Unfolding the 8-Bit Era

Fabio Zünd[*]
fzuend@inf.ethz.ch

Pascal Bérard[†*]
pberard@inf.ethz.ch

Alexandre Chapiro[†*]
achapiro@inf.ethz.ch

Stefan Schmid[†*]
schmist@inf.ethz.ch

Mattia Ryffel[†]
mryffel@inf.ethz.ch

Markus Gross[†*]
grossm@inf.ethz.ch

Amit H. Bermano[†*]
amberman@inf.ethz.ch

Robert W. Sumner[†*]
sumnerb@inf.ethz.ch

ABSTRACT

We propose a hardware and software system that transforms 8-bit side-scrolling console video games into immersive multiplayer experiences. We enhance a classic video game console with custom hardware that time-multiplexes eight gamepad inputs to automatically hand off control from one gamepad to the next. Because control transfers quickly, people at a large event can frequently step in and out of a game and naturally call to their peers to join any time a gamepad is vacant. Video from the game console is captured and processed by a vision algorithm that stitches it into a continuous, expanding panoramic texture, which is displayed in real time on a 360 degree projection system at a large event space. With this system, side-scrolling games unfold across the walls of the room to encircle a large party, giving the feeling that the entire party is taking place inside of the game's world. When such a display system is not available, we also provide a virtual reality recreation of the experience. We show results of our system for a number of classic console games tested at a large live event. Results indicate that our work provides a successful recipe to create immersive, multiplayer, interactive experiences that leverage the nostalgic appeal of 8-bit games.

CCS Concepts

•**Computing methodologies** → **Image and video acquisition; Image processing;** *Virtual reality;*

Keywords

games, panoramic stitching, virtual reality

[*]ETH Zurich, Universitätsstrasse 8, 8092 Zürich, Switzerland
[†]Disney Research Zurich, Stampfenbachstrasse 48, 8006 Zürich, Switzerland

CVMP 2015, November 24-25, 2015, London, United Kingdom

 ISBN 978-1-4503-3560-7/15/11…$15.00
DOI: http://dx.doi.org/10.1145/2824840.2824848

Figure 1: Side-scrolling games unfold across the walls of the room to encircle the players, immersing them into the game's world. The games are played using the console, enhanced to allow up to eight players to participate, in the center of the room.

1. INTRODUCTION

Video games are a cultural phenomenon. Through their unique combination of visual, narrative, auditory, and interactive elements, video games provide an engaging medium of expression within our society. The *8-bit era*, dominated by the Nintendo Entertainment System (NES) [3], included a focus on side-scrolling graphics with pivotal leaps in game design, mechanics, and story that deeply influenced nearly every game that followed [23]. As a case in point, the original Super Mario Bros. series largely defined the platforming genre and pioneered a new level of game feel characterized by loose and fluid movement through an expansive world [20]. These games touched the lives of a huge number of gamers and their gameplay still holds up today.

Although 8-bit games have had a dramatic collective cultural impact, the actual experience of playing them is largely an individual one. With few exceptions, hardware and design limitations restrict

gameplay to one or two players in front of a low-resolution display. This setup confines gameplay to a small region, restricts social interaction, and limits the number of players that can enjoy a classic gaming experience. Even with modern game hardware, party games rarely extend beyond a small number of people playing in front of a display.

In this paper, we propose a custom hardware and software setup to transform classic side-scrolling games into collective experiences in which the games become immersive group activities. We take advantage of the nostalgic appeal of 8-bit games and their compelling yet accessible gameplay by using a classic NES console as our game hardware. Our system enhances the NES in two dramatic ways. First, we add a custom piece of hardware that takes as input eight NES gamepads and time multiplexes the output so that the real-time control is handed automatically from one gamepad to the next either every five seconds or based on progress through a game. Second, we capture the NES video output and direct it to a computer vision system that stitches video frames into a continuous, expanding texture similar to a panoramic photo. This texture is displayed live on a 360 degree projection system that allows the game to unfold on the walls of a large event space. When such a display system is not available, we also provide a Virtual Reality (VR) recreation of the experience. A conceptual illustration of our system is included in Figure 1, along with a photo of the system in action.

Taken together, our system transforms classic gaming into an immersive, cooperative multiplayer experience designed to enhance large parties and other social events. The eight-way multiplexing hardware encourages multiple people to play and adds a new level of social interaction on top of existing gameplay. Because control transfers quickly, people at a large event can frequently step in and out of the game. Whenever a gamepad is unused, others playing naturally call to their peers to join in before the control reaches the vacant gamepad. The panoramic stitching and 360 degree projection allow a side-scrolling game to encircle a large party, giving the feeling that the entire party is taking place inside of the game's world. By using a real NES console, as opposed to an emulator, we maintain the tangible connection to the physical Nintendo hardware and the nostalgic appeal of loading real game cartridges into the system. Successful operation sometimes even requires blowing on the cartridge's connectors to clear away dust, as many gamers fondly (or not so fondly) remember from their childhood.

Our core contributions include the conceptual design of our system that unfolds 8-bit side-scrolling games around a large party as well as the technical design of the time-multiplexing hardware and computer vision algorithms for panoramic stitching of video game input. We show results of our system on a number of NES games at a live event with over four-hundred participants as well as in a VR scene that recreates the feeling of the large event space.

2. RELATED WORK

Retro gaming is a general term referring to a modern community where old games, mostly those produced in the 1980s and early 1990s, are played or collected. Many people find the video games they played as children to have a nostalgic allure [19], resulting in a significant cultural impact of *old school gaming*. This connection has been acknowledged and leveraged by researchers for various purposes. Areas such as psychotherapy [8] and speech therapy [21] benefit from the engaging aspects of retro games to motivate patients. Generations impacted by classic games are subjected to targeted teaching methods that take their interests into account [9]. Others appreciate the elegant designs of early gaming systems and strive to preserve the characteristic visual look of pixel art when adapting content to fit modern architectures [13, 14]. In our work, we try to preserve the feeling of retro gaming as much as possible, while simultaneously adapting the content to be displayed in a modern immersive setting.

