

RemoteEyes: A Remote Low-Cost Position Sensing Infrastructure for Ubiquitous Computing

Changsong Shen, Baosheng Wang, Florian Vogt, Steve Oldridge and Sidney Fels
Human Communication Technologies Lab

Department of Electrical and Computer Engineering, University of British Columbia
2356 Main Mall, Vancouver, BC, Canada V6T 1Z4

Tel +1-604-822-4583, Fax +1-604-822-5949, Email: {csshen, baosheng, fvogt, steveo, ssfels}@ece.ubc.ca

ABSTRACT

This paper describes the architecture of a remote low-cost position sensing infrastructure for ubiquitous computing. To reduce the cost, each tracked object carries an inexpensive active tag that emits a modulated near-infrared signal, which is received and decoded by a camera module. Each camera module consists of a CMOS camera connected to a Field Programmable Gate Array (FPGA) and Ethernet communication hardware. Remote sensing can be achieved by means of Ethernet. The use of inexpensive FPGA to implement the most time- and data-critical tracking algorithms provides a cost-effective platform with low latency. The preliminary results suggest it is successful at tracking objects and feasible for achieving medium fidelity tracking performance in both space and time.

Keywords: position, sensing, network, FPGA, ubiquitous computing

INTRODUCTION

Context information is required by many ubiquitous computing applications to provide a transparent interface through which the user and the machine form a seamless system. Location and position of people and the objects they interact with are one of the most important contextual properties. Numerous systems have been developed to determine location information, such as the widely-used global position system (GPS) and wireless enhanced 911(E911) [6] services. However, the GPS and E911 services are unable to accurately determine this information inside buildings. Because the overwhelming majority of ubiquitous computing applications are indoors, a local positioning system (LPS), operating within the localized domain of a building or even a single room, but using remote access, can be an important component of these applications.

To provide absolute 3D position information, various tag-based systems have been researched. Previous work on implementing these tag-based LPS for remote indoor use mainly focused on the designs of sensors with different transmission media. A RF-based system called “RADAR” can offer an accuracy of about 1-3 meters by using the IEEE 802.11 WaveLAN wireless networking technology [9], as does the PinPoint 3D-iD system[10]. However, this is not realistic for some applications since the object being tracked must then support a wireless LAN, which may be impractical on small or power-constrained devices. In the Active Bat [1] system, tags emit an ultrasonic pulse to a grid of ceiling-mounted receivers in response to a request a controller sends via short-range radio. The system can locate tags to within 9cm of their true position for 95% of the measurements at a rate of 25 computations per room

per second. Due to the large fixed-sensor infrastructure installed throughout the ceiling, this system is costly and hard to deploy. In [4, 11, 12, 13], inexpensive sensors, i.e. infrared tags, are used as an alternative. However, [4] only locates objects but does not give 2-D or 3-D position. [11] uses a four-wire network, which is costly and not popular. Moreover, its update time is up to ten seconds. Sumi et al. use an “artificial retina” connected to a microcontroller to track and decode modulated tags similar to our approach [12, 13]. However, due to the limitations of the microcontroller, their tracking system selectively places bounding boxes around the bright spots when detected and processes each one in a serial manner. Thus, the system does not scale to large number of tags. Further, the resolution of their camera is quite low (128x128) reducing the fidelity of the tracking. We desire a system that offers a low-cost, low-latency, medium fidelity and remote accessible infrastructure that supports a large number of tags and cameras.

In this paper, we use low-cost camera modules that track inexpensive near-infrared tags. Each tag has a unique identifier, allowing multiple tags to be tracked simultaneously. Each camera module consists of a CMOS camera connected to a Field Programmable Gate Array (FPGA) and communication hardware. In our current implementation, camera modules (also called tracking units) communicate to a host server over the internet. While a number of solutions are available for implementing the TCP/IP stack for Ethernet, we use a commercial board with an Intel XScale embedded processor. Our requirement for long distance transmission, low cost and reasonable bandwidth led to our decision to use Ethernet rather than other communication protocols such as USB [3], RS-232 [8] or direct line [2]. Though,

our solution is not dependent on any particular protocol and is not the focus of this paper.

