Small-form Spatially Augmented Reality on the Jetson TX1

Rahul Prabala
Stanford University
rprabala@stanford.edu

Abstract

This paper demonstrates a small-form projector-camera system built from the Intel RealSense SR300 Camera, the Texas Instruments DLP3000, and the NVIDIA Jetson TX1. The components are integrated into a Spatially Augmented Reality (SAR) system, and three separate demonstrations are shown to work. The system is compared to existing projector-camera techniques, and shown to be comparable in performance while taking advantage of advances in technology to reduce the required size of the entire system.

1. Introduction

Spatially Augmented Reality (SAR) provides an immersive way for users to interact with virtual objects in the real world, without the use of head mounted displays. Instead of conventional displays, the advent of DLP projectors has given rise to many different SAR applications. With the cost of projectors and computing devices decreasing as their capabilities increase, SAR devices can easily become as ubiquitous as cell phones. However, the high computational demand of SAR had previously set large requirements on the size of such systems. The need to perform image manipulation and transformation necessitates a computational system capable of quickly processing large amounts of data–typically reserved for GPUs on laptops or desktops.

In this paper, we construct a projector-camera system used for SAR with a small form factor without sacrificing performance. By creating an SAR-capable system, devices will not be limited to small screen sizes for mobile use. Rather, they can use any object in the world to render onto, vastly expanding the realm of possibilities for both users and developers alike.

2. Related Work

While Spatially Augmented Reality as a concept has been around for many years, Raskar *et al*. formalized its definition as rendering objects in a viewer's physical space rather than in their view [10]. Raskar also constructed a system capable of retexturing and rendering projections onto a known three-dimensionsal model [11]. Other work has been done on aligning multiple projectors that are otherwise unrelated, such as in [9, 8], in addition to constructing a portable projector-camera system.

To perform the calibration, several techniques have been proposed. Lee *et al*. add light detectors at the corner of a projection surace within the view frustrum, upon which a series of gray-coded patterns reveals the underlying quadrilateral [4]. Once the quadrilateral is known, applying homography techniques as in [12] will compute an accurate projection onto the surface. An integrated projector-camera system that performs this prewarping is also shown in [7].

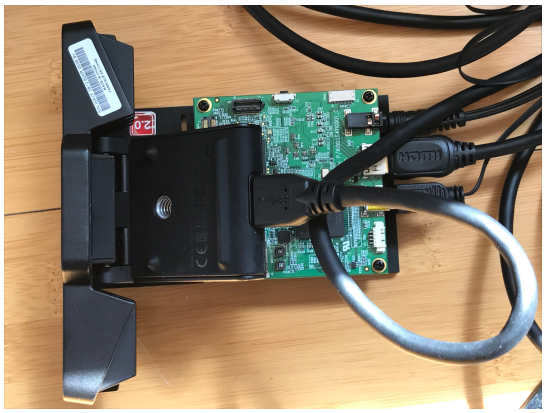
Modern projector-camera systems are capable of performing SAR in real-time, such as in [5]. An extensive overview of SAR algorithms and projection mapping, as well as optimized techniques for dealing with non-planar and specular surfaces can be found in [1, 2].

3. Hardware System

The projector-camera system is comprised of three components: the NVIDIA Jetson TX1, the Intel RealSense SR300, and the Texas Instruments DLP3000 EVM. The TX1 is used for its compute power, the SR300 as a camera, and the DLP3000 to project processed images back into the scene. The combined system can be seen in Figure 1.

3.1. NVIDIA Jetson TX1

Addressing the concerns laid out in the introduction, the Jetson TX1 is powerful enough to handle image processing, with a quad core ARM A57 and an onchip Maxwell GPU with 256 CUDA cores. This gives ample room for optimizations to improve the speed of the computations while still maintaining a low power and size profile at only 50*mm* by 87*mm*. The Jetson TX1 is shown in Figure 2.



(a) Top view of the capture and display system.

(b) Front view of the capture and display system.

Figure 1: Construction of the Hardware System.

