Self-calibrating optical object tracking using Wii remotes

Ian F. Rickard[a]

James E. Davis[a]

[a]University of California Santa Cruz, 1156 High St, Santa Cruz, CA, USA

ABSTRACT

This paper describes an optical tracking method developed using Wii Remotes. The motion and rotation of an object with markers and an accelerometer is tracked using the optical sensors in two Remotes. Initialization is complicated by the nature of the Wii Remote's sensor: while it can track up to four points, there is no trivial means to uniquely identify an individual marker. To resolve this ambiguity, a brute force approach is used. All possible correspondences are considered, using the Remotes' and tracked object's accelerometers as inclination sensors to identify potentially correct ones. The resulting method is applied to create a largely self-calibrating six-degree-of-freedom input device. We also provide documentation sufficient for others to use the Wii Remote in research.

Summary

Keywords: Computer Graphics, Camera Calibration, Object Registration, Smart Cameras, Input Devices, Wii Remote

1. INTRODUCTION

The Wii Remote represents a new class of sensor device. It combines a high frame-rate image sensor and point tracker System-on-a-Chip and a 3-axis accelerometer into an inexpensive device that can be trivially and wirelessly connected to most computers. In considering uses for such a device, multiple camera systems become an obvious applications due to the device's cost and ease of connection.

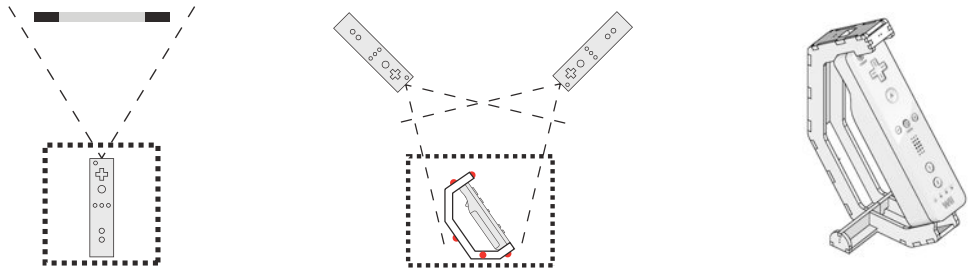
Unfortunately, the nature of their sensor adds a challenge to many potential applications. While it returns the position, size, and intensity of up to four points of infrared (IR) light, there is no trivial way to tell one point from another. This abstraction eliminates most modern approaches for point-based registration as they require either a known correspondence between model and image points, or a much larger number of projected points. However, traditional registration methods do not consider the potential for external constraints on camera pose.

At rest, the Wii Remote's accelerometer detects gravity's pull and can thus be used as an inclinometer. It provides a good estimate of the pitch and roll (together referred to here as inclination) of the Remote leaving only yaw (azimuth) unconstrained. Using this constraint, we develop a technique to register an object of known geometry to an image with only three or four points. This technique is then applied to create a self-calibrating six-degree-of-freedom input device.

Instead of attempting to intelligently determine correspondences, the small number of unknowns allows for a guess-and-check approach. Each possible mapping of markers on the tracked object (which we call an artifact) to observed points is attempted and the resulting transformation is checked for consistency with the observed points and inclinations. Though this method produces a correct mapping less than 40% of the time, incorrect mappings rapidly become inconsistent as the tracked object rotates.

In section 2 we discuss related registration techniques, and research with Wii Remotes. An overview of the system is presented in section 3, with details of the calibration, tracking, refinement, and re-calibration in section 4. Section 5 presents some paths additional research could take. Finally, Appendices A and B provide an introduction to, and technical reference for the Wii Remote.

Further author information: (Send correspondence to I.F.R.)
I.F.R.: E-mail: inio@soe.ucsc.edu
J.E.D.: E-mail: davis@soe.ucsc.edu



(a) (b) (c)

Figure 1. **a:** when interacting with the Wii console, the user (who's influence is indicated by the dotted box) moves a Wii Remote which detects the position of the fixed sensor bar in its field of view (bounded by dashed lines). **b:** With this system the user manipulates an artifact which is tracked by fixed Wii Remotes. **c:** A render of the fourth revision of the handheld artifact.