For an effective camera-based position sensing system, multiple cameras are required for 3D positioning as well as maintaining line-of-sight. Thus, managing large quantities of video data is one of the obstacles in implementing vision-based approaches for local positioning and other object tracking algorithms. For example, in [8], a low-cost 450MHz PC is used to tracking persons. The PC software bundled with the stereo cameras computes dense disparity images of size 320×240 pixels at a rate of only about 4Hz. In [3], a 2.4GHz Pentium 4 or an equivalent dual-processor system is required for two cameras at 640×480 and 30 frames per second in order to track the objects in real-time. We wish to apply tracking algorithms using cameras that require enhanced performance without increasing cost.

We use FPGA technology to meet the demands of the LPS design. Our current system uses Altera’s low-cost Cyclone EP1C12, which costs about \$10CDN per chip. By using inexpensive programmable logic, not only is the cost of the system greatly reduced for high-performance tracking, but also the different application requirements regarding cost and tracking performance trade-offs can be flexibly satisfied when the embedded CPU and FPGA chips share the computing tasks. More importantly, the tracking algorithms can be adjusted in the field when needed, and scaled by migrating the algorithms to more cost-effective FPGA chips with negligible design effort. The interface between the embedded processor and FPGA chips is based on memory-mapped I/O for simplicity and flexibility. An additional feature of our approach is that we directly configure the FPGA from the embedded CPU, providing further time and cost savings while remaining highly flexible. In contrast, traditional FPGA based systems are configured either through a download cable or some configuration device, such as an external ROM.

The remainder of this paper is organized as follows. First, we propose an overview of the system architecture. Then the process pipeline implementation is presented. Next, we describe our implementation and hardware. Finally, we draw conclusions and propose future work.

SYSTEM ARCHITECTURE

As shown in Figure 1, our system is designed to track people-sized objects in large, indoor spaces remotely, where the active tags/badges are worn by the people or objects being tracked.

The proposed system is comprised of multiple remote units connected to a host computer over Ethernet. Each remote unit applies a CMOS digital camera chip, a low-cost FPGA, and an embedded microprocessor. Tracking

and recognition of tags are performed by each remote unit. The resulting 2D coordinates of the tags are transferred to the host computer for further processing. From the 2D coordinates of multiple cameras, the host computer is able to calculate the 3D location of a tag, based on a calibrated coordinate system.

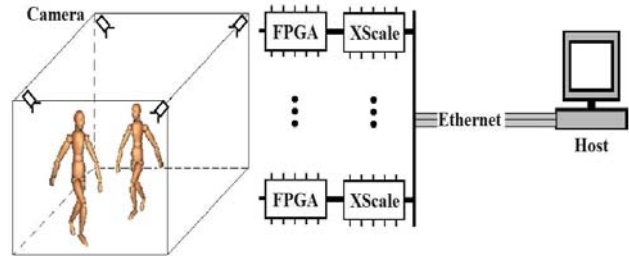


Figure 1. Deploying sensing units in a remote building

The system processing pipeline is shown in Figure 2. The main tasks include: blinking pattern encoding, raw image capturing, bright spot detection (BSD), bright spot grouping (BSG), invalid bright-spot removal, blinking pattern decoding and 3D coordinate transformation.