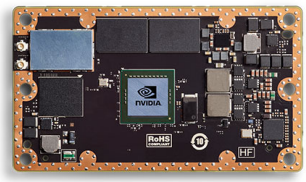


Figure 2: Jetson TX1

3.2. Intel RealSense SR300

The SR300 has three intrinsic capture streams: color (RGB), infrared (IR), and depth. By using structured IR patterns as opposed to stereo depth techniques employed in the RealSense R200, the camera is able to construct a depth map of the scene at an effective range of up to $2m$, with a resolution of $1/32\ mm$. Because the intrinsics and extrinsics are known for each stream with respect to each other, it is possible to deproject the depth map onto the color stream, constructing the depth map of what is imaged by the color sensor. The SR300 is shown in Figure 3.

Figure 3: RealSense SR300

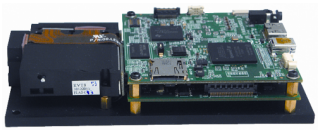


Figure 4: DLP3000

3.3. TI DLP3000

The DLP3000 is a MEMS device capable of applying patterns at very high frequencies. Though it supports higher resolutions, it has 608 x 684 diagonal micromirrors arranged in an addressable array. While commonly used to generate structured light patterns and for spatial light modulation, it lends itself quite well to SAR due to the ability to control each pixel individually. The DLP3000 is shown in Figure 4.

4. Capture and Display Pipeline

The capture and display pipeline describes the steps taken to generate and project an augmented image into the scene. An overview of the pipeline is presented in Figure 5.

4.1. Calibration

In order to align the camera and projector coordinates to avoid parallax between the captured images and displayed augmentation, we project a simple calibration pattern consisting of a rectangle with the bottom edge removed. The outer edges of the rectangle will represent the FOV of the projector. This allows for the conversion from camera coordinates to projector coordinates computationally. For this method to be successful at determining the FOV of the projector in camera coordinates, the rectangle cannot be obscured by objects. Hence, the bottom edge is removed to avoid false positives during the detection phase that skew the computed boundaries of the FOV. To remove artifacts from the scene interfering with the detection, two frames

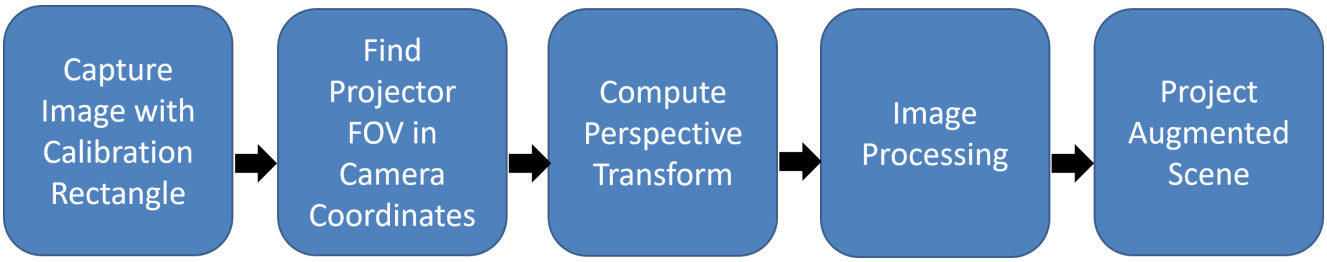


Figure 5: Flowchart of the Capture and Display Pipeline

are captured: one with the outline, and one without, so that their difference can be passed on to the next stage.

4.2. Rectangle Detection and FOV Computation

The problem of finding the projector FOV essentially reduces to finding the corners of the rectangle from the previous step. With the raw difference image from the prior stage of the pipeline, we convert the pixel values into HSV space to more easily distinguish false positives. For example, specular surfaces in the image will reflect the calibration pattern, but will have different saturation and value parameters, can be thresholded out. The pixel values that meet the criteria are kept, and the rest are masked out.