Immersive display has been an active topic for both research and industry in the past years. While pioneering efforts such as the CAVE automatic virtual environment [6] are complex to set up and may accommodate no more than a single user, modern systems often try to enhance the viewers' experience by augmenting standard display technologies. Commercially available systems such as IMAX [10] provide viewers with a wider field of view than standard cinemas. Projection [11] and additional illumination [22] can be employed in tandem with standard displays in order to present content to the peripheral vision of observers. Finally, modern VR prototypes shift the viewing experience to a wearable VR headset. In this work, we test our system on two different immersive display setups: a custom commercial 360 degree projection system and the popular Oculus Rift DK2 [18]).

Image tracking is used in our system to correctly place the current video frame in the global context of an expanding panoramic texture. Due to our target video setup, we consider only side-scrolling games. As a result, the tracked frame can only move horizontally. The speed of this movement, however, is determined by the player. It is not uniform and can include standing still or even backtracking. Camera movement in side-scrolling games was analyzed in depth by Keren [12].

To track the movement, corresponding points in the input frame and the output buffer must be found. Many algorithms to find scene correspondences exist, ranging from dense correspondence algorithms like optical flow [4] and stereo reconstruction [7] to sparse algorithms based on feature detection like SIFT [16]. Robust matching based on RANSAC [24] is used in applications such as panorama stitching. All of these methods deal with complex situations with many degrees of freedom. While these methods solve a wide array of problems, the tracking required for our application is limited to 1D shifts and requires real-time performance. For these reasons we choose a straightforward confidence weighted model, described in Section 4.2.

3. OVERVIEW

We present a system that bridges the gap between classic 8-bit side-scrolling console video games and state-of-the-art media display systems to deliver compelling, multiplayer, immersive game experiences. Figure 2 depicts an overview of our system's architecture. While our system is not limited to a specific console, we optimized it for the NES. The video signal generated by the console is captured by a tracking PC, with careful attention to quality and latency. The tracking PC identifies background motion in order to stitch the video frames together into an expanding panoramic texture image. We deployed and tested our system at the conference banquet during the Eurographics 2015 conference, which was held in a large event space that contains an integrated state-of-the-art 360 degree projection system. Our system used this display system to seamlessly wrap the game texture around the event room as players played. We also recreated the feeling of this live event in VR using the Oculus Rift DK2.

Our system is tailored for this party scenario, where several players engage in the game together. This cooperation is enabled through multiplexing several NES gamepads, activating only one of them at each point in time. The gamepads are switched between players based either on a fixed time interval or on the current position of the game in the 360 degree projection. The dynamic and automatic transition between active gamepad control encourages interaction

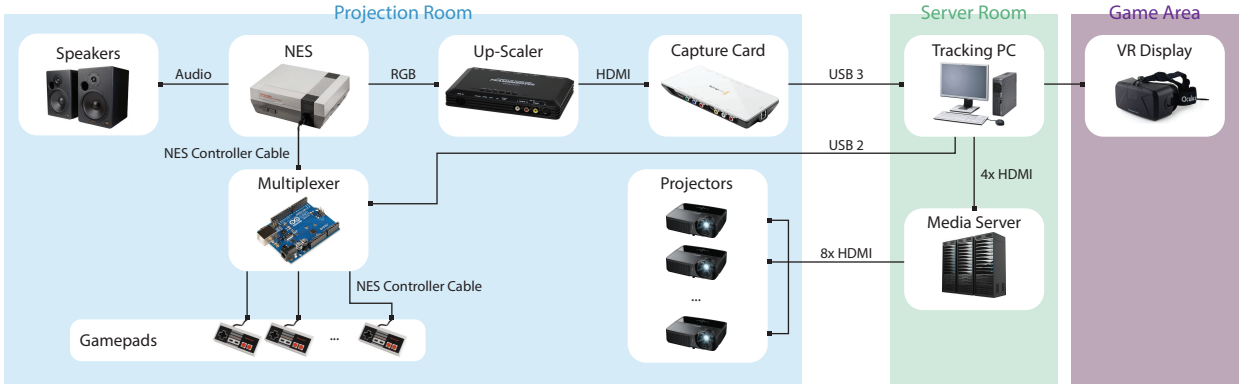

Figure 2: System architecture overview. The NES output video signal is first captured and sent for analysis. The tracking PC processes the video stream, tracks the background, and creates a wide, panoramic image. This image is either incorporated into a VR environment or sent to a 360 degree projection system. The projection system receives the video streams, processes, and outputs them to the projectors. For the projection configuration, the division between the projection hall and server room is indicated by blue and green backgrounds, respectively.

between the players for successful gameplay and increases the social aspect of the system.

In our VR demo, a simple scene is created based on the original 360 degree projection event space. The scene contains four walls which are textured with the tracked image. The player is located in the center of the room and can follow the progress of the game by turning his or her head.

4. SYSTEM

In this section, we discuss the different software and hardware aspects of our system in detail.

4.1 Video capture

The first step of our system is to capture the analog output of the gaming console. Although seemingly simple, this task turned out to be non-trivial. First, the NES outputs a 240p resolution signal (320×240 pixels). Most modern capture hardware does not handle this outdated standard well or does not handle it at all. An up-scaling device allows the conversion of this signal to a more modern and standard one. However, up-scaling operations typically include unwanted artifacts, such as vertical screen shaking. These subtle inaccuracies cause a jittering effect that slightly decreases the visual quality and significantly affects the tracking performance. Software stabilization techniques increase the latency of the system and are thus undesired.