2. RELATED WORK

Marker-based 6DOF tracking of objects has been the foundation of Augmented Reality (AR) systems since their inception in the early 90s. Recently Kotake et al. have considered applying inclinometers to reduce the number of markers needed and to accelerate initialization[1] or assist initialization of edge-based tracking without markers.[2] However, neither of these approaches consider the case here of correspondence-free point-based initialization. While other work has been done in the area of simultaneous pose and correspondence determination for this problem,[3] it targets much higher point counts than the Wii Remote could provide.

The Wii Remote has at this point seen use in research,[4] though little has been formally published. Most well known in the field of alternate applications of the device is Johnny Lee's work at CMU using the Wii Remote for finger, hand, or head tracking, as well as multi-touch 2D input devices.[5] More closely related to this work is Kreylos' 6DOF Input device where a single hand-held wii Remote is pointed towards a fixed tracking target.[6] However, that configuration greatly limits the range of rotation on two axes, as the tracking target must remain within the Wii Remote's limited field of view.

3. SYSTEM OVERVIEW

In comparison to the operation of the Wii Console, the system described here is "backwards". Where the Wii console has the user manipulating a sensor which detects fixed IR sources (Fig. 1(a)), we instead fix the sensors, with the user manipulating the artifact and it's IR sources (Fig. 1(b)).

The artifact (Figs. 1(c) and 2(b)) contains six IR LEDs. They are aligned with the faces of a cube, and mounted on faces flush with the artifact's convex hull to minimize occlusions. As each emitter is visible over slightly more than a full hemisphere, this results in three to five emitters being visible from any direction, neglecting occlusions from the user's arm or power tether. Previous artifact designs are shown in Figure 2(c).

The artifact also includes a Wii Remote for two primarily purposes: to provide an accelerometer and ergonomic grip. Using a Wii Remote here has the advantage of providing buttons, vibration feedback, accurate pointer-based cursor manipulation, additional capabilities through its expansion port, and future the potential for audio feedback. To reduce weight, and because the LEDs are already drawing too much power for batteries to be reasonable, the artifact's Remote's batteries are replaced with a voltage regular powered from the same power tether as the LEDs.

Two Remotes are suspended above the workspace with their cameras directed down towards it. These act as tracking sensors for the artifact's IR emitters. The distance from the work area to the sensor cameras needs to be at least twice the desired work area size. We found a sensor distance of 50cm produced a comfortable work area while maintaining good resolution. Figure 2(a) shows a typical workspace.

As these Remotes are located above the workspace, replacing their batteries and accessing their buttons to reconnect them proved a significant annoyance. Similarly to the artifact's Remote, the batteries are replaced with an external power source. Further modification was done to provide external syncing, with a circuit being added to pull the sense line for the SYNC button low when a button near the workspace is pressed.

(a) (b) (c)

Figure 2. **a:** Overview of the system. **b:** The current version of the handheld artifact. Two markers are clearly visible, with a third barely visible at near where the power tether leaves the unit. **c:** Evolution of the artifact design, in order by rows from top-left.

4. SYSTEM OPERATION

Within this section, unless otherwise specified, units are in sensor pixels or centimeters as appropriate. These work out to be about the same with a sensor distance of 50-100cm. A rotation of 10° in the handheld also results in about the same displacement. Note that while this system operates in units of sensor pixels, the Wii Remote outputs data with 1/8th sensor pixel precision. Analysis using a motion-control table shows it's total accuracy to be better than 1/4 pixel, so this precision is justified.

4.1 Initializing the Camera and Artifact Poses

At startup, the relative position of the sensor Remotes to each other and to the artifact is unknown. As the number of points on the artifact is low, a brute-force approach can be used to determine the artifact's pose relative to each camera. The relative transformation between the two tracking Remotes can then be easily derived.