Following this, the image is converted into a binary image using a Gaussian adaptive threshold, and adjacent binary values are combined using dilation and erosion operators to form contiguous boundaries in the image. Using the methods proposed in [13], we then find the contours in the image and remove those attributed to noise using the perimeter as a metric. The remaining contours are assumed to be the boundaries of the rectangle. From the boundaries, we extract the four corners of the rectangle by finding the points in each quadrant of the image that are the furthest away from the center of the image. For more accuracy, we also implemented extrapolation from line segments approximated from the points in each contour. This method has the advantage of being more robust to occlusions of the calibration pattern, at the tradeoff of additional computation time. Pseudocode for this approach appears in Algorithm 1.

Algorithm 1 Compute Projector FOV

```
function COMPUTE_FOV(I)
    Pixels = ∅
    for i ∈ I do
        if InHSVRange(i) then
            Pixels ← i
    Pixels ← DILATEANDERODE(Pixels)
    BinaryImage ← ADAPTIVETHRESHOLD(Pixels)
    Contours ← FINDCONTOURS(BinaryImage)
    for c ∈ Contours do
        if PERIMETER(c) < SizeThreshold then
            Contours.remove(c)
    Quadrants ← SEGMENTPOINTS(Contours)
    Corners = ∅
    for all points ∈ Quadrants do
        maxDist = 0
        CornerInQuadrant = ∅
        for p ∈ points do
            if DIST(p) > maxDist then
                maxDist = DIST(p)
                CornerInQuadrant = p
        Corners ← CornerInQuadrant
    return Corners
```

4.3. Perspective Transformation

Once the four corners are known, the image can be transformed from the camera's coordinate space into the projector's coordinate space using the four points to form the basis of a homography. Using the four corners, a 3x3 homography matrix M is computed such that:

$$\begin{bmatrix} c_i u_i \\ c_i v_i \\ c_i \end{bmatrix} = M \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \tag{1}$$

where (u_i, v_i) is in the projector's coordinate space, and (x_i, y_i) is in the camera's coordinate space, and c_i is a constant scale factor. The index i represents the four points that are used in the computation. Once M is computed, then it is applied to the source image to compute pixels in the transformed image as:

$$(u, v) = \left(\frac{M_{11}x + M_{12}y + M_{13}}{M_{31}x + M_{32}y + M_{33}}, \frac{M_{21}x + M_{22}y + M_{23}}{M_{31}x + M_{32}y + M_{33}} \right) \tag{2}$$

Once this transformation is applied to the image, and we have the image in projector coordinates, we can then proceed to the next stage in the pipeline.

4.4. Image Processing

With the image in the proper coordinate space, any processing applied to the image will be rendered in the scene with perfect fidelity as we can control each pixel in the DLP3000. We demonstrate three spatially augmented reality applications.

4.4.1 Edge Projection

The edges present in the scene are detected, highlighted, and projected back into the scene. Prior to thresholding the image, the image must be denoised so as to not incur any false positives. The interested reader is directed to [14], [3] for more information on denoising techniques applied here. After denoising, the images are thresholded using an adaptive Gaussian. Any pixel remaining is determined to be an edge of interest in the scene, and is highlighted.

4.4.2 Scene Geometry

Using the depth map stream from the SR300, we can obtain a depth image in addition to the color image. Since the camera intrinsics and extrinsics are known with relation to each other, we can project the depth stream into the color stream's coordinates, obtaining a depth map of the scene as perceived by the color stream. We can then colorize the depth map to showcase the geometry of the scene by projecting different colors to objects at a different depth.

4.4.3 Privacy Zone

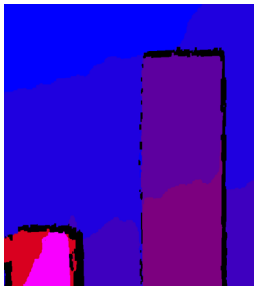
As presented in [6], the Privacy Zone defines a region of three-dimensional space in which objects are not "allowed" to be. In the example demonstrated, objects on the left side of the image plane, and within 0.4m of the sensor are colored red to indicate that they are not allowed to be inside that region.

4.5. Display

After the image has been processed, it is written out to the DLP3000 to be projected back into the scene. Since each pixel corresponds to a mirror on the DLP, the resulting projection requires no further processing.