Therefore, we choose to incorporate an up-scaler device that is optimized for low-resolution console signals. We find that the *XRGB-mini Framemeister* up-scaler [17] produces high-quality, low-latency signals for the aforementioned low-resolution obsolete video standard. By converting the video stream to 576p at 50 Hz, the device is able to improve signal fidelity and quality without increasing latency. The up-scaled signal is then captured using a *Blackmagic Design Intensity Shuttle* capturing device [2].

4.2 Tracking

We dynamically construct an expanding panoramic image of the game levels by copying the input frames into a wide output buffer with a fixed vertical resolution. The correct placement of the input frame relative to the output buffer is found by tracking game's apparent camera motion. Due to our target 360 degree projection system that wraps horizontally around the walls, we consider only horizontal side-scroller games. Thus, our tracking algorithm must consider only one degree of freedom of camera movement. We propose several strategies to perform this calculation and compose the input frames with the existing output buffer pixels. Figure 3 depicts an overview of our approach.

4.2.1 Matching Error

Given the previous frame position, we search locally in its neighborhood to find the horizontal pixel offset of the current frame relative to the previous one. The search is done bidirectionally but can be restricted to a single direction for specific games.

We compute a matching error for every one-pixel offset in the neighborhood. For normal camera speeds, a neighborhood ranging from -20 to +20 pixels is sufficient. For each offset value, a matching error is computed from all input pixels overlapping the non-masked pixels of the output buffer. An output pixel is masked if it has never been painted or if it is being overwritten. The error consists of the aggregate of the L1-norms of the color difference, normalized by the number of pixels in the overlap region. We use the L1-norm instead of the more common L2-norm since the two signals that we compare rarely match perfectly due to transient foreground elements. The L1-norm places a lower penalty on outliers and is favorable in this situation. The matching error is defined as

$$e = \frac{1}{N}\sum_{i=1}^{N} |Ro_i - Ri_i| + |Go_i - Gi_i| + |Bo_i - Bi_i|, \qquad (1)$$

where Ri_i, Gi_i, Bi_i are the red, green, and blue color values of the i-th input image pixel in the overlap region and Ro_i, Go_i, Bo_i reference the i-th pixel of the output buffer.

The overlap region is only sampled along a reduced number of lines for performance reasons. We use 40 lines in all of our experiments. We also manually exclude areas from the tracking if they contain static foreground elements like scoreboards and other UI features. These areas are stored as presets in our application so that games can be switched quickly. Once the matching errors are computed, we find the offset with the minimum error e_1 and update the current camera position.

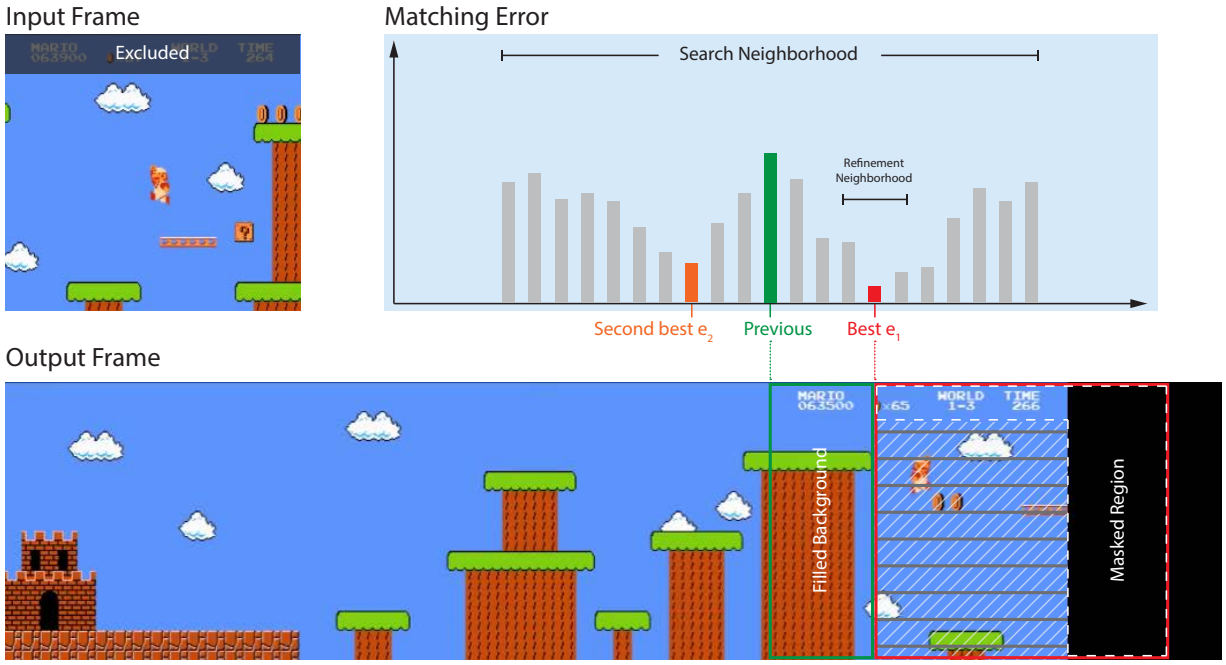

Figure 3: Camera tracking. Top left: the current input frame with a manually labeled region excluded from the tracking. Top right: Illustration of the matching error for camera movement estimation of the current input frame. Bottom: the output buffer with the current input frame position (red). A strip of pixels corresponding to the camera movement is filled in (green).

4.2.2 *Tracking Confidence*

During gameplay, the scene may change completely, such as when switching between levels or showing a *game over* screen. Since the camera cannot be tracked in this situation, our system keeps track of the current camera position and overwrites the current frame within the buffer. To detect such a screen refresh, a confidence value for the tracked position is computed. A non-minimum suppression is applied to the matching errors by labeling all of the offset positions that have two adjacent neighbors with higher matching errors. The minimum error e_1 is used together with the second lowest labeled matching error e_2 to compute the confidence:

$$\text{confidence} = 1 - \frac{e_1}{e_2}. \tag{2}$$

Finally, a threshold is applied to determine if the tracking succeeded. In our examples, we use a threshold value of 0.1.

