the disparity map in the 2D warp engine as part of the image transformation step, which converts the search space to horizontal seams.

Since the location of the stitching seam can change in each new frame, we must dynamically generate a new blending LUT. The blending LUT is only computed for the region of the output where the center image overlaps with the left and right images. These optimizations, along with targeted DSP intrinsic optimizations in the blending LUT generation loop, reduce the compute time for the Seam Detection algorithm to 9.5 ms on DSP2.

In the Stitching algorithm, the nonoverlapping portions of the left and right images are simply copied to the output image using efficient hardware Direct Memory Access (DMA) memory transfers, since there is no blending in these regions. For the blended regions, the algorithm was optimized using DSP intrinsic functions to take full advantage of the C66x SIMD architecture. The core software loops performing the alpha-blend operation blends almost two output pixels per one DSP cycle. Since the compute in this case is faster than the memory access rate through the cache, the DMA engine was employed using a ping-pong buffer scheme to bring blocks of both images and the blending LUT from DDR into the DSP2 L2 RAM and to write the output back from DSP L2 RAM to DDR. With this scheme, the DDR data access time and the blending compute time overlap, thereby reducing the overall time for the stitching algorithm to process each output frame to 4.3 ms on DSP2.

Finally, we also note that the resulting RSVP image that is displayed to the driver should be overlayed with a visualization of part of the vehicle to orient the driver with a sense of relative distances between objects in the scene, as demonstrated in Figure 10. This overlay is fixed and loaded in DDR memory in RGB565 format. The Display Subsystem (DSS) supports reading three video streams and one graphics overlay for color conversion, blending, scaling, and compositing on the final video output port. In our implementation, we use one video stream for the stitched video output and one graphics overlay for the vehicle visualization. The DSS graphics overlay is configured with an alpha blend in hardware which provides an efficient implementation of transparent overlays. With these optimizations, we achieve real-time performance with a system latency of 124 ms.

7. Discussion

Because this RSVP display is intended for automotive applications, safety is, of course, critical. The typical way drivers learn to adjust their rear-view mirrors creates huge blind spots just past the FoV of their periphery, which drivers can only see if they look over their shoulders. The camera configuration in RSVP reduces the extent of this blind spot significantly; however, it also introduces two additional blind spots at either corner of the rear bumper. Figure 12 illustrates this issue. An object right behind the bumper but outside of the FoV of either stereo camera will not be captured by any camera. Additionally, if an object is right next to the bumper and captured in either L'_{mirror} or R'_{mirror}, stitching the RSVP image together could remove the object from the display entirely, or the object could appear ghosted, both of which are problems. However, if the vehicle is traveling at low enough speeds and/or in reverse, other integral parts of ADAS, such as surround-view, parking assist, or backup assist, would be preferred applications to use over RSVP.

Also, increasingly popular with ADAS is combining information from different sensor modalities. If a radar sensor is able to detect the presence of an object visible in either blind spot captured by L'_{mirror} or R'_{mirror} but not captured by the stereo cameras, the seam selection algorithm can be modified to bias the respective seam toward the center of the output RSVP image, which would ensure as much of L'_{mirror} or R'_{mirror} is included in the output as possible, so the object will indeed be projected in the RSVP display.

8. Conclusion

We have introduced a novel system involving a new camera configuration and RSVP-generation algorithm to replace the three rear-facing mirrors in most current automobiles. Our camera system comprises (1) two side cameras–one on either side of the vehicle–to replace the side rear-view