5. Experimental Results

In this section, we present our results, as well as the demonstrations and applications used to showcase the performance of the system. An example of the pipeline demonstrated using the Edge Projection application in the previous section can be seen in Figure 6. The scene geometry application can be seen in Figure 7. The privacy zone application can be seen in Figure 8.



(a) Scene Geometry in projector FOV coordinates

(b) Projected scene geometry.

Figure 7: Scene Geometry

In terms of performance, as implemented, the Jetson-based system has a theoretical maximum of 20 frames per second, taking full advantage of the 60fps capture speeds on both the color and depth streams of the SR300, as well as the HDMI port on the DLP3000. Two frames are required to capture the rectangle, and approximately one frame's worth of time will be used for the computation. However, the demonstrated speed of the system is roughly 5.5fps. This is due to only using the CPU of the TX1, and additional speedups can be gained from implementing the image processing and rectangle thresholding on the GPU. Additionally, the image download to the DLP was done over USB instead of HDMI in order to facilitate the per-pixel control. The primary source of error in the projection is the disparity of corner selection for the calibration rectangle: on average, 2-3 pixels. Lastly, the form factor of the entire system is quite small, as desired: 120 *mm* x 117 *mm* x 40 *mm*.

6. Analysis

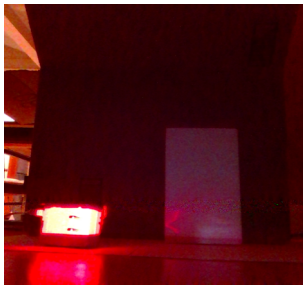
We present an analysis and discussion of the findings in this paper in comparison to other work in the field, as well as limitations of the current implementation. We conclude this section with further direction and improvements to be made.

6.1. Discussion of results

While the difficulty of generating augmented scenes with visual accuracy is high, the system we constructed

Figure 6: Example pipeline stage results, using Edge Projection to demonstrate the processing step.

(a) Privacy Zone in projector FOV coordinates



(b) Projected image, with object in ”forbidden” zone.

(c) Projected image, with object moved further back.

Figure 8: Privacy Zone

is quite capable of handling this, as demonstrated by the low disparity value for the computed rectangle. The Jetson is more than capable of handling the image processing required to allow for interactivity, though the CPU cannot keep up with real-time. This was surprising, as the algorithms selected, while certainly computation heavy, are not overwhelming, and can be performed on desktop and laptop machines with ease. However, given that the Jetson is optimized for its size and power consumption, it is better suited for mobile applications than standard computation systems.

Each of the applications show a surprising degree of accuracy and effectiveness. For example, the scene geometry augmentation correctly identifies that the plug is rotated out of the plane of the projector, and the edge projection augmentation highlights the prong that is in the FOV of the projector. While not quite real-time, the capture and display pipeline does identify the features in the scene quickly enough to produce several frames. As mentioned above, the speed can be improved by parallelizing many of the operations and implementing them on the GPU.

6.2. Comparison to Related Work

Prior work in this area has been fruitful, yielding several implementations of spatially augmented reality camera-projector systems. For more details, see the related work section.

Real-time projection and spatially augmented reality has been implemented, such as in [5], so while 5.5 frames per second is interactive, it is not quite at the level of state of the art. However, in terms of sizing, the approach laid out there requires the user to wear a computer in order to perform the processing required. This is similar to [8], in which an additional computer is necessary coupled to the projector, measuring 177 mm x 127mm x 38.1mm (not including the size of the computer or camera). We also demonstrate SAR without preimaging of the model as required in [11].

In terms of visual accuracy, our 2-3 pixels of disparity falls well within the 2% guidline laid out in [1], as well as matching work done in [9] at .3-2 pixels of disparity. However, again, we are able to achieve this level of accuracy while still constraining the size of the system.

6.3. Limitations and Future Work

The primary limitation in the system is the calibration pattern and pipeline stage. Because the pattern is projected using visible light, being able to differentiate the pattern from its surroundings is very scene and environment dependent. Depending on the amount of ambient light, it becomes easier to misclassify the calibration pattern's pixels. In addition, specularity in the scene requires further tuning of the HSV parameters in order to classify the pixel as part of the pattern. To combat some degree of scene dependence, the calibration pattern can be projected using light from the IR spectrum instead of the visible spectrum. This has two advantages: it removes the calibration pattern from the user's view, and removes some of the scene dependence on the calibration process, though it does not completely remove it.