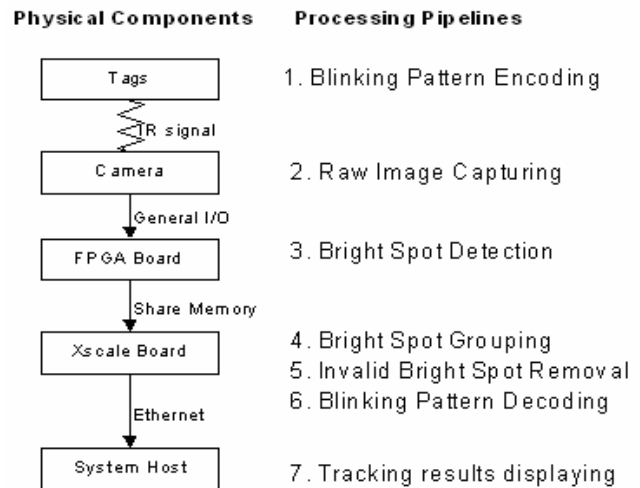


Figure 2. Processing pipeline of the proposed system

In the proposed system, bright spots are emitted by the blinking tags and captured by the camera module. The FPGA board is responsible for detecting the bright spots in the image from the camera module. Both grouping bright spots and decoding of blinking patterns are performed on the XScale board. Finally, the host PC determines the objects’ 3D coordinates based on multiple cameras and displays tracking results.

PROCESS PIPELINE IMPLEMENTATION

Blinking Pattern Encoding

We use an active tag to create blinking bright spots for tracking. The tag is a near-infrared LED that blinks a

unique bit pattern for identification at a known rate. The transmission of the pattern starts with a start code, then a unique ID. We currently use a small microcontroller to modulate the IRED at 60Hz. It behaves as a serially encoded bit stream. Hence, the length of the bit pattern impacts the latency for decoding a complete tag ID. For example, an 8 bit code with 5 start bits (supporting up to 256 simultaneously tracked tags) requires $(8+5)/60 = 217\text{ms}$ to transmit for decoding. Once a tag has been located, predictive tracking can be used to reduce tracking latency as updates can occur every time when a bright spot has been detected.

Raw Image Capturing

We use a surface mount OV6130 Black and White Camera chip for our image sensor. It is soldered onto a customized board with multiple configuration choices. It can capture 8-bit, 352×288 images at a maximum of 60 fps. Moreover, an I²C bus is used to configure the camera chip with multiple features. We place a bandpass infrared filter over the lens to reduce background noise. Data from the camera and the camera clock feeds the bright spot algorithms.

Bright Spot Detection (BSD)

The BSD algorithm is implemented on the FPGA to locate all the bright spots in a video frame. The bright spots come from the active LEDs in a particular frame. The algorithm design is constrained by several factors. First, the FPGA contains limited memory. Second, the video data comes into the FPGA row by row and is not randomly accessible. Third, we desire minimal latency. Thus, we need to process pixel data once it arrives from the camera rather than use a frame buffer that incurs a frame delay. As a result of these constraints, the algorithm should avoid random access of pixel data and does not use a frame buffer.

Our algorithm works by scanning the image row by row as the camera data arrives. It looks at each horizontal scanline pixel by pixel. If it sees a positive difference between the previous and current pixel intensity value, and the intensity difference is greater than a pre-defined threshold, it will register an up-edge.

Table 1. BPGIN assigned to each pixel

0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	0	2	1	1	0	0
0	3	2	1	1	1	0
0	0	2	1	1	0	0
0	0	0	1	0	0	0
0	0	0	0	0	0	0

As shown in Table 1, if an up-edge has been detected, the algorithm assigns a non-zero bright pixel group identification number (BPGIN) to the pixel; otherwise, the pixel would be assigned a zero. Adjacent bright pixels are assigned the same BPGIN number. Afterwards, the algorithm looks for a corresponding down-edge. If the algorithm fails to find any down-edge within a given number of pixels, the pixel group is rejected as being too big to be a tag, and rejects the initially detected up-edge, and starts looking for another up-edge. Upon reaching the end of a pixel row, the horizontal centers of any detected bright spots are passed to the bright spot grouping algorithm.

Bright Spot Grouping (BSG)

The BSG algorithm is needed because the camera or tags often will vibrate slightly from its equilibrium position, even when it appears stationary. A jittery bright spot could appear in the image as two or more distinct bright spots when actually it is one point.

