https://doi.org/10.2352/ISSN.2470-1173.2019.3.SDA-660

Development of a Camera-Based Projection Mapping System for Non-Flat Surfaces

Daniel Adams, Tri Tai Steven Pham, Kale Watts, Subhash Ramakrishnan, Emily Ackland, Ham Tran Ly, Joshua Hollick, Andrew Woods; Curtin University; Perth, Western Australia, Australia.

Abstract

While most conventional projection surfaces are flat, there are many situations where the surfaces are uniquely shaped – such as the Cylinder display at the Curtin University HIVE visualisation facility. This paper describes a student project to develop a system which will automatically generate a set of warp-blend meshes for multiple projectors so that the projected output will have correct display geometry across the full projection surface. The HIVE Cylinder display consists of a 180° half-cylinder display surface with 8 m diameter and 3 m height. Three projectors illuminate the display surface and the display is stereoscopic capable using frame-sequential 3D and LC shutter glasses. The software works by sending a series of calibration test patterns to the projectors, which are photographed by a camera, processed to calculate a projection map, output as a set of warp-blend meshes, which will result in a continuous screen surface that appears to have correct geometry to an observer at a predefined location. The project has been undertaken by a group of students working on several modular components of the overall system: Camera to Computer Interface, Image Generator, Image Processor, Geometry Computation, and Warp-Blend Output.

Keywords

Projection Mapping, Image Processing, Screen Distortion Correction, Cylindrical Displays.

Introduction

This paper describes the development of a camera-based projection mapping system for non-flat surfaces, primarily intended for use on the Cylinder display at the HIVE visualisation facility at Curtin University (Perth, Western Australia) [1][2].

In most applications, the projection surface used for image and video projection purposes is a simple flat surface, which is easily corrected to produce a geometrically 'square' display using simple keystone correction controls often built into the projector. However, there are many instances where the projection surface is not flat and may be curved in one or two dimensions and when an uncorrected image is projected onto such a display the image can be significantly distorted such that straight lines are shown curved, vertical lines are not vertical, and horizontal lines are not vertical.

The Cylinder display at the Curtin HIVE visualisation facility is one example of a non-flat projection surface and is the system used for this project. The HIVE Cylinder display consists of a 180° half-cylinder display surface with 8m diameter and 3m height. The display surface consists of a lace and grommet style stretched polymer screen strung from a steel frame from Stewart Filmscreen. The image is projected onto the screen using three ProjectionDesign WUXGA DLP 3D projectors fitted with wide-angle lenses and mounted from the ceiling. The stretched polymer screen surface bows in the middle so that the surface is not uniformly cylindrical. Fig. 1 shows a laser scan of the projection screen from the left-hand side which illustrates the cylindrical curvature and the bowing in the centre of the screen towards the centre. The combination of several factors means that the projected image on the screen does not produce a geometrically uniform image: (a) the cylindrically curved screen, (b) the central bowing in the screen, (c) the use of wide-angle lenses, and (d) the off-centre projection angle from high above the screen. These factors all contribute towards a complicated geometric distortion of the image output by the projectors.

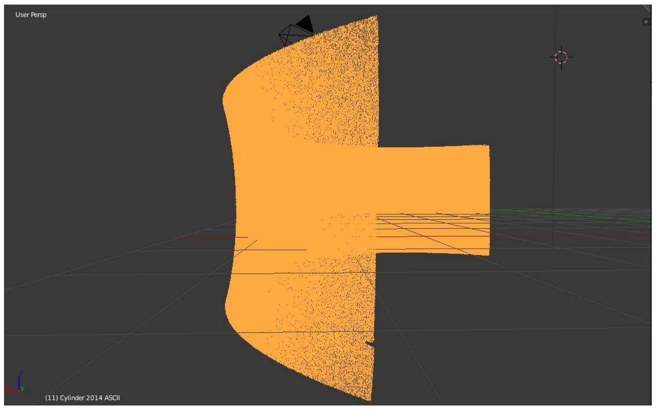


Figure 1. *A laser scan of the HIVE Cylinder projection screen. [18]*

The projectors are fitted with warp-blend units which allow the image to be warped to apply a geometric correction to allow the resultant image on screen to be shown geometrically correct. Blending refers to the process of adjusting the intensity of the image in areas where two projectors overlap so as to produce a continuous single image [3]. The geometric correction can be entered manually point-by-point using a theodolite however it is extremely time consuming. There are also some software solutions available that use a camera to photograph the display and software to calculate the geometric correction, however we found these solutions did not correct the central bowing of our screen. This provided the motivation for this project.

An automated camera-based software solution would be much faster and more accurate than a human-in-the-loop manual system. Calibrations could also be performed more regularly (since they are quick to do) and even potentially done in real-time.

In the context of this project, the term projection mapping is used to refer to the process of mapping of geometrically corrected images onto a non-uniform projection surface so that a displayed image looks correct to an observer.

This project aims to provide a robust implementation where the image deformation is taken into consideration, resulting in a more accurate calibration of the display. It also intends to produce several quality of life improvements such as: allowing for the user to move the viewing position in software to a new predefined location, being able to save and revert to different calibrations and configurations, to provide meaningful error logs for debugging, and the extensibility to work with other possible display configurations.

Methodology

The project was divided into several distinct modules so that the software could be developed in several distinct stages and additionally so that the work could be allocated between the different project participants.