Currently, all of the algorithms are implemented on the CPU. This underutilization can be overcome by using the on board GPU of the Jetson. GPU implementations as well as coding to project using specularity and other surface constraints can be found in [1]. Additionally, the DLP is also not being utilized to the fullest. Given that it is capable of displaying several binary patterns in rapid succession, it would be much faster to add in optical techniques to calibrate the projector such as in [4]. This would eliminate the extra time taken to download the calibration patterns. Alternatively, storing the calibration patterns onboard the DLP in its framebuffer would achieve a similar effect.

7. Conclusion

In this paper, we have demonstrated that spatially augmented reality can be realized with a very small volumetric footprint. Utilizing this small form as an example, we will soon see cell phones with the capability to employ SAR. Reduction in size without sacrificing performance paves the way for truly ubiquitous SAR, allowing the real world to become a canvas for digital exploration.

8. Acknowledgements

The author would like to thank Matt O'Toole for his invaluable input and guidance in shaping this project.

References

[1] O. Bimber, D. Iwai, G. Wetzstein, and A. Grundhöfer. The visual computing of projector-camera systems. In *Computer Graphics Forum*, volume 27, pages 2219–2245. Wiley Online Library, 2008.

[2] O. Bimber and R. Raskar. Modern approaches to augmented reality. In *ACM SIGGRAPH 2006 Courses*, page 1. ACM, 2006.

[3] A. Buades, B. Coll, and J.-M. Morel. Non-local means denoising. *Image Processing On Line*, 1:208–212, 2011.

[4] J. C. Lee, P. H. Dietz, D. Maynes-Aminzade, R. Raskar, and S. E. Hudson. Automatic projector calibration with embedded light sensors. In *Proceedings of the 17th annual ACM symposium on User interface software and technology*, pages 123–126. ACM, 2004.

[5] P. Mistry and P. Maes. Sixthsense: a wearable gestural interface. In *ACM SIGGRAPH ASIA 2009 Sketches*, page 11. ACM, 2009.

[6] M. O'Toole, R. Raskar, and K. N. Kutulakos. Primal-dual coding to probe light transport. *ACM Trans. Graph.*, 31(4):39–1, 2012.

[7] R. Raskar and P. Beardsley. A self-correcting projector. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 2, pages II–II. IEEE, 2001.

[8] R. Raskar, P. Beardsley, J. Van Baar, Y. Wang, P. Dietz, J. Lee, D. Leigh, and T. Willwacher. Rfig lamps: interacting with a self-describing world via photosensing wireless tags and projectors. In *ACM Transactions on Graphics (TOG)*, volume 23, pages 406–415. ACM, 2004.

[9] R. Raskar, J. Van Baar, P. Beardsley, T. Willwacher, S. Rao, and C. Forlines. ilamps: geometrically aware and self-configuring projectors. In *ACM SIGGRAPH 2006 Courses*, page 7. ACM, 2006.

[10] R. Raskar, G. Welch, and H. Fuchs. Spatially augmented reality. In *First IEEE Workshop on Augmented Reality (IWAR98)*, pages 11–20. Citeseer, 1998.

[11] R. Raskar, G. Welch, K.-L. Low, and D. Bandyopadhyay. Shader lamps: Animating real objects with image-based illumination. In *Rendering Techniques 2001*, pages 89–102. Springer, 2001.

[12] R. Sukthankar, R. G. Stockton, and M. D. Mullin. Smarter presentations: Exploiting homography in camera-projector systems. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 1, pages 247–253. IEEE, 2001.

[13] S. Suzuki et al. Topological structural analysis of digitized binary images by border following. *Computer vision, graphics, and image processing*, 30(1):32–46, 1985.

[14] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *Computer Vision, 1998. Sixth International Conference on*, pages 839–846. IEEE, 1998.