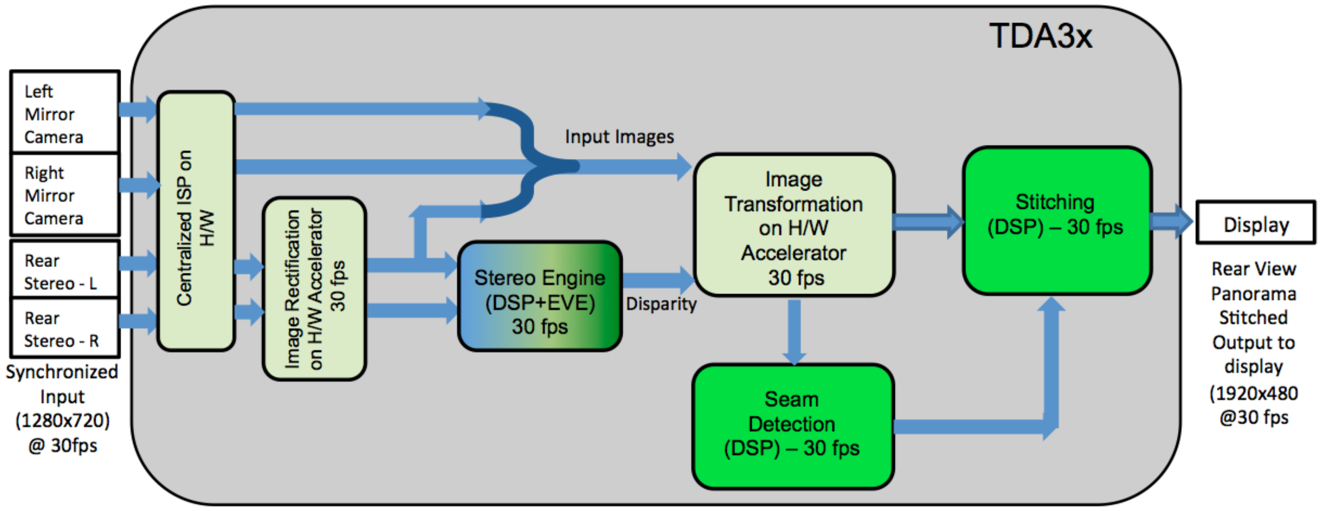

TDA3x
Left Mirror Camera
Right Mirror Camera
Rear Stereo - L
Rear Stereo - R
Synchronized Input (1280x720) @ 30fps
Centralized ISP on H/W
Image Rectification on H/W Accelerator 30 fps
Input Images
Stereo Engine (DSP+EVE) 30 fps
Disparity
Image Transformation on H/W Accelerator 30 fps
Stitching (DSP) – 30 fps
Seam Detection (DSP) – 30 fps
Display
Rear View Panorama Stitched Output to display (1920x480 @30 fps

Figure 9: Block diagram for embedded implementation

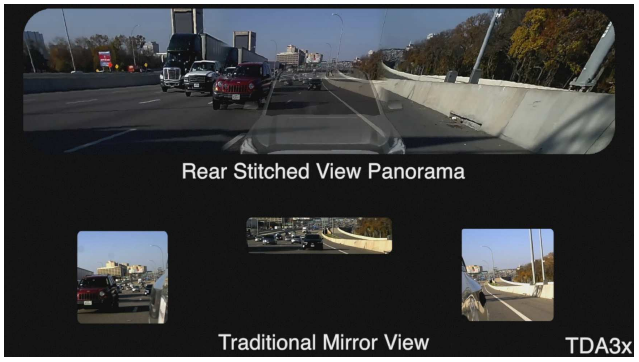


Figure 10: Screenshot of the embedded demo comparing the RSVP view and the traditional three-mirror view

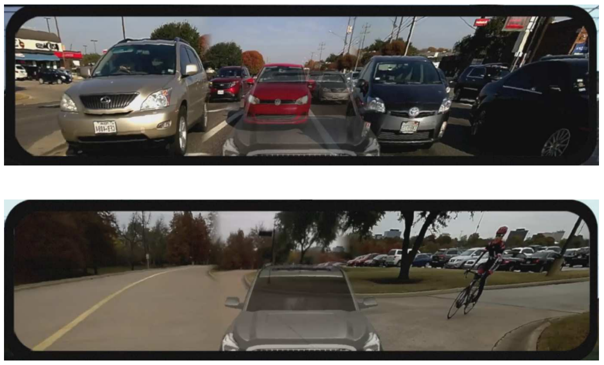


Figure 11: Examples of generated RSVP images

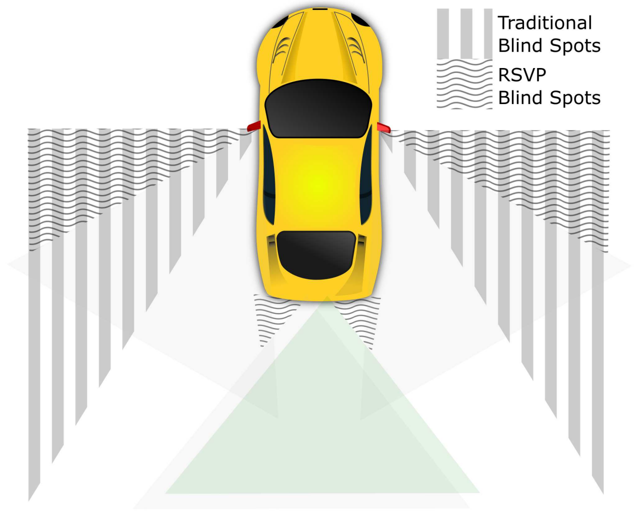


Figure 12: Comparison between blindspots in the traditional rear-view mirror system and blindspots in RSVP

mirrors, and (2) a stereo pair of cameras on the vehicle's rear to replace the center rear-view mirror looking through the rear window and to provide a means of extracting depth information from the scene. After initial calibration of the entire system, transformations of each camera feed can be computed to project the captured images onto a virtual surface, which represents the output RSVP image and can be thought of as being captured by a single rear-facing virtual camera.