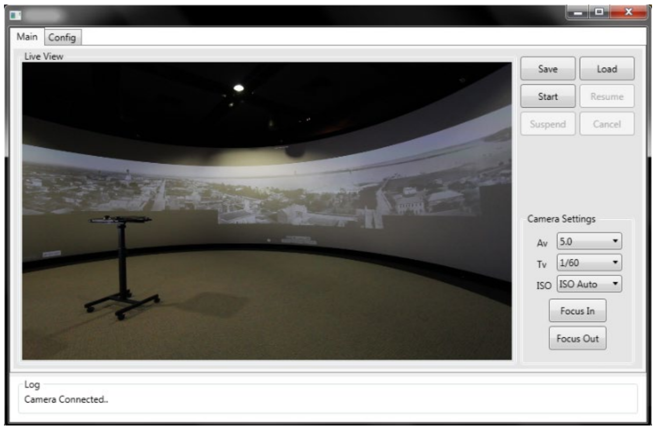
The modules are:

- Camera to Computer Interface
- Image Generator
- Image Processor
- Geometry Computation
- Warp-Blend Output

The 'Camera to Computer Interface' module is responsible for interacting with the digital still camera, capturing and downloading images. This stage should present a consistent interface that is suitable for use with a DSLR camera such as the Canon EOS 1200D. The 'Image Generator' module must generate and display a series of test patterns on the screen, for camera capture. The 'Image Processor' module should be able to take an image from the camera and detect the relevant features that are required for the screen distortion correction. These features would likely be a combination of natural features (edge of screen) and projected features (test pattern corners/control points). The 'Geometry Computation' engine should use these features in the image to calculate the final display geometry. The 'Warp-Blend Output' module should be capable of writing output files for the ProjectionDesign/Barco WB2560 warp-blend units.

Camera to Computer Interface

The Camera to Computer Interface module is responsible for remotely controlling the aperture, shutter speed, ISO, and Focus settings of the Canon EOS 1200D camera, and allowing the user to view a live feed of what the camera sees within a Graphical User Interface (GUI) window (see Fig. 2) of the software. The module utilises the Canon Digital SLR Camera Software Developers Kit [4][5][6]. It must also provide a framework of automating the test image capture sequence, whereby the camera takes a photo without intervention from the user and saves it to file.



Av 5.0
Tv 1/60
ISO ISO Auto

```
Camera Connected..
```

__Figure 2.__ Camera live view as seen in the GUI.

On start-up of the software the Canon SDK is initialised and a session is opened with a connected camera. All the necessary event handlers and callback functions are assigned their appropriate commands, which are issued based on their respective event listeners. When the camera is no longer in use (i.e. when the user closes the application), the camera session is also closed and the SDK is terminated.

Image Generator

The Image Generator module is responsible for generating a sequence of test patterns to be sent through each projector and displayed on the screen. The module should also be able to blackout all other projectors not in use when a single test pattern is required to only be displayed through a single projector.

When the user clicks the "Start" calibration button in the GUI, the Image Generator module loads the blank white screen test pattern into memory and displays it on all projectors. Control is given to the Camera to Computer Interface to capture an image of the test pattern on the screen (see Fig. 3).

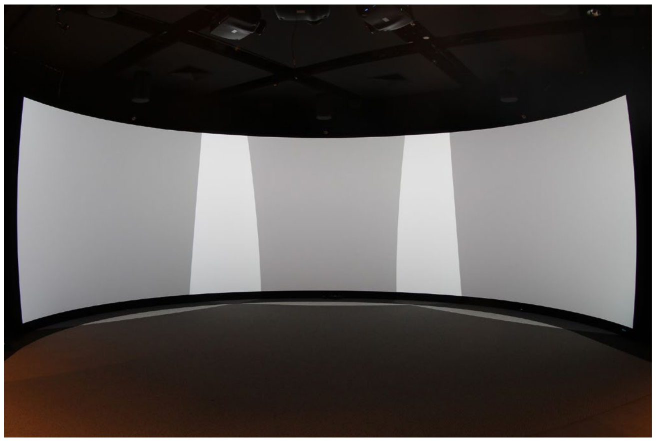


__Figure 3.__ Blank white screen test pattern output by all three projectors onto the HIVE Cylinder screen.

Once the Image Processor module verifies that the whole screen is in full view of the camera, the Image Generator module must begin the sequence of projecting the rest of the test patterns to each individual projector, again sequentially giving control to the Camera to Computer Interface to capture and save the images.

A series of ten test pattern combinations are sent to the three projectors. Firstly a full-white test pattern is sent to all projectors. Next each projector is sent three individual test patterns (full-white, 2x2 boxed test pattern, and 48x30 boxed test pattern) while the other projectors are set up black. See Fig. 4, Fig. 5 and Fig. 6 as examples of the test patterns on screen. The full-white test pattern on all projectors allows the bounds of the projection screen to be determined. The full-white test pattern on each individual projector allows the bounds of each projector to be determined. The 2x2 box test pattern allows the centre pixel of each projector to be determined. The 48x30 boxed test pattern allows the geometry of the projected image to be determined.