For each subset of three points visible in a camera and the 24 logical permutations * of the artifact's emitters onto those points, the solutions to the three-point pose from perspective problem are found. [7] This results in at most 96 possible poses for the subset of emitters, from which the artifact's camera-relative translation and rotation (represented as a quaternion) are derived using Horn's closed-form method. [8]

For each of the putative poses, the artifact's emitters facing less than 100° away from the sensor (later referred to as front facing emitters) are projected into image space ($p_{1\ldots N}$) and the artifact's Remote's normalized acceleration is transformed into camera space ($\mathbf{x}$). These are then compared to the observed points ($o_{1\ldots M}$) and normalized acceleration ($\mathbf{g}$) to assign the pose a score (s) based on equation 1. A value of 10 for the scaling factor w was found to work well empirically. The lowest score for each camera is found and the transform for each camera is initialized to the inverse of the artifact's relative transform.

$$s = w \arccos^2(\mathbf{x} \cdot \mathbf{g}) + \sum_{m=1}^{M} \min_{n \in 1..N} |o_n - p_m|^2 \tag{1}$$

The lowest scored pose is correct only about 40% of the time. However, incorrect poses quickly decay into inconsistency as the artifact is moved and rotated. Because consistency is continuously evaluated while tracking (as discussed in the next section) these incorrect poses are short-lived.

*Considering the emitters as vertices of an octahedron and ignoring occlusions, if only three points are observed they must correspond to corners of one of the eight triangles. Since this triangle must be front-facing, only three permutations are possible. Thus $8 \times 3 = 24$ potential correspondences.

4.2 Tracking and Consistency Checking

Once an initial artifact pose and camera configuration is known, gradient descent is used to track motion of the artifact. Two passes are used to maintain the discovered correspondence between points. In the first pass, only translation is optimized with rotation locked at the previous frame's value. In the second pass, both rotation and translation are free to vary. For both passes, the error metric is the sum square image-space distance between each observed point and the closest front-facing emitter.

The new pose is checked for consistency with the observed points and acceleration with two tests. In the first test, each point returned by a camera, in order of distance to the closest protected emitter, is assigned to the nearest unassigned projected front-facing emitter. The system is considered inconsistent if the distance for any point is greater than two sensor pixels. In the second test, the artifact's acceleration is calculated by subtracting the recovered artifact-relative velocity of the previous frame from that of the current frame, and dividing by the time step, and adding the mean acceleration measured by the cameras. The system is considered inconsistent if this calculation differs from the observation by the greater of 0.3g or 30%.

In the case of inconsistency, one of two actions is taken. If it has been less than 10 seconds since a full calibration (described in the previous section) or the system has been inconsistent for more than 100 out of the last 1000 samples of active tracking, then the history (described in the next section) is discarded and full re-initialization is performed. Otherwise, only the artifact pose is re-initialized, as follows. It's pose relative to each camera is determined using the same approach as a full initialization, but considering projection error into both cameras, not just the one for which the pose is being estimated, when scoring the putative poses. The resulting pair of best-scored poses are averaged and then gradient descent optimized as in normal tracking to produce the final new artifact pose.

4.3 Calibration Refinement

To allow the camera properties (both extrinsic and intrinsic) to be refined, a history of up to 100 point correspondence pair observations is kept. Each entry consists of a pair of points on the artifact, along with the image-space position of those emitters in both cameras. This history is populated during initialization with synthetic data, generated by projecting opposing pairs of emitters into each camera.

When the gradient descent optimization has an extremely low risidual error (RMS distance below 0.25 pixels) and the artifact is moving slowly (less than 5cm/s and 30°/sec), the observed emitters for each sensor are checked for overlap. If, under these conditions, two emitters are simultaneously observed by both tracking sensors, this observation is considered for addition to the history.