The three images used for visualization (L_{mirror}, R_{mirror}, and L_{stereo}) and the disparity map, computed between the stereo image pair, are transformed and passed to our seam-detection algorithm, which uses disparity matching to then determine optimal min-cost seam lines along which to combine adjacent camera frames. Then a single weighting LUT is generated using prior knowledge of where each camera projects in the output, so the images can be combined efficiently. Our embedded implementation of this algorithm demonstrates the efficiency and feasibility of integrating a real-time RSVP display in current ADAS systems.

References

[1] Santokh Singh, "Critical reasons for crashes investigated in the national motor vehicle crash causation survey," Tech. Rep., 2015.

[2] Prashanth Viswanath, Kedar Chitnis, Pramod Swami, Mihir Mody, Sujith Shivalingappa, Soyeb Nagori, Manu Mathew, Kumar Desappan, Shyam Jagannathan, Deepak Poddar, Anshu Jain, Hrushikesh Garud, Vikram Appia, Mayank Mangla, and Shashank Dabral, "A diverse low cost high performance platform for advanced driver assistance system (adas) applications," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2016.

[3] Hunjae Yoo, Ukil Yang, and Kwanghoon Sohn, "Gradient-enhancing conversion for illumination-robust lane detection," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 3, pp. 1083–1094, 2013.

[4] Andreas Mogelmose, Mohan Manubhai Trivedi, and Thomas B Moeslund, "Vision-based traffic sign detection and analysis for intelligent driver assistance systems: Perspectives and survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 4, pp. 1484–1497, 2012.

[5] David Geronimo, Antonio M Lopez, Angel D Sappa, and Thorsten Graf, "Survey of pedestrian detection for advanced driver assistance systems," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 7, pp. 1239–1258, 2010.

[6] Massimo Bertozzi and Alberto Broggi, "Gold: A parallel real-time stereo vision system for generic obstacle and lane detection," *IEEE transactions on image processing*, vol. 7, no. 1, pp. 62–81, 1998.

[7] Massimo Bertozzi, Alberto Broggi, Alessandra Fascioli, and Stefano Nichele, "Stereo vision-based vehicle detection," in *Intelligent Vehicles Symposium, 2000. IV 2000. Proceedings of the IEEE*. IEEE, 2000, pp. 39–44.

[8] Raphael Labayrade, Didier Aubert, and J-P Tarel, "Real time obstacle detection in stereovision on non flat road geometry through" v-disparity" representation," in *Intelligent Vehicle Symposium, 2002. IEEE*. IEEE, 2002, vol. 2, pp. 646–651.

[9] Florin Oniga and Sergiu Nedevschi, "Processing dense stereo data using elevation maps: Road surface, traffic isle, and obstacle detection," *IEEE Transactions on Vehicular Technology*, vol. 59, no. 3, pp. 1172–1182, 2010.

[10] Cosmin D Pantilie, Silviu Bota, Istvan Haller, and Sergiu Nedevschi, "Real-time obstacle detection using dense stereo vision and dense optical flow," in *Intelligent Computer Communication and Processing (ICCP), 2010 IEEE International Conference on*. IEEE, 2010, pp. 191–196.

[11] Arturo De la Escalera, J Ma Armingol, and Mario Mata, "Traffic sign recognition and analysis for intelligent vehicles," *Image and vision computing*, vol. 21, no. 3, pp. 247–258, 2003.

[12] Uwe Franke, Dariu Gavrila, Steffen Gorzig, Frank Lindner, F Puetzold, and Christian Wohler, "Autonomous driving goes downtown," *IEEE Intelligent Systems and Their Applications*, vol. 13, no. 6, pp. 40–48, 1998.

[13] Andrzej Ruta, Yongmin Li, and Xiaohui Liu, "Real-time traffic sign recognition from video by class-specific discriminative features," *Pattern Recognition*, vol. 43, no. 1, pp. 416–430, 2010.