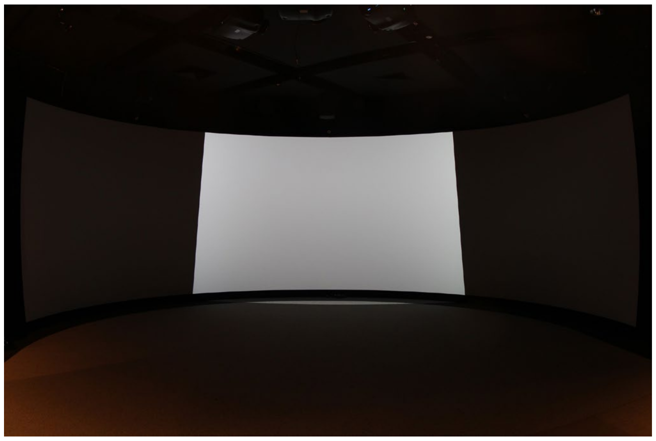


Figure 4. Blank screen test pattern on projector 2.

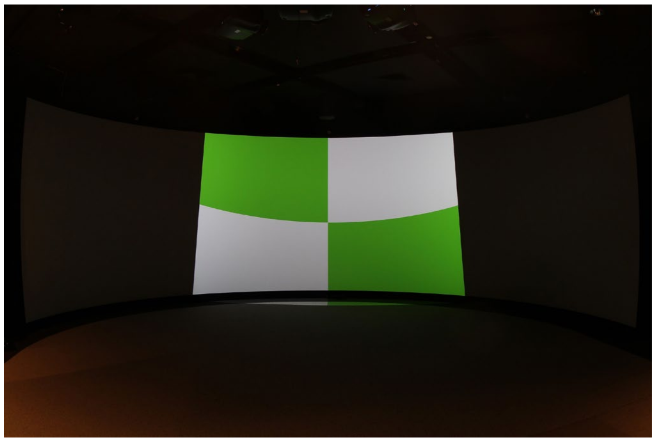


Figure 5. 2x2 boxed test pattern on projector 2.

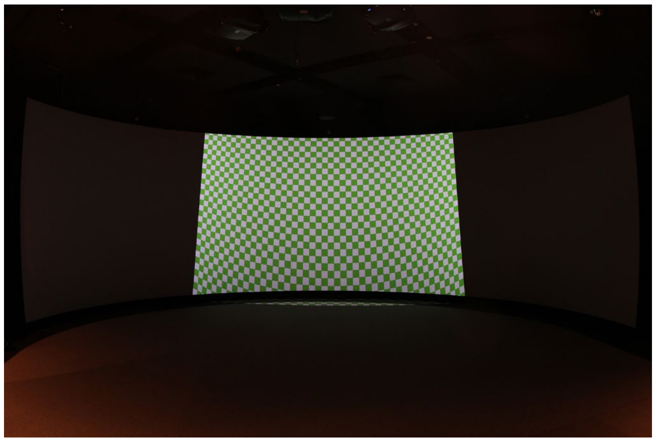


Figure 6. 48x30 boxed test pattern on projector 2.

Image Processor

The Image Processor module utilises the OpenCV: Image Processing library [7][8], and is responsible for: confirming that a valid test image has been captured (i.e. that the full screen is completely in the frame of the camera), deriving the intrinsic and extrinsic properties of the camera via pose estimation, and performing feature extraction on the captured test images. With this information it must scan the detected features from an unstructured list of coordinates into an organised data structure, map these points to their corresponding actual projected coordinates, and finally iteratively interpolate between each detected point until the maximum error when querying the data structure is no more than 2.5 pixels (see Fig. 7). It must also enable the user to manually adjust the screen dimensions by an amount predefined in a GUI configuration window.

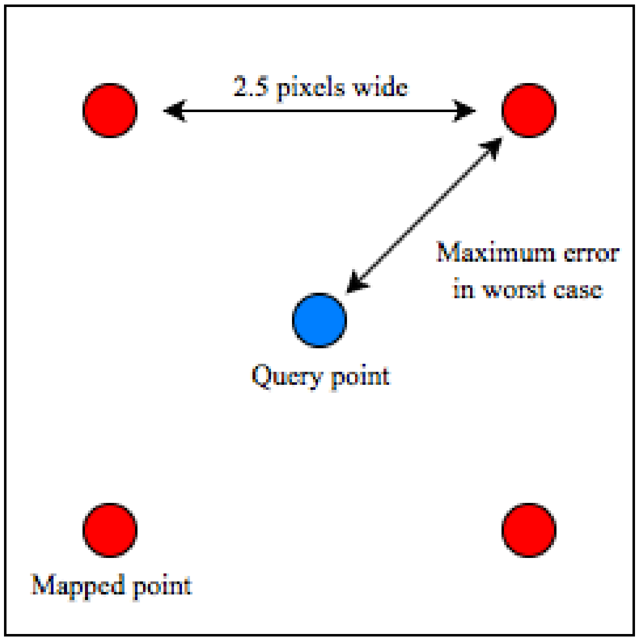


Figure 7. Quantification of the maximum query error in the worst case.

When a captured image is first loaded into memory by the Image Processor Module it undergoes an initial stage of pre-processing before any feature detection is performed. This includes a conversion to monochrome and binarisation to more clearly differentiate between the boxes in the test pattern, then finally noise reduction. Next the edges of the whole screen are detected and a mask is generated to constrain the feature extraction space (see Fig. 8).