The observation is treated as an 8-vector (the image-space position of each emitter in each camera), and the history is searched for previously recorded observations with a total cartesian distance from the new one of less than 10 pixels. For efficiency, the history is kept in a k-d^9 tree to accelerate this search. If a matching observation less than 10 seconds old is found, the new observation is dropped. If the nearest matching observation is older than that, it is removed from the history and the new one is added. If no entry matches and the history is already at 100 correspondence pairs, the oldest entry is dropped and the new one added.

Whenever the observation history is updated, a low priority thread is started to perform two operations. First, it rebuilds the k-d tree. Until the new tree is built, any new correspondence pairs are ignored. It then performs a simultaneous gradient descent optimization on the focal length of both sensors, and relative pose of the second. The error metric used in this optimization is the sum of two components, both based on the 3D location of each point as found using parallax. The first error component is the sum distance between each point's position projected into each camera it's observed location. The second is the error in the distance between the 3D points and the actual distance between emitters in cm. The intent of these two metrics is to simultaneously maintain stereo calibration and geometric coherence. When done, the updated camera calibration takes effect on the next frame.

5. FUTURE WORK

At the moment, the geometric configuration of the artifact is arbitrary. The position of the LEDs could be optimized to minimize the occurrence of ambiguous cases during initial calibration. The capabilities of the Wii Remote in the artifact are also not fully used. Vibration, the player LEDs, or sound could be used to warn the user when they approach the limits of the working volume.

The described refinement method is fragile to outliers. A better approach would be use apply more traditional stereo calibration techniques with RANSAC. The approach used was developed as it relied only on already-implemented techniques and took advantage of known scene structure.

With the cost and ease of connection of Wii Remotes, there's no reason to limit the system to two sensor cameras. Additional cameras could be used to expand the tracking area or improve the system's resiliency to dropout. Alternately, as the Wii Remote is sensitive to a fairly wide range of IR wavelengths, narrow-band filters and different LEDs could be used to track two artifacts through the same space.

APPENDIX A. USING THE WII REMOTE

Many parts of this and the following section are based on the work of others shared online through Wikis and IRC chat. Most early work was contributed by editors of the Wiili wiki.[10] In particular, Andy753421 and Volsung contributed greatly in the early days: making initial contact, discovering that it was a HID device, and enumerating the reports. Except where otherwise mentioned, the following should be considered the work of the Wii Remote research community in general. Where possible, specific credit has been given.

A.1 Connection

At present, only one method is known to connect the Wii Remote to a host computer. The procedure as as follows:

1. Place the Wii Remote into discoverable mode by either pressing the "1" and "2" buttons simultaneously, or pressing the "SYNC" inside the battery compartment. The Remote will stay in discoverable mode for 20 seconds.
2. From the host computer, initiate a Bluetooth inquiry for device class major:5 minor:1. The Wii Remote can then be identified by it's Product ID (0x0306) and Vendor ID (0x057e).
3. Open a baseband connection to the Wii Remote. It has been suggested that performing an SDP query at this point will make the connection process more reliable.
4. Depending on API support, either open the Wii Remote as a Bluetooth HID device or open L2CAP channels on PSM 17 (output) and PSM 19 (input).

The above described connection process is unreliable, with many factors appearing to influence the success rate. The following variations may have some effect:

- After reading the device descriptor but before opening any L2CAP channels or performing an SDP query, the Wii console closes the baseband, waits one second, and then reconnects to the Remote.
- Some implementations, Wii console included, perform an SDP query before opening the L2CAP HID channels.
- The authors believe that connections following a SYNC-button press show a higher success rates than those following pressing 1+2.

It's worth noting that the Wii Remote can also quickly reconnect to a previous host when any button is pressed. The mechanism through which this reconnection occurs is unfortunately, not yet understood.

A.2 General Use

Once connected to a host computer, communication with the Wii Remote occurs though output and input HID "reports". These labels are from the perspective of the host, with output being messages to the Remote and input returning from it.

If L2CAP channels are used, an additional single-byte HID header is present on reports. Output reports should be prefixed with the byte 0x52, while input reports will have a prefix of 0xA1. These values are part of the Bluetooth HID specification and are not unique to the Wii Remote.