After the BSD block has finished processing the bright pixel data from a given video frame, it sends the output data to BSG algorithm, implemented on the XScale microprocessor. In addition, for each group of pixels, a minimum and maximum coordinate are assigned. After this preliminary grouping stage, a center coordinate is computed by taking the average between the maximum and minimum coordinates for each pixel group. If the distance between the centers of any two bright spots is less than a predefined threshold value, these two bright spots will be grouped together. The new center coordinate will be calculated by weighting the coordinates of both bright spots, where the weight is decided by their sizes. The size of the merged bright spot is the sum of each spot size.

However, depending on the orientation of the IRED relative to the lens of the camera, the IRED signal could appear as non-contiguous bright pixel blocks on the video frame. As a result, pixels with different BPGIN could belong to the same IRED signal. Further grouping must be done to ensure these non-contiguous pixel blocks originating from the same source are grouped back together. Grouping decision is based on the relative proximity among different pixel groups. If their centre coordinates are within a pre-defined minimum distance, they are grouped together to form a single tag.

Invalid Bright Spot Removal (IBSR)

The grouped bright spots are passed to an invalid bright spot removing filter. This filter checks that bright spots are not too small or too large according to various tuned parameters. This process removes bright spots due to non-IRED light sources and from IREDs that are too close to the lens, which cause a vertical stripe. Occasionally, horizontal stripes appear due to noise, and these are also

removed by this process. These checks are sufficient to filter out most spurious bright spots with our current setup. Additional tests, such as checking the bright spot aspect ratio can also be performed at this stage if necessary.

Blinking Pattern Decoding

This stage takes the bright spot data and examines its variations over time for meaningful patterns, encoded as a series of bits that represents a unique identifier for the blinking object. The IRED's blink at the same rate as the camera's frame rate. We use the start code to determine if the asynchronous stream of data from a particular tag is synchronized with camera. We do this by having one of the off time of the start code to be slightly shorter than the others allowing detection of the proper start code if the camera is synchronized. If it is, then the subsequent bright spot is considered a 1 and if there is not a bright spot it is counted as a 0. If there are too many 0s in a row, the tag is considered out of sight. Occasionally, the system is out of sync with a particular tag, but this is only for one identification broadcast as the tag adds a small delay after each broadcast. This novel technique allows us to send data at 60bps (matched to the camera rate) with only a small loss of id packets.

PROCESSING AND HARDWARE

Figure 3 shows the remote sensing unit that was implemented. The right side of the figure shows the camera attached to the FPGA board. The XScale module attaches to the FPGA board with two connectors and multiple processors can be stacked as needed.

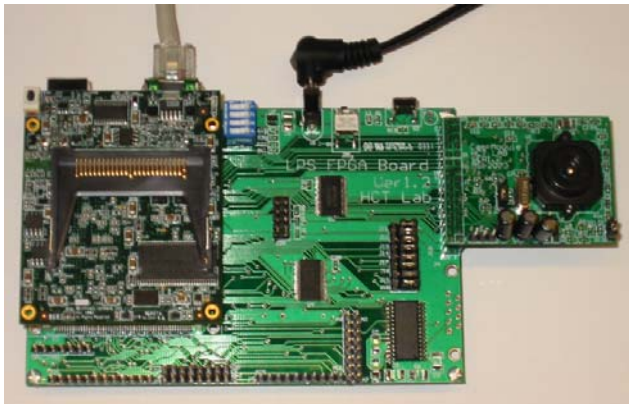


Figure 3. Top view of the Remote Sensing Unit

A typical tag is also shown in Figure 4.