4.2.3 *Refinement*

Since the camera can move by subpixel values due to the scaling and jitter in the input signal, we refine the current position at subpixel levels by computing the matching scores for N positions per pixel in a 2-pixel neighborhood. The position with the minimum error is the final frame position. We used $N = 10$.

4.2.4 *Frame Composition*

There are different strategies to composite the input frame into the output buffer. The pixels at the current position should always be taken from the input frame since this is where the actual game action takes place. When the camera moves, on the other hand, a strip of pixels can be filled in with the information from one or more previous frames. We propose two different strategies.

- Direct: In the simplest case we use only the previous frame to fill in the missing background pixels. Since the previous frame contains both foreground and background objects, both will be copied. Foreground objects will sometimes remain frozen in the buffer texture.

- Median: By taking into account multiple frames, we can estimate the background pixel colors more robustly and treat foreground objects as outliers. This estimation is done by storing the last F frames from which we compute the median color for each pixel. We found a value of $F = 20$ to be a good compromise. On one hand, more frames provide a better background color estimation. But, on the other hand, static foreground elements like score boards will produce smeared artifacts when additional frames are used.

Both modes have their advantages and disadvantages. Figure 4 depicts a side-by-side comparison. Since the direct mode is not able to suppress foreground elements, they will remain in the final output. Depending on their movement during processing, they can be squeezed, stretched, or torn apart. The median mode, on the other hand, can suppress foreground elements, but it bears two disadvantages. First, if the tracking is inaccurate the generated image is blurry. Second, foreground objects might suddenly appear or disappear as soon as they exit the active frame, depending on whether they are contained in the majority of the previous frames or not. The most appropriate method depends on the game and on which artifacts are preferred by the user. We found the sharper results of the direct mode to be most appealing and used this mode during deployment of our system.

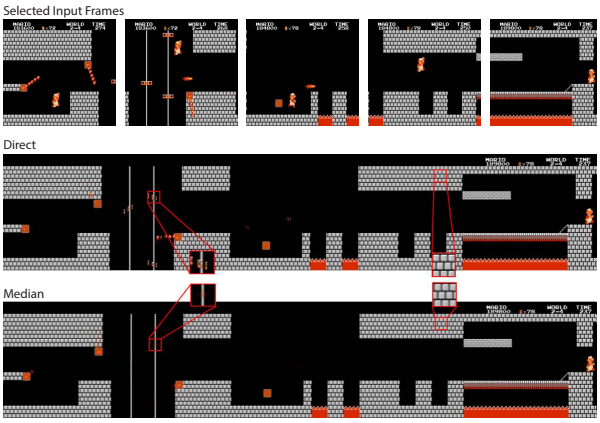

Figure 4: Comparison of frame composition modes. The direct mode does not suppress foreground elements and can result in artifacts that include stretching, squeezing, and tearing of foreground objects. The median mode suppresses foreground objects but can sometimes lead to blurring. Examples are highlighted by enlarged patches.

4.3 Display

We evaluated our system using two output configurations. The first consists of a 360 degree projection system in a large event space. The second places the user into an immersive VR environment.

4.3.1 360 Degree Projection

Our system was deployed during the conference banquet of Eurographics 2015. The banquet took place in a high-ceilinged rectangular hall equipped with a 360 degree projection system that can display seamless video on all four walls of the room, as seen on Figure 10. The input consists of four 1080p video signals, which are fed into two *Coolux Pandora's Box Quadserver Pro* [5] media servers. The media servers drive eight projectors, two for each wall, and take care of aligning and blending the inputs. The walls depict different aspect ratios and the servers distort the input images in order to cover the entire area. Our system compensates for this aspect ratio disparity with a GPU algorithm for real-time performance. Our NES console was placed in the center of the room and the game unfolded around the walls of the venue throughout the evening as attendees played.

4.3.2 Virtual Reality

For the VR immersion setting, we employ the Oculus Rift DK2 Head-Mounted Display (HMD) [18]. Inspired by the 360 degree projection scenario, the player is placed in the middle of a virtual room whose four walls depict the game world. The player enjoys a life-size virtual map that unfolds around him or her as the game evolves. Figure 5 shows our system in action.

4.4 Gamepad Control

In order to enhance the social experience of the players, our prototype utilizes eight gamepads allowing the fast exchange of active players. Players stand around the console, facing each other and the game above them. The console is placed on a table, and the gamepads are positioned in a circular manner around it, as depicted in Figure 6. Gamepads are activated automatically without pausing the game, which requires coordination between the players in order to achieve smooth gameplay.

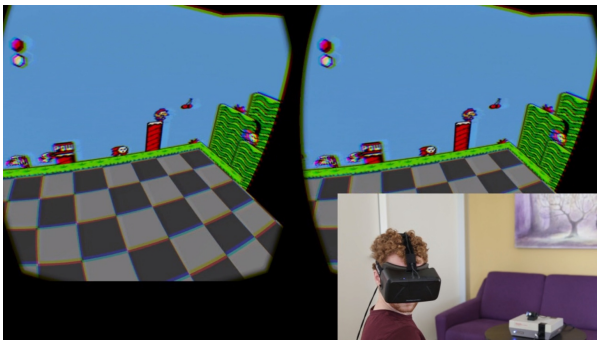

Figure 5: VR setup. Employing the Oculus Rift DK2, the tracked buffer spans across four walls in a scene inspired by the 360 degree projection setup. In the main image, the views shown to the player's eyes are presented. The player, shown in the lower right, enjoys a life-sized game realm as it unfolds during gameplay.