The Wii Remote will send back user input in one of many reports, documented in section B.3. All reports except 0x3D include the button state in the first two bytes, labeled as they are named. The remaining fields in those reports have the following meaning:

xlr_x / y / z Accelerometer reading. x / y / z refer to the left/bottom/front of the remote, respectively. These are believed to be the high eight bits of 10-bit values, with some of the low bits encoded in the five unknown bits in the button bytes.

ir_data Data from IR sensor, formatted as specified.

extn Data from the extension controller. These are bytes as read from the controller, starting at 0x00.

A.3 IR Sensor (Camera)

The IR sensor is a infrared-sensitive image sensor with a point tracking processor. Though the actual sensor resolution is believed to be 128x96, the point tracker reports positions with 1/8 pixel precision, resulting in an effective resolution of 1024x768. When tested using LEDs with identical packaging and emission patterns, we have found the sensor to be several times more sensitive to light at 940nm than 850nm (two common wavelengths for IR LEDs). However, it has been reported to be usefully sensitive to wavelengths as short as 532nm when the Wii Remote's IR pass filter is replaced with a laser line filter.

The Wii console uses a sequence of six commands to activate the IR sensor. These command frequently fail, so before sending the next command, driver software should wait for the response from the previous. If a failure is reported or no response is received for many milliseconds, the driver should re-send the previous command

1. enable the IR sensor's clock (report 0x13)
2. assert the IR sensor's chip-enable (report 0x1A)
3. write 0x01 to IR register 0x30
4. write 9-byte sensitivity data to IR register 0x00 though 0x08
5. write 2-byte sensitivity data to IR register 0x1A and 0x1B
6. write mode number to IR register 0x33
7. write 0x08 to IR register 0x30

These commands were initially reverse engineered by either Cliff or Marcan, with the specifics lost to abandoned wikis. The meaning of the sensitivity blocks is described in section B.4.1

APPENDIX B. WII REMOTE TECHNICAL REFERENCE

B.1 HID Output Reports

Output reports are sent from the host device to the Wii Remote. In bitfield descriptions "`0?`" and "`1?`" indicate that the Wii console clears or sets these bits when sending the report, but the meaning of the bit is unknown.

B.1.1 Rumble

The low bit of the first byte of all output reports controls the Wii Remote's rumble feature. If this `rumble` bit is set (1) the rumble motor is turned on. If the bit is clear (0) the rumble motor is turned off.

B.1.2 0x10: Unknown

BYTE	BIT 0	1	2	3	4	5	6	BIT 7
0	?							rumble

The Wii Remote's HID descriptor mentions this report, but it has never been observed to have a unique effect. It has a one-byte payload, the last bit of which controls the rumble feature. It may exist solely to control the rumble feature. On success, the Wii Remote will respond with report 0x22.

B.1.3 0x11: Configure player LEDs

BYTE	BIT 0	1	2	3	4	5	6	BIT 7
0	p4led	p3led	p2led	p1led	0?		1?	rumble

Controls the four LEDs at the base of the Wii Remote (near the expansion port). If p*n*led is 1, then the player *n* indicator will be lit. Any combination of LEDs may be turned on simultaneously. On success, the Wii Remote will respond with report 0x22.

B.1.4 0x12: Select input report

BYTE	BIT 0	1	2	3	4	5	6	BIT 7
0	0?					stream	1?	rumble
1	input_report							

Controls input reports returned by the Wii Remote. The value in input_report is the input report code desired 0x30 to 0x37, 0x3D or 0x3E. When report 0x3E is requested, reports 0x3E and 0x3F are returned every other report. If stream is set, reports will be returned continuously. Otherwise, reports will only be sent when the payload changes.

B.1.5 0x13: IR clock enable

BYTE	BIT 0	1	2	3	4	5	6	BIT 7
0	0?					irck_en	1?	rumble

if irck_en is set, enable the IR sensor's 24MHz clock. On success, the Wii Remote will respond with report 0x22.