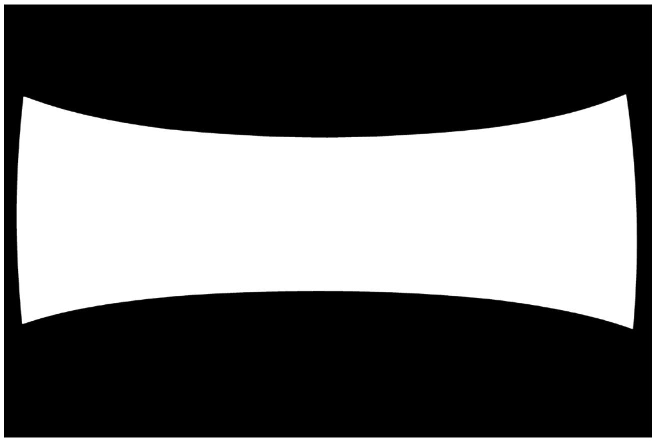


Figure 8. Full screen mask.

This mask is applied to the image and a new mask of an individual projector is generated constraining the feature extraction space even further (see Fig. 9).

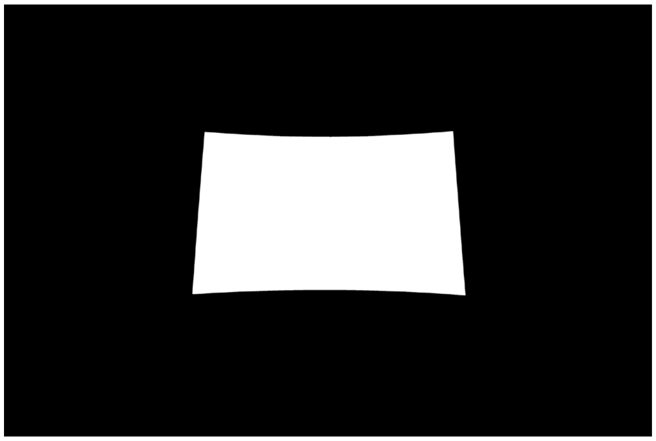


Figure 9. Projector 2 mask.

The corner points of the full-screen mask are then detected and saved in a data structure to be used for pose estimation. The module then extracts the centre of projection from the 2x2 box test capture (see Fig. 10) and a series of control points on the 48x30 box test capture (see Fig. 11).

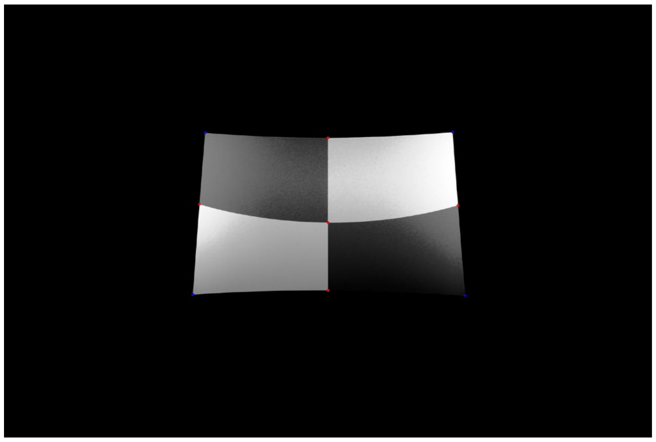


Figure 10. Detected points (red dots) on the 2x2 boxed test pattern capture.

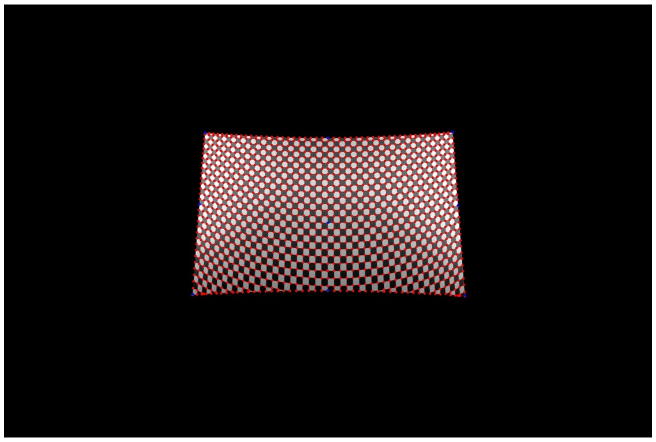


Figure 11. Detected points (red dots) on the 48x30 test pattern capture.

A 'middle out' algorithm was designed to scan the detected features from the unstructured list of coordinates into an organised data structure of control points, from the only 'concrete' point known being the centre of projection (see Fig. 12 and Fig. 13).

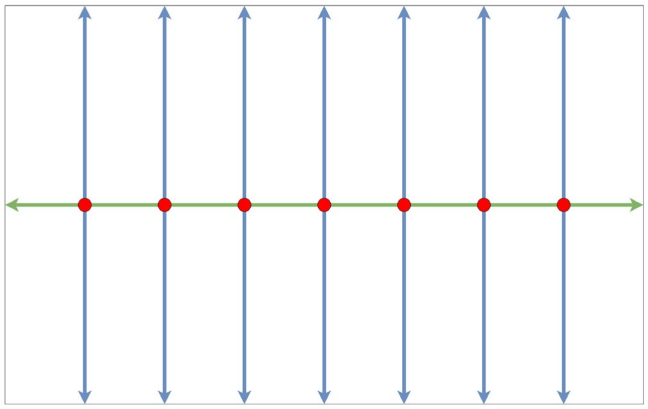


Figure 12. Middle out scan, primary and secondary scan being the green and blue arrows respectively.