Figure 4. Top view of the IR tag with two LEDs

The FPGA Board has an Altera Cyclone-series EP1C12 chip, with 200Kbits on-chip memory. Processing speeds can be as high as 200MHz. The FPGA implements the BSD algorithm. The XScale Board is a commercial CPU board with an Intel PXA 250 (XScale) Applications Processor. The operational speed is 400MHz and the system bus is 100MHz. It supports 64MB SDRAM and 32MB flash memory and includes multiple communication interfaces, such as USB, Ethernet, RS-232 etc.

The display and transforming of each tag's coordinates from the multiple cameras' 2D coordinates to an absolute 3D coordinate is performed by a host PC. This is not computationally expensive so we use a 400MHz personal computer with 128Mbytes SDRAM for our tracking host. It uses TCP/IP over Ethernet to communicate with all the tracking modules. At this point, our host behaves as a LPS server so that applications that require tag position information can request it from it. It runs Debian Linux.

The coverage area and resolution are dependent on the distance between the camera and objects as well as lens used for camera. For a typical distance, two metres, the resolution and coverage area are 0.3cm and 1.15m² respectively using a 30° lens; 1.1cm and 16m² respectively with a 90° lens. In other words, a 4(w) × 8(l) × 3(h) m³ room can be covered at 1.1cm resolution with only two cameras using 90° lenses.

To reduce costs further, all functions could be performed on the FPGA/XScale combination with only minimal changes.

CONCLUSION AND FUTURE WORK

The increasing ubiquitous computing applications for long distance indoor use make it necessary to develop a remote position sensing infrastructure. This causes a major challenge in cost during the implementations of the latter. Previous work in [1-13] fails to either provide the remote capability or achieve good performance and low cost simultaneously.

In this paper, we present a new remote low-cost position sensing system. It provides a position sensing infrastructure for ubiquitous computing. In the future, we will do more detailed user evaluations about the precision of this system. We plan to develop context-aware applications based on this infrastructure for exhibitions and museums continuing the work in [12-13]. This platform is also the basis of our work with single and multi-person laser tracking [5].

ACKNOWLEDGEMENTS

We would like to thank Zion Kwok and Kara Poon.

REFERENCES

- [1] A. Harter, A. Hopper, P. Steggles, A. Ward, and P. Webster, "The anatomy of a context-aware application", Proceedings of 5th Mobicom 1999, pp. 59-68, Seattle, WA, August 1999, ACM Press
- [2] B. R. Kyker, "Local positioning system", WESCON/95. Conference record. "Microelectronics Communications Technology Producing Quality Products Mobile and Portable Power Emerging Technologies", pp. 756-761, 7-9 November 1995
- [3] E. Aitenbichler and M. Muhlhauser, "An IR local positioning system for smart items and devices", Proceedings of ICDCSW'03, pp. 334-339, 2003
- [4] ELPAS' LPS technology, <http://www.elpas.com/>
- [5] F. Vogt and S. Sidney Fels, "Tracking Multiple Laser Pointers for Large Screen Interaction", Extended Abstracts of ACM UIST 2003, pp. 95-96, 2003.
- [6] J. Caffery, and G. Stuber, "Subscriber Location in CDMA Cellular Networks", IEEE Transactions on Vehicular Technology, Vol. 47, No. 2, pp. 406-416, 1998.
- [7] J. Hightower and G. Borriello, "Location Systems for Ubiquitous Computing Computer, Issue on Location Aware Computing", IEEE Transactions on Computers, Volume 34 No. 8, 2001, pp. 57-66, 2001.
- [8] J. Krumm, "Multi-camera multi-person tracking for EasyLiving", Proceedings of 3rd IEEE International Workshop on Visual Surveillance, pp.3-10, July 2000
- [9] P. Bahl and V. Padmanabhan, "RADAR: An in-building RF-based user location and tracking system", Proceedings of IEEE INFOCOM, Vol. 2, pp. 775-784, March 2000.
- [10] RF-Technologies, PinPoint Asset Tracking Solutions, <http://www.rftechnologies.com/pinpoint/>
- [11] R. Want, A. Hopper, V. Falcao, and J. Gibbons, "The