B.1.6 0x15: Status request

BYTE	BIT 0	1	2	3	4	5	6	BIT 7
0	0?							rumble

System status request. Wii Remote will respond with report 0x20.

B.1.7 0x16: Write

BYTE	BIT 0	1	2	3	4	5	6	BIT 7
0	0?					i2c	0?	rumble
1	addr[23:16]							
2	addr[15:8]							
3	addr[7:0]							
4	0	0	0	length				
5	data0							
⋮	⋮							
20	data15							

Writes data to EEPROM (when i2c is 0) or one of the I^2C peripherals (when i2c is 1). for I^2C writes, addr[23:17] selects the device (see section B.5) and addr[16:8] is ignored. On success, the Wii Remote will respond with report 0x22. If fewer than 16 bytes are being written, data should be placed in the lower data fields.

B.1.8 0x17: Read

BYTE	BIT 0	1	2	3	4	5	6	BIT 7
0	0?					i2c	0?	rumble
1	addr[23:16]							
2	addr[15:8]							
3	addr[7:0]							
4	length[15:8]							
5	length[7:0]							

Reads data from EEPROM (when i2c is 0) or one of the I^2C peripherals (when i2c is 1). for I^2C writes, addr[23:17] selects the device (see section B.5) and addr[16:8] is ignored. On success, the Wii Remote will respond with one or more 0x21 reports.

B.1.9 0x1A: IR chip enable

BYTE	BIT 0	1	2	3	4	5	6	BIT 7
0	0?					ir_en	1?	rumble

if `ir_en` is set, asserts the enable line to the IR sensor. On success, the Wii Remote will respond with report 0x22.

B.1.10 Audio-related output reports

Reports 0x14, 0x18 and 0x19 are related to audio playback through the Wii Remote's speaker. This capability is not yet understood to a useful degree, so in the interest of space these reports are omitted. For the current state of understanding of these reports, see the wikis[10][11][12]

B.2 HID Input Reports - Response & Status

Input reports fall into two classes: system status or responses to output reports, and user input data. Responses have a high nibble of 2, while user input data has a high nibble of 3.

B.2.1 0x20: Status response

BYTE	BIT 0	1	2	3	4	5	6	BIT 7
0	?							
1	?							
2	p4led	p3led	p2led	p1led	stream	spk_en	ext_con	?
3	?							
4	?							
5	v_bat							

Sent in response to output report 0x15. `pnled`, `stream`, `spk_en`, and `ext_con` indicate the player LED status, user input reporting mode, speaker power enable, and detection of an extension controller, respectively. `v bat` is the output of the battery voltage AD, though the mapping to actual voltages is not known.

B.2.2 0x21: Read data

BYTE	BIT 0	1	2	3	4	5	6	BIT 7
2	size				status			
3	addr[15:8]							
4	addr[7:0]							
5	data0							
⋮	⋮							
20	data15							

Sent in response to output report Reads. `size` is one less than the number of data bytes returned. `satus` is one of the following codes: 0: success; 7: received I^2C NACK; 8: unmapped address. `addr` is the starting address of this block.

B.2.3 0x22: Command acknowledge

BYTE	BIT 0	1	2	3	4	5	6	BIT 7
0	?							
1	?							
2	cmd							
3	?							

B.3 HID Input Reports - User Input

User input reports all follow a fairly consistent format. The first two bytes of all reports but 0x3D are the buttons. In the interest of space, these bytes are omitted when identical to report 0x30. The fields of user input reports are described above in section A.2.