4.4.1 User Interaction

The switching mechanism we developed has two modes. The first mode progresses the control around the table after a fixed time period. The second mode gives control to the player that faces the current position of the game within the projected surrounding. In both cases, we noticed that visual cues must be given to players in order to achieve a smooth in-game transition. The visual cues are fourfold, addressing different needs of the players. First, as seen in Figure 6, a small light next to each gamepad indicates when the corresponding gamepad is active. This cue helps a player pick up the active gamepad. However, we observed that players may not notice the activity indication light when focusing on the gameplay. Therefore, second, each gamepad is assigned a number, and the active gamepad's number is always displayed next to the active playing region through the projection system, as depicted in Figure 6. Third, a transition between gamepads must be indicated. Without this additional stimulus, a short period of inactivity is induced until the new active player realizes it is his or her turn to play. Therefore, we set the transitions to always be in the same direction around the table, making them easier to follow. Both the displayed number and the activity indication light blink for two seconds before each transition. An alert player or an alert observer notices the blink and is ready to start playing as soon as control is passed to him or her. Forth, the active playing region is covered with the new active gamepad number for a short period of time, two frames, upon transition. These indicators yielded successful and smooth gameplay interactions during the event.

4.4.2 NES Gamepad Hardware

The NES gamepad is connected via a seven-wire cable to the console. The pinout is shown in Figure 7. Pin 1 to 3 are used to trigger and collect data from the gamepad, pin 4 and 5 are not used (not connected), pin 6 acts as ground and pin 7 is the voltage source. Since the gamepad is based on a Complementary Metal-Oxide-Semiconductor (CMOS), any voltage roughly around 5 V will work. An original gamepad features eight buttons: the four directions, start, select, A and B. Every button can assume two states, pressed or non-pressed, which are digitally modeled as *HIGH* and *LOW* states. These states reach the console via a poll-based mechanism. When the console wants to know the states of the gamepad's buttons, it sends a short pulse (*HIGH*) on the latch line (pin 2). This

Figure 6: Game setup for the 360 degree projection configuration. Left: The console is set on a designated octagonal table in the middle of the room. Eight gamepads are connected to it in a circular manner. Top right: The active gamepad number is indicated next to the playing region at all times. Bottom right: an indicator light is illuminated when the corresponding gamepad is active.

brief *HIGH* state of the latch line (otherwise *LOW*) is interpreted by the gamepad electronics as a request to store the current state of all buttons. This process is implemented with the help of a parallel-to-serial 8-bit shift register [15]. Subsequently, the captured button states can be polled, one-by-one, from the shift register by sending a *HIGH* pulse on the clock wire (pin 1). The requested information will be made available on the data line (pin 3).

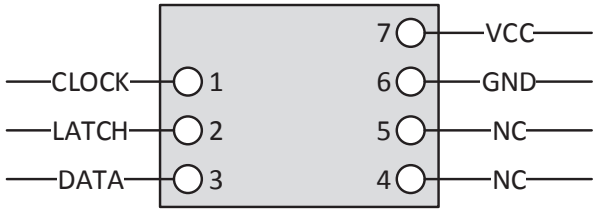

Figure 7: NES gamepad cable pinout.

4.4.3 *Gamepad Multiplexing*

The NES console supports up to two gamepads simultaneously. Since our design requires eight gamepads with control hand-over, we built a custom piece of electronics to selectively multiplex the eight gamepads. The system must be able to give control to any of the gamepads during a running game experience. Figure 8 depicts the schematic of the Arduino-based [1] multiplexing device and Figure 9 shows the actual hardware implementation. The bus width is given as the number in square brackets. Voltage source and ground (omitted in the scheme) for all gamepad connectors, including the one connecting to the console, are connected in parallel. Gamepad ground and Arduino ground are hooked up together to have the same reference potential for the multiplexer/demultiplexer components. All latch lines and clock lines coming from the eight gamepad connectors are connected to an 8-bit demultiplexer module. The eight data lines are connected to an 8-bit multiplexer unit. The single demultiplexer and multiplexer output line for latch, clock, and data is connected to the console gamepad plug. The arrow heads indicate the input/output behavior for all wires and buses. A three-wire selection bus is connected in parallel to all multiplexer/demultiplexer components, making it possible to select one of the eight gamepads and connect its wires through to the console. The selection wires are operated by an Arduino. The system software determines the currently active gamepad and sends the request over USB to the serial interface of the Arduino where the firmware is driving the selection bus to enable the chosen gamepad. The Arduino is also used to operate the LEDs associated with each gamepad, enabling the light during the time when a player has control. The multiplexing hardware is completely transparent for the NES console and gamepads, and does not require any changes to the original hardware.

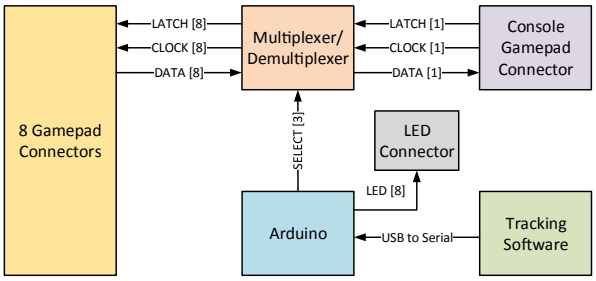

Figure 8: Simplified scheme of the gamepad multiplexing hardware. Bus width is indicated by numbers in square brackets.