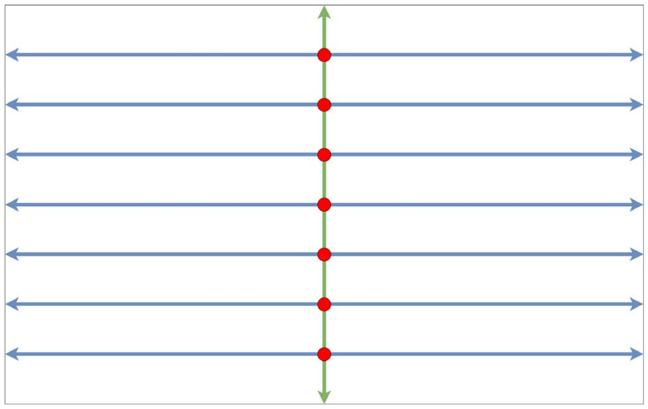


Figure 13. Middle out scan transposed, primary and secondary scan being the green and blue arrows respectively.

The reason why two different directions are required to scan the unstructured list of detected features (see Fig. 11) into an organised data structure, is because there are limitations to the points that can be reached by the scan in a single direction. This problem is magnified in the edge cases where the test pattern being projected onto the screen contains non-uniform cutoffs (see Fig. 14 and Fig. 15).

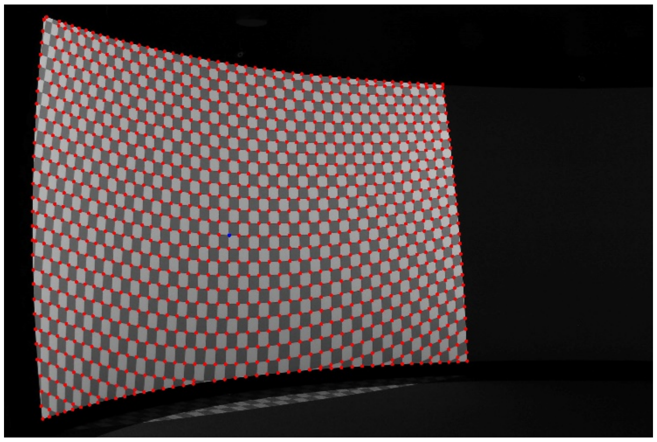


Figure 14. Scanned points (red dots) on the 48x30 test pattern capture using middle out scan from Fig. 12.

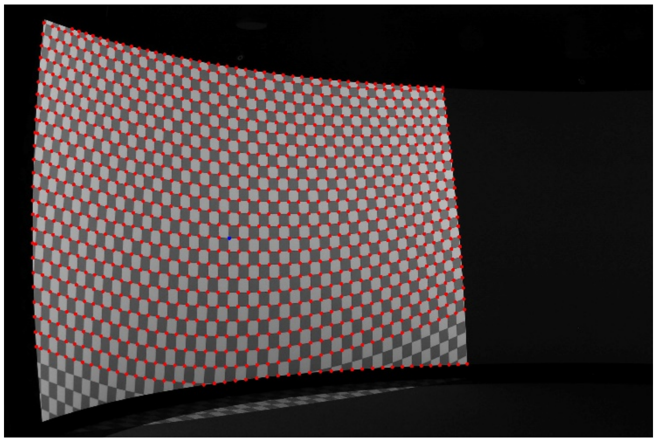


Figure 15. Scanned points (red dots) on the 48x30 test pattern capture using middle out scan from Fig. 13

Essentially what needs to be achieved, is to take the unordered one-dimensional list of feature points and place them into a 2D array, of which each point therein is correctly located in relation to all other feature points. This format should mirror how points appear in relation to each other as seen on the original test pattern. Combining the results of the middle out algorithm from two directions provides a robust method to follow the lines of the test pattern boxes achieving an accurate relation mapping.

The algorithm works by searching for the nearest neighbour at an expected distance and angle from the most recently mapped 'concrete' control point, then appending this new point to a table data structure of 'concrete' control points (see Fig. 16).

This process continues in the above-mentioned middle out fashion until all unstructured coordinates have been mapped to a corresponding 'concrete' control point. Once this table data structure of 'concrete' control points has been created we again use the centre of projection and the expected distance between points to calculate their corresponding flat ideal Projected Coordinates (see Fig. 17). Finally to increase the density of control points we subdivide the table and perform point interpolation, effectively doubling the density of the table (see Fig. 18). This process is repeated until the ideal distance between the points is reduced from the original 40 pixels as seen on the original test pattern, down to 2.5 pixels.

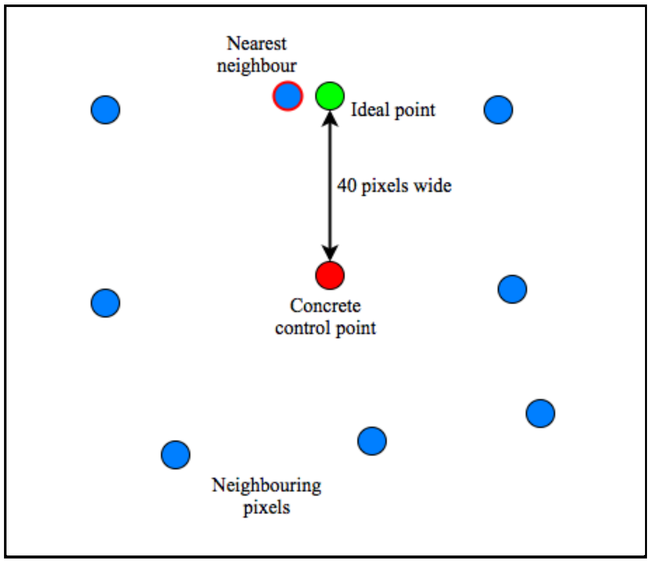


Figure 16. Nearest neighbour search.