B.3.1 0x30: User input: buttons only

BYTE	BIT 0	1	2	3	4	5	6	BIT 7
0	?	?	?	plus	up	down	right	left
1	home	?	?	minus	a	b	one	two

B.3.2 0x31: user input: buttons and accelerometer

BYTE	BIT 0	1	2	3	4	5	6	BIT 7
2 ··· 4	xlr_x[9:2], xlr_y[9:2], xlr_z[9:2]							

B.3.3 0x32: User input: buttons and 8-byte extension data

BYTE	BIT 0	1	2	3	4	5	6	BIT 7
2 ··· 9	ext0 ··· ext7							

B.3.4 0x33: User input: buttons, accelerometer, and IR format 3

BYTE	BIT 0	1	2	3	4	5	6	BIT 7
2 ··· 4	xlr_x[9:2], xlr_y[9:2], xlr_z[9:2]							
5 ··· 16	ir_data (format 3)							

B.3.5 0x34: User input: buttons and 19-byte extension data

BYTE	BIT 0	1	2	3	4	5	6	BIT 7
2 ··· 20	ext0 ··· ext18							

B.3.6 0x35: User input: buttons, accelerometer, and 16-byte extension data

BYTE	BIT 0	1	2	3	4	5	6	BIT 7
2 ··· 4	xlr_x[9:2], xlr_y[9:2], xlr_z[9:2]							
5 ··· 20	ext0 ··· ext15							

B.3.7 0x36: User input: buttons, IR format 1, and 9-byte extension data

BYTE	BIT 0	1	2	3	4	5	6	BIT 7
2 ··· 11	ir_data (format 1)							
12 ··· 20	ext0 ··· ext8							

B.3.8 0x37: User input: buttons, accelerometer, IR format 1, and 6-byte extension data

BYTE	BIT 0	1	2	3	4	5	6	BIT 7
2 ··· 4	xlr_x[9:2], xlr_y[9:2], xlr_z[9:2]							
5 ··· 14	ir_data (format 1)							
15 ··· 20	ext0 ··· ext8							

B.3.9 0x3D: User input: 21-byte extension data

BYTE	BIT 0	1	2	3	4	5	6	BIT 7
0 ··· 20	ext0 ··· ext20							

B.3.10 0x3E/0x3F: User input: buttons, accelerometer, and IR format 5

When format 0x3E is requested, output alternates between these formats. 0x3E contains two points of IR data and 0x3F contains the other two.

BYTE	BIT 0	1	2	3	4	5	6	BIT 7
0	?	xlr_z[7:6]/[3:2]		plus	up	down	right	left
1	home	xlr_z[9:8]/[5:4]		minus	a	b	one	two
2	xlr_x[9:2]/xlr_y[9:2]							
3 ··· 20	ir_data (format 5)							

B.4 IR Sensor Reference

B.4.1 Configuration strings

The initialization procedure in section A.3 makes use of two configuration strings which control the camera and point-trackers behavior. The table below shows what the authors have learned of these strings through empirical testing.

Config 1 (00-08)	twoorseven	0x00	0x00	0x71	0x01	0x00	maxsize	0x00	gain
Config 2 (1A-1B)	gainlimit	minsize							

twoorseven Unknown function. Wii console uses 2 for most settings and 7 for the highest sensitivity.

minsize & maxsize Range of point areas the sensor will track. Measured in sensor pixels.

gain Larger values reduce sensor gain. Behavior is non-linear.

gainlimit Must have a value less than gain for sensor to function.

B.4.2 Data formats

The IR sensor's point tracker outputs data in one of three known formats, 1, 3, and 5. Format 1 encodes only the location of each point:

BYTE	BIT 0	1	2	3	4	5	6	BIT 7
0	x0[7:0]							
1	y0[7:0]							
2	y0[9:8]		x0[9:8]		y1[9:8]		x1[9:8]	
3	x1[7:0]							
4	y1[7:0]							
5 ··· 9	*repeat above for points 2 and 3*							

Format 3 adds a size measurement, possibly indicating the point's size relative to the configured size range:

BYTE	BIT 0	1	2	3	4	5	6	BIT 7
0	x0[7:0]							
1	y0[7:0]							
2	y0[9:8]		x0[9:8]		size0[3:0]			
3 ··· 11	*repeat above for points 1 through 3*							

Format 5 includes a bounding box, and peak intensity measurement. The bounding rectangle rectangle is output in the 128x96 sensor resolution. This format only includes two points in each report and alternates between returning points 1+2 and 3+4.