[14] Xiaohong W Gao, Lubov Podladchikova, Dmitry Shaposhnikov, Kunbin Hong, and Natalia Shevtsova, "Recognition of traffic signs based on their colour and shape features extracted using human vision models," *Journal of Visual Communication and Image Representation*, vol. 17, no. 4, pp. 675–685, 2006.

[15] Giulia Piccioli, Enrico De Micheli, Pietro Parodi, and Marco Campani, "Robust method for road sign detection and recognition," *Image and Vision Computing*, vol. 14, no. 3, pp. 209–223, 1996.

[16] Jannik Fritsch, Tobias Kuhnl, and Andreas Geiger, "A new performance measure and evaluation benchmark for road detection algorithms," in *Intelligent Transportation Systems-(ITSC), 2013 16th International IEEE Conference on*. IEEE, 2013, pp. 1693–1700.

[17] Marcos Nieto, Andoni Cortés, Oihana Otaegui, Jon Arróspide, and Luis Salgado, "Real-time lane tracking using rao-blackwellized particle filter," *Journal of Real-Time Image Processing*, vol. 11, no. 1, pp. 179–191, 2016.

[18] David Pfeiffer and Uwe Franke, "Efficient representation of traffic scenes by means of dynamic stixels," in *Intelligent Vehicles Symposium (IV), 2010 IEEE*. IEEE, 2010, pp. 217–224.

[19] Janice Pan, Vikram Appia, and Alan C Bovik, "Virtual top-view camera calibration for accurate object representation," in *Image Analysis and Interpretation (SSIAI), 2016 IEEE Southwest Symposium on*. IEEE, 2016, pp. 21–24.

[20] Sergiu Nedevschi, Radu Danescu, Ciprian Pocol, and Marc Michael Meinecke, "Stereo image processing for adas and pre-crash systems," in *International Workshop on Intelligent Transportation*, 2008, pp. 55–60.

[21] Mohamed El Ansari, Stéphane Mousset, Abdelaziz Bensrhair, and George Bebis, "Temporal consistent fast stereo matching for advanced driver assistance systems (adas)," in *Intelligent Vehicles Symposium (IV), 2010 IEEE*. IEEE, 2010, pp. 825–831.

[22] James O Secor, "Consolidated rear view camera and display system for motor vehicle," Feb. 22 1994, US Patent 5,289,321.

[23] Alfredo Ramirez, Eshed Ohn-Bar, and Mohan Trivedi, "Panoramic stitching for driver assistance and applications to motion saliency-based risk analysis," in *Intelligent Transportation Systems-(ITSC), 2013 16th International IEEE Conference on*. IEEE, 2013, pp. 597–601.

[24] Kenneth Schofield, Mark L Larson, and Keith J Vadas, "Rearview vision system for vehicle including panoramic view," Sept. 23 1997, US Patent 5,670,935.

[25] Yuri Boykov, Olga Veksler, and Ramin Zabih, "Fast approximate energy minimization via graph cuts," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 23, no. 11, pp. 1222–1239, 2001.

[26] Fan Zhang and Feng Liu, "Parallax-tolerant image stitching," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 3262–3269.

[27] Aseem Agarwala, Mira Dontcheva, Maneesh Agrawala, Steven Drucker, Alex Colburn, Brian Curless, David Salesin, and Michael Cohen, "Interactive digital photomontage," in *ACM Transactions on Graphics (TOG)*. ACM, 2004, vol. 23, pp. 294–302.

[28] Matthew Uyttendaele, Ashley Eden, and Richard Skeliski, "Eliminating ghosting and exposure artifacts in image mosaics," in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*. IEEE, 2001, vol. 2, pp. II–II.

[29] James Davis, "Mosaics of scenes with moving objects," in *Computer Vision and Pattern Recognition, 1998. Proceedings. 1998 IEEE Computer Society Conference on*. IEEE, 1998, pp. 354–360.

[30] Yu-Chih Liu, Kai-Ying Lin, and Yong-Sheng Chen, "Birds-eye view vision system for vehicle surrounding monitoring," in *International Workshop on Robot Vision*. Springer, 2008, pp. 207–218.

[31] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[32] Demetrios V Papadimitriou and Tim J Dennis, "Epipolar line estimation and rectificaion for stereo image pairs," *IEEE Transactions on Intelligent Transportation Systems*, vol. 5, no. 4, pp. 672–676, 1996.