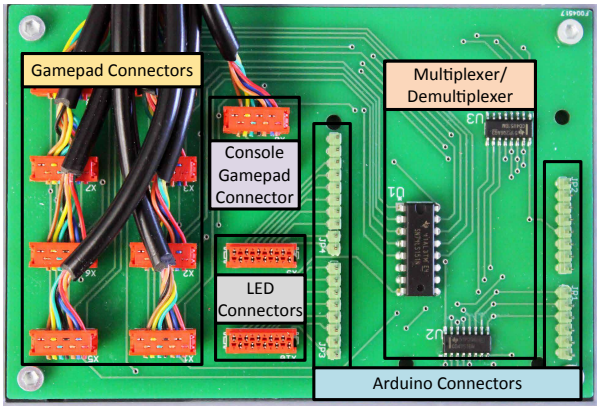

Figure 9: Custom-designed multiplexing hardware for eight NES gamepads.

5. RESULTS

The tracking PC, performing the tracking and producing the output, is a i7 3.2Ghz machine with a Geforce GTX 770 graphics card. It generates twenty 12 M-pixel frames per second, with approximately 120 ms latency, most of which originates from the capture hardware. For the 360 degree projection configuration, the projection system introduces additional delay, but does not affect the frame rate. Tracking takes 3 ms to 4 ms on average on thus does not significantly contribute to the overall latency.

We tested our system on seven classical games, both in the VR configuration, as shown in Figure 5, and the 360 degree projection configuration, summarized in Figure 10. Figures 11 and 12 depict the unfolded output buffer of our tracking process using the direct method. For context, selected input frames from the console are shown above the buffer, in their respective positions. The game world is clearly and continuously captured in the unfolded buffer.

Figure 11 depicts the well known *Super Mario Bros.* trilogy. These games include many foreground characters that pass by while

Figure 10: The system as it was deployed during the conference banquet of Eurographics 2015. Different games provide varying ambiance, and immerse players and spectators in the 8-bit realm.

side-scrolling, and are therefore sometimes visible in the tracked buffer. Figure 12 depicts the other games that were tested. *Excitebike* is a fast-paced game in which the whole buffer is transversed very quickly, implying quick camera motion per frame of up to 15 pixels. By contrast, *Castlevania* is slower in this aspect. Note how the buffer is overwritten when the scene changes from a forest setting to the castle interior. In *Life Force*, the input frame includes a black bar on the right hand side, and therefore the current position within the buffer is clearly noticeable, as well as a level change. *Probotector* is a game in which the map evolves very clearly and cleanly.

Unfortunately, ground truth data is not available. However, one can still visually validate the results by comparing a segment of the output buffer with a single input frame as shown in Figure 11 and Figure 12. Mismatches produced by foreground elements, drift or artifacts are directly visible.

6. CONCLUSION

We have presented a hardware and software system to transform classic side-scrolling games into immersive, multiplayer experiences. By using a real NES console and games, we take advantage of the nostalgic appeal of the 8-bit game era. We tested our system live at a large event with over four-hundred people and observed strong engagement. Throughout the evening, people continually played the system, actively cooperating with one another to advance in the game levels. We observed a new range of social dynamics, with strangers talking and laughing with one another, warning each other to be ready when control was passed on to the next player, and calling other participants to join when a gamepad was vacant. The 360 degree projection contributed a special ambiance to the evening, and provided entertainment for those not playing. At a few instances during the evening, a collective cheer from the audience erupted when the players beat a difficult level.

Limitations in our system direct us to opportunities for future research. Games with highly repetitive features can lead to tracking failures, as can be seen in the *Excitebike* result in Figure 12. Our system does not explicitly distinguish foreground sprites from background elements, leading to frozen sprite characters in the stitched textures. Tracking improvements as future work could address these issues and also offer new ways to enhance the game experience. Explicitly distinguishing between background and foreground elements would allow us to add additional depth perception into the VR version of our system so that the background is offset in space from the foreground.

Latency is a critical issue in any interactive system. Although our capture and processing algorithms are designed to execute as fast as possible, some latency is unavoidable. In the VR setting, our test players did not notice latency that would hinder them from successfully playing the game. However, some latency was noticeable during the Eurographics 2015 event, as the 360 degree projection system incurred additional latency to our capture and processing.

As our target was 360 degree projection, we focused on side-

scrolling games and developed a tracking algorithm designed for this use case. Our method does not currently support vertical scrolling or more complicated camera movement. Future work could consider alternate display geometries beyond circular projection as well as vertical scrolling. Our VR system provides an ideal tool to test and debug various setups in preparation for real-world deployment. Likewise, we demonstrated two control switching methodologies: temporal switching after a fixed number of seconds or switching based on the physical position of the game projection. Exploring other control switching modes is an area of future work.

Perhaps the most interesting opportunity for future work entails accommodating more advanced game consoles. Although our system was designed for the NES console, we are confident it would work on other classic 2D consoles such as the 16-bit Super Nintendo Entertainment System (SNES) [3]. However, future console generations focus on 3D graphics and violate the assumptions of our tracking algorithm. Dynamic scene reconstruction for 3D games combined with VR could provide a novel, compelling way to experience such games.

7. ACKNOWLEDGEMENTS

The copyright to all imagery from *Super Mario Bros.*, *Super Mario Bros. 2*, *Super Mario Bros. 3*, and *Excitebike* lies with *Nintendo Co., Ltd.* The copyright to all imagery from *Castlevania*, *Life Force*, and *Probotector* lies with *Konami Corporation*. We would like to thank Alessia Marra and Maurizio Nitti for their artistic support and Jan Wezel for his engineering work.

8. REFERENCES

[1] Arduino Uno Board Overview. Website, February 2014. http://arduino.cc/en/Main/ArduinoBoardUno.

[2] Blackmagic design intensity shuttle. Website, July 2015. https://www.blackmagicdesign.com/products/intensity.

[3] Nintendo. Website, July 2015. https://www.nintendo.com.

[4] T. Brox, C. Bregler, and J. Malik. Large displacement optical flow. In *Conference on Computer Vision and Pattern Recognition, 2009.*, pages 41–48. IEEE, 2009.

[5] Coolux. Pandoras box server. Website. http://www.coolux.de/products/pandorasboxserver/.