BYTE	BIT 0	1	2	3	4	5	6	BIT 7
0	x0/2[7:0]							
1	y0/2[7:0]							
2	y0/2[9:8]		x0/2[9:8]		size0/2[3:0]			
3	?	xmin0/2[6:0]						
4	?	ymin0/2[6:0]						
5	?	xmax0/2[6:0]						
6	?	ymax0/2[6:0]						
7	?							
8	intensity0/2[7:0]							
9 ··· 17	*repeat above for point 1/3*							

B.5 Memory Maps

The Wii Remote's internal EEPROM contains several pieces of data. The only part of interest is the block from 0x16-0x1F (repeated at 0x20-0x29) which contains calibration data for the accelerometer. This data is laid out as follows:

BYTE	BIT 0	1	2	3	4	5	6	BIT 7
0	`xlr_x_0g[9:2]`							
1	`xlr_y_0g[9:2]`							
2	`xlr_z_0g[9:2]`							
3	`?` (low bits?)							
4	`xlr_x_1g[9:2]`							
5	`xlr_y_1g[9:2]`							
6	`xlr_z_1g[9:2]`							
7	`?` (low bits?)							
8 ··· 9	`?` (checksum?)							

The three I2C devices (speaker controller, IR sensor, and extension port) are selected by setting the i2c bit in a read write operation and using one of the following values for the high byte (bits 23-16) of the address. The low bit of this byte is not significant.

`0xA2` Speaker controller

`0xA4` Extension port

`0xB0` IR sensor

ACKNOWLEDGMENTS

The authors would like to thank Professor Kevin Karplus, for developing the original concept of using Wii Remotes for 6DOF manipulation, and the early Wii Remote hackers for taking the critical first steps.

REFERENCES

[1] Kotake, D., Satoh, K., Uchiyama, S., and Yamamoto, H., "A hybrid and linear registration method utilizing inclination constraint," *ISMAR 2005: 4th IEEE and ACM International Symposium on Mixed and Augmented Reality* , 140–149 (Oct. 2005).

[2] Kotake, D., Satoh, K., Uchiyama, S., and Yamamoto, H., "A fast initialization method for edge-based registration using an inclination constraint," *ISMAR 2007: 6th IEEE and ACM International Symposium on Mixed and Augmented Reality* , 239–248 (Nov. 2007).

[3] David, P., DeMenthon, D., Duraiswami, R., and Samet, H., "Softposit: Simultaneous pose and correspondence," *Int J Comput Vis* **59**, 259–284 (September 2004).

[4] Schou, T. and Gardner, H. J., "A wii remote, a game engine, five sensor bars and a virtual reality theatre," *OZCHI '07: Proceedings of the 19th Australasian conference on Computer-Human Interaction* , 231–234 (2007).

[5] Lee, J. C., "Wii remote projects." http://www.cs.cmu.edu/johnny/projects/wii/ Accesssed 9 Nov 2008 (December 2007).

[6] Kreylos, O., "Wiimote hacking." http://idav.ucdavis.edu/~okreylos/ResDev/Wiimote/MainPage.html Accessed 9 Nov 2008 (September 2007).

[7] Fischler, M. A. and Bolles, R. C., "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Comput Graph Image Process* **24**(6), 381–395 (1981).

[8] Horn, B. K. P., "Closed-form solution of absolute orientation using unit quaternions," *J Opt Soc Am* **4**(4), 629–642 (1987).

[9] Bentley, J. L., "Multidimensional binary search trees used for associative searching," *Comm ACM* **18**(9), 509–517 (1975).

[10] "Wiili: Main page." http://www.wiili.org/index.php/Main_Page Accessed 30 Nov 2008.

[11] "Wiimote project: Main page." http://wiki.wiimoteproject.com/ Accessed 30 Nov 2008.

[12] "Wiibrew: Main page." http://wiibrew.org/wiki/Main_Page Accessed 30 Nov 2008.