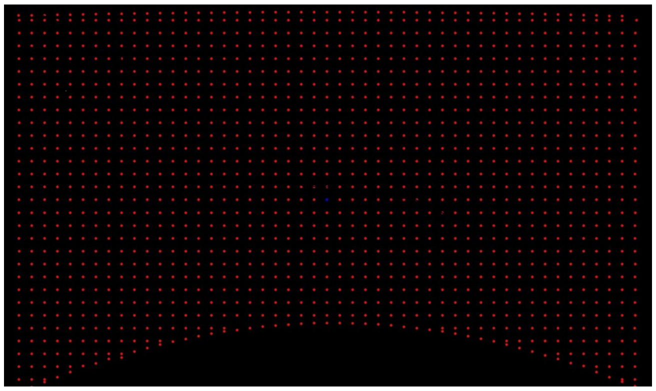


Figure 17. Control points mapped to flat ideal Projected Coordinates.

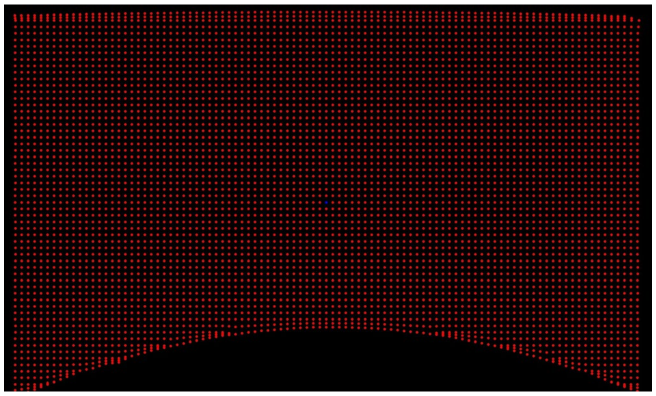


Figure 18. Projected Coordinates after first round of subdivision and interpolation.

Geometry Computation

The Geometry Computation module is responsible for using the control points and extracted features from the Image Processor module to calculate the correct display geometry and to generate a warp mesh for the screen. This is achieved through the use of projective geometry and by employing the concept of the pinhole camera model [9]. It must also enable the user to manually adjust the viewing position that the display is calibrated around by an amount predefined in a GUI configuration window.

Upon successful completion of all image processing, the system begins the process of geometry computation and screen calibration. For this section we now need to define and differentiate between the multiple coordinate systems for the multiple screen spaces used throughout the project:

- **Ideal Virtual Coordinates:** Coordinates that exist on an ideal virtual screen space, representative of where the detected coordinates are expected to be.
- **3D Object Coordinates:** Coordinates found on the laser scan of the physical screen, with the origin being the location of the laser scanning device.
- **2D Image Coordinates:** Coordinates found on the test captures, with the origin being the top left corner of the image.
- **Projected Coordinates:** Coordinates that have been mapped from the unstructured 2D Image Coordinate space, to a space representative of where the detected coordinates actually are in respect to the Ideal Virtual Coordinates.

The Geometry Computation module works by projecting a point we refer to as the Ideal Virtual Coordinate through an imaginary plane which is then wrapped around the physical screen represented as a laser scan (see Fig. 19). This is achieved through the use of ray-casting [10]. In order to locate where the light ray intersects with the laser scan, all rays must be projected from a consistent origin, this being the centre of the laser scan. The spherical coordinate system [11] along with the Armadillo library [12][13] were used to determine a necessary angle at an Ideal Virtual Coordinate, finally generating the 3D Object Coordinate representing its nearest neighbour on the laser scan. The end result is a 2D to 3D projection of an ideal point onto the laser scan.

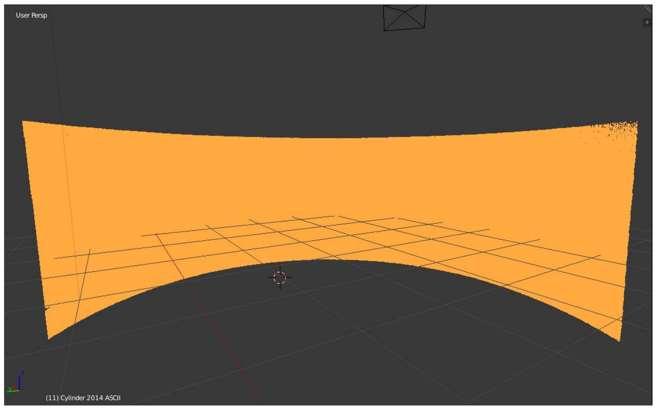


Figure 19. *Laser scan 3D Object Coordinates.*

Using the intrinsic and extrinsic properties of the camera derived from the pose estimation calculations in the Image Processor module, the corner points of the 3D laser scan is aligned with the corresponding corner points of the full screen test image.

We then perform a 3D to 2D projection on the newly generated Object Coordinates to find the relative 2D Image Coordinates as detected by the Image Processor module. This is done by querying a table of 2D Image Coordinates provided by the Image Processor module with the use of a greedy search which minimises the Euclidean distance between the 3D to 2D projected point and the actual 2D Image Coordinate. Once located, we retrieve the Projected Coordinate of the corresponding 2D Image Coordinate, completing the process of mapping it to its own flat ideal space.