[6] C. Cruz-Neira, D. J. Sandin, and T. A. DeFanti. Surround-screen projection-based virtual reality: the design and implementation of the CAVE. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 135–142. ACM, 1993.

[7] Y. Furukawa and J. Ponce. Accurate, dense, and robust multiview stereopsis. *Transactions on Pattern Analysis and Machine Intelligence*, 32(8):1362–1376, 2010.

[8] J. E. Gardner. Can the Mario Bros. help? Nintendo games as an adjunct in psychotherapy with children. *Psychotherapy: Theory, Research, Practice, Training*, 28(4):667, 1991.

[9] M. Guzdial and E. Soloway. Teaching the Nintendo generation to program. *Communications of the ACM*, 45(4):17–21, 2002.

[10] IMAX Corporation. IMAX: a motion picture film format and a set of cinema projection standards, 2010.

[11] B. R. Jones, H. Benko, E. Ofek, and A. D. Wilson. IllumiRoom: peripheral projected illusions for interactive experiences. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 869–878. ACM, 2013.

[12] I. Keren. The theory and practice of cameras in side-scrollers. Website, March 2015. http://www.gdcvault.com/play/1022243/Scroll-Back-The-Theory-and.

[13] J. Kopf and D. Lischinski. Depixelizing pixel art. In *Transactions on graphics*, volume 30, page 99. ACM, 2011.

[14] F. Kreuzer, J. Kopf, and M. Wimmer. Depixelizing pixel art in real-time. In *Proceedings of the 19th Symposium on Interactive 3D Graphics and Games*, pages 130–130. ACM, 2015.

[15] A. LaMothe. *Game programming for the propeller powered hydra*. Parallax, Inc, S.l, 2006.

[16] C. Liu, J. Yuen, and A. Torralba. Sift flow: Dense correspondence across scenes and its applications. *Transactions on Pattern Analysis and Machine Intelligence*, 33(5):978–994, 2011.

[17] Micomsoft. DP3913515 XRGB-mini framemeister compact up scaler unit. Website. http://www.micomsoft.co.jp/xrgb-mini.htm.

[18] Oculus VR. Oculus Rift development kit 2. Website, 2014. https://www.oculus.com/dk2/.

[19] J. Suominen. The past as the future? nostalgia and retrogaming in digital culture. *Fibreculture*, 11, 2008.

[20] S. Swink. *Game Feel: A Game Designer's Guide to Virtual Sensation*. Morgan Kaufmann Game Design Books. Taylor & Francis, 2009.

[21] C. T. Tan, A. Johnston, A. Bluff, S. Ferguson, and K. J. Ballard. Retrogaming as visual feedback for speech therapy. In *SIGGRAPH Asia 2014 Mobile Graphics and Interactive Applications*, page 4. ACM, 2014.

[22] A. Weffers-Albu, S. de Waele, W. Hoogenstraaten, and C. Kwisthout. Immersive TV viewing with advanced Ambilight. In *International Conference on Consumer Electronics*, pages 753–754. IEEE, 2011.

[23] M. J. P. Wolf, editor. *The Video Game Explosion: A History from PONG to PlayStation and Beyond*. Greenwood Press, 2008.

[24] W. Zhang and J. Košecká. Generalized ransac framework for relaxed correspondence problems. In *Third International Symposium on 3D Data Processing, Visualization, and Transmission*, pages 854–860. IEEE, 2006.

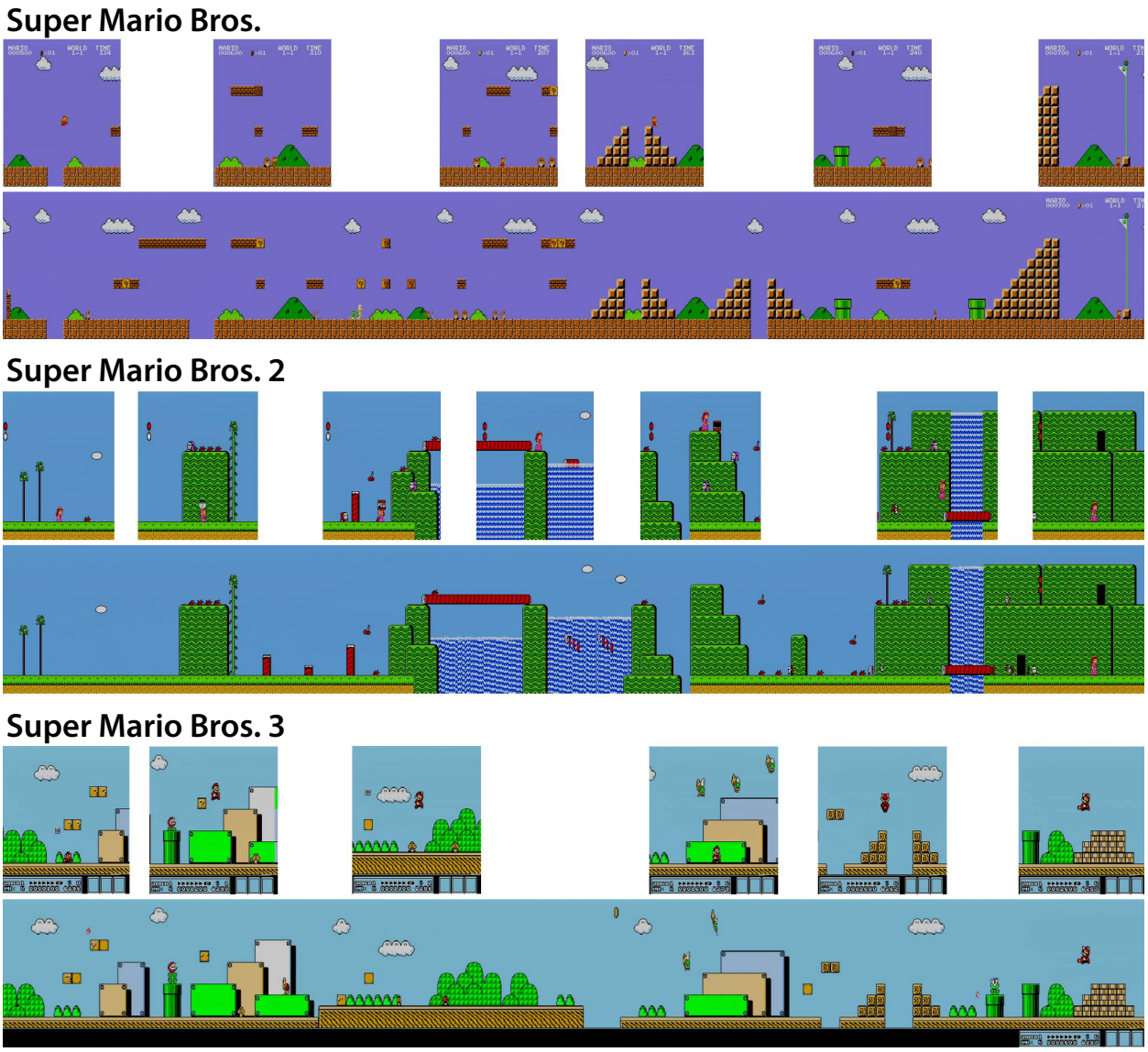

Figure 11: The unfolded output buffer of our tracking process, for the *Super Mario Bros.* trilogy. For every game, the first row depicts selected input frames coming from the console, in their respective positions within the output buffer. The second row depicts the output tracking buffer. The combination of input frames continuously maps out the game realm in the buffer.

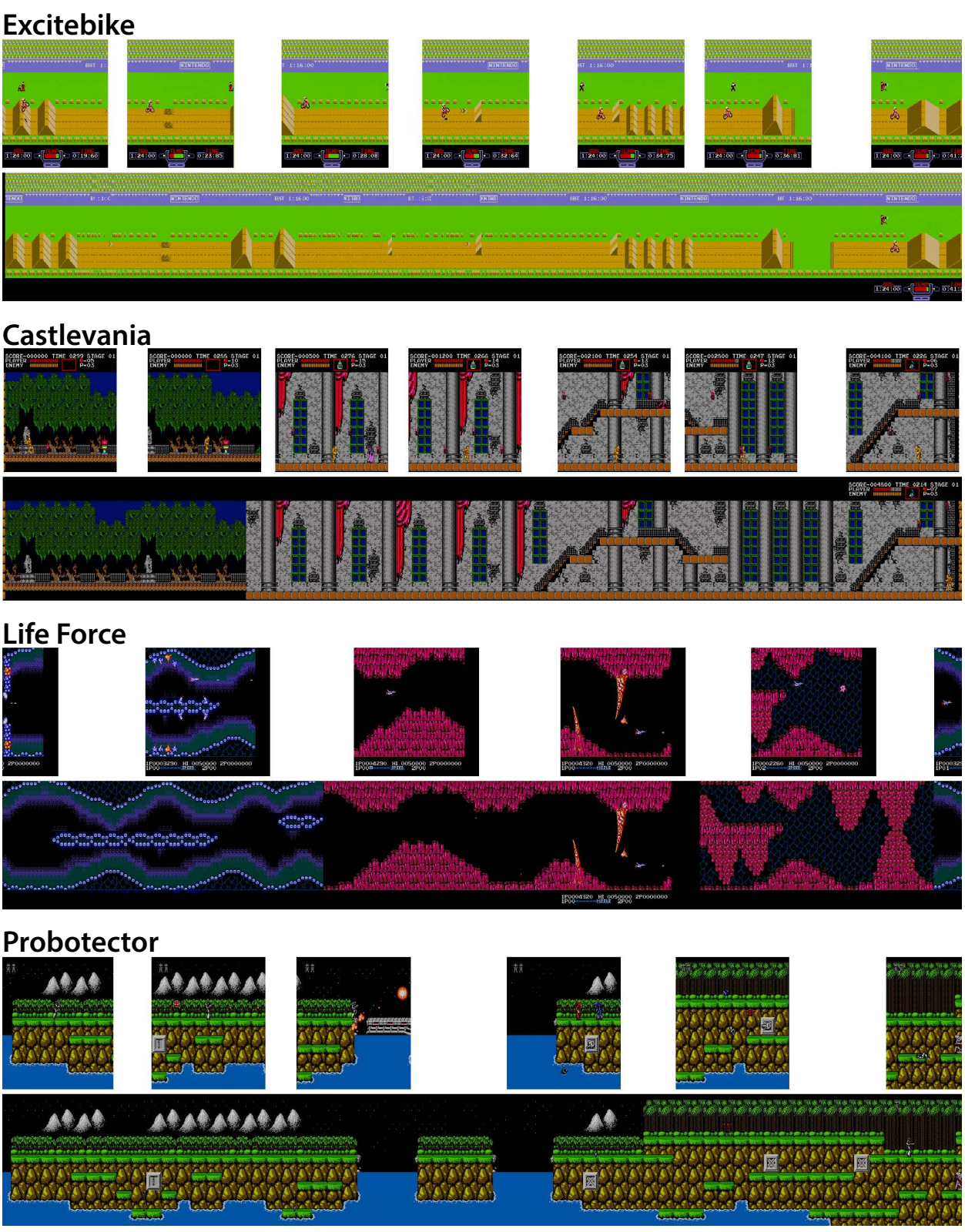

Figure 12: The unfolded output buffer of our tracking process, for four of the tested games. For each game, the first row depicts selected input frames coming from the console, in their respective positions within the output buffer. Scene changes overwrite the current position in the buffer.