The final geometry data structured as a warp mesh consisting of Ideal Coordinates and actual Projected Coordinates is then written to an intermediary output file along with calculated projector overlaps, which will be read in by the Warp-Blend Output module.

Warp-Blend Output

The core purpose of the Warp-Blend Output module is to interface with the ProjectionDesign/Barco WB2560 warp-blend hardware [14] in order to send and apply the generated warp mesh to the projector output. It must also apply an illumination and transparency blend to the overlapping regions of the projectors, and provide a framework that enables the user to save a calibration, avoid overriding the current calibration, and be able to revert to a previously used calibration. The LIDSSH [15], XSD [16], Boost [17] and Qt [18] libraries were utilised to support and optimize this functionality.

On start-up of the software the system loads all required plugins and devices from a configuration file for the Warp-Blend Output module. The user will enter all device preferences in a GUI configuration window and then click “Start Calibration”. The Warp-Blend devices are configured by assigning them each to a screen, plugin, name, and serial number by specifying relevant network information, and are finally initialised. When the “Start” calibration button is clicked all devices are put into bypass mode (no distortion correction or illumination blending) to allow for correct feature detection. Upon successful completion of the Geometry Computation module and screen calibration, the system exits bypass mode and loads the intermediary output file from the Geometry Computation module. The Warp-Blend Output module sends and applies the generated warp mesh to the projector output, along with an illumination and transparency blend to the overlapping regions of the projectors. The system allows the user to preview the resulting calibration and prompts them to either apply it, or revert back to the previous calibration.

Conclusion

This paper has provided an overview of a Curtin University Capstone Computing Project undergone by six undergraduate Computing students for the Curtin HIVE, over the course of 2018. The project has been a great learning experience for all students and the HIVE staff.

The project is in the final stages of becoming a functional prototype, with each module integrated with the system. The results of the final output calibration are currently being debugged and refined. Future plans for the software will be centred on increasing the accuracy of the calibration, reducing the time it takes to complete a calibration, and general usability improvements of the GUI.

Acknowledgements

The authors would like to thank Curtin University Computer Science Course Coordinator Dr. Hannes Herrmann for his support and supervision of this project throughout the course of 2018.

References

[1] A. Woods, S. Datta, J. Hollick, and P. Bourke, “The design, install and operation of a multi-user, multi-display visualisation facility” (in preparation).

[2] A. Woods, "Curtin HIVE - Hub for Immersive Visualisation and eResearch" Stereoscopic Displays and Applications XXVII conference, Electronic Imaging Symposium, San Francisco, California, February 2016. [Presentation Only] Video available online: https://youtu.be/iSLESjFZuXg

[3] R. Raskar, G. Welch, and H. Fuchs, "Seamless Projection Overlaps Using Image Warping and Intensity Blending" in Fourth International Conference on Virtual Systems and Multimedia, Gifu, Japan, November 1998.

[4] Canon, 'Canon Digital SLR Camera Software Developers Kit', 2018. [Online]. Available: https://www.canon.com.au/support/support-news/support-news/digital-slr-camera-software-developers-kit [Accessed 6 June 2018].

[5] Canon, 'Canon EDSDK Tutorial in C#', 2018. [Online]. Available: https://www.codeproject.com/Articles/688276/Canon-EDSDK-Tutorial-in-Csharp [Accessed 6 June 2018].

[6] Canon, 'EDSDKAPI (EOS Digital Software Development Kit Application Programming Interface', 2018. [Online] Available: https://www.fotostein.at/Software/EDSDKAPI [Accessed 6 June 2018].

[7] OpenCV.org, 'OpenCV 3.4.11', 2018. [Online]. Available: https://opencv.org/ [Accessed 27 April 2018].

[8] OpenCV.org, 'OpenCV Tutorials', 2018. [Online]. Available: https://docs.opencv.org/2.4/doc/tutorials/tutorials.html [Accessed 27 April 2018].

[9] D. A. Forsyth, and J. Ponce. *Computer Vision: A Modern Approach*, 2nd ed. Upper Saddle River, New Jersey: Pearson Education, 2012.

[10] T. Möller, E. Haines, and N. Hoffman, *Real-time rendering*, 3rd ed. Boca Raton, Florida: Taylor & Francis Group, 2008.

[11] Casio Computer Company Ltd, 'Spherical to Cartesian coordinates Calculator' 2018. [Online] Available: https://keisan.casio.com/exec/system/1359534351 [Accessed 26 October 2018].

[12] C. Sanderson and R. Curtin, 'Armadillo: C++ library for linear algebra & scientific computing', 2019. [Online]. Available: http://arma.sourceforge.net/ [Accessed 1 February 2019].

[13] C. Sanderson and R. Curtin, 'Armadillo: a template-based C++ library for linear algebra' *Journal of Open Source Software*, Vol. 1, pp. 26, 2016.

[14] Barco, 'WB 1920 & 2560 Barco-series User Guide', Barco, Kortrijk, Belgium, April 2014. Available: https://www.barco.com/en/support/docs/601-0318?revision=01

[15] A. Adamantiadis and A. Schneider, "LIBSSH - The SSH Library', 2018. [Online]. Available: https://www.libssh.org/ [Accessed 1 February 2019].

[16] Code Synthesis, 'XSD - XML Schema to C++ Data Binding Compiler', 2018. [Online]. Available: https://www.codesynthesis.com/projects/xsd/ [Accessed 1 February 2019].

[17] B. Dawes and D. Abrahams, 'Boost C++ Libraries', 2018. [Online] Available: https://www.boost.org/ [Accessed 1 February 2019].

[18] Qt Group, 'Qt Cross-platform software development for embedded and desktop', 2018. [Online] Available: https://www.qt.io/ [Accessed 1 February 2019].

[19] Blender Foundation, 'The Blender project - Free and Open 3D Creation Software', 2018. [Online] Available: https://www.blender.org/ [Accessed 16 October 2018].

Author Biographies

Daniel Adams is an undergraduate student at Curtin University, studying a Bachelor of Science degree in Computing, Majoring in Computer Science.

Tri Tai Steven Pham is an undergraduate student at Curtin University, studying a Bachelor of Science degree in Computing, Majoring in Computer Science.

Kale Watts is an undergraduate student at Curtin University, studying a Bachelor of Science degree in Computing, Majoring in Computer Science.

Subhash Ramakrishnan is an undergraduate student at Curtin University, studying a Bachelor of Science degree in Computing, Majoring in Cyber Security.

Emily Ackland is an undergraduate student at Curtin University, studying a Bachelor of Science degree in Computing, Majoring in Computer Science.

Ham Tran Ly is an undergraduate student at Curtin University, studying a Bachelor of Science degree in Computing, Majoring in Software Engineering.

Joshua Hollick is a Visualisation Technology Specialist at the Curtin HIVE, and a PhD Candidate in the Department of Spatial Sciences, Curtin University. Joshua's background is in electronic engineering, computer science and mathematics. His research interests include photogrammetry, sensor fusion and noisy data processing. He has worked on several projects involving underwater applications of photogrammetry, data visualisation and heritage mapping such as the imaging expedition and data processing of the HMAS Sydney (II) and HSK Kormoran.

Dr Andrew Woods is manager of the Curtin HIVE visualisation facility and a senior research fellow with the Centre for Marine Science and Technology at Curtin University. He has expertise in imaging and visualisation, underwater cameras, photogrammetric 3D reconstruction and underwater vehicles with applications in oil and gas, and maritime archaeology. He has Bachelor, Masters and PhD degrees in electronic engineering and stereoscopic imaging. In 2017 he was recognised as one of Australia's Most Innovative Engineers by Engineers Australia.

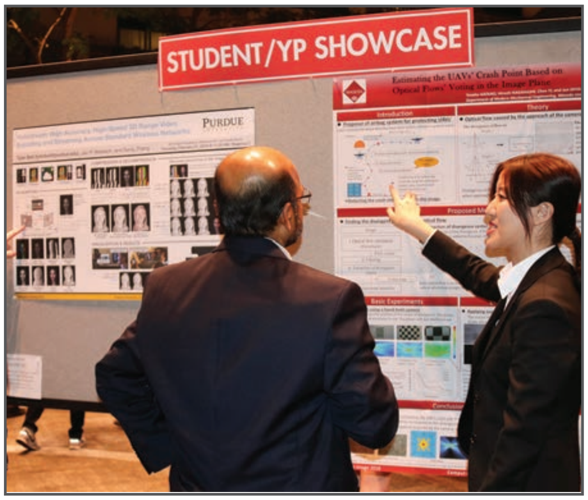
JOIN US AT THE NEXT EI!

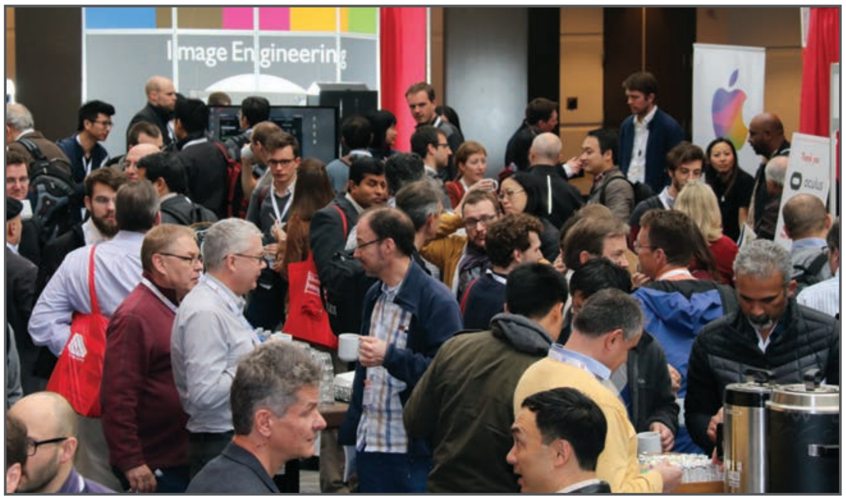
IS&T International Symposium on

Electronic Imaging

SCIENCE AND TECHNOLOGY

Imaging across applications . . . Where industry and academia meet!





AR is a fundamental and transformative interface and platform.
20+ years from my initial working demo to today's AR displays
Optimistic that it won't take another 20 to reach ubiquitous consumer AR!
My first working AR prototype
Today's platforms
Ubiquitous consumer AR

• SHORT COURSES • EXHIBITS • DEMONSTRATION SESSION • PLENARY TALKS •
• INTERACTIVE PAPER SESSION • SPECIAL EVENTS • TECHNICAL SESSIONS •

www.electronicimaging.org

IS&T
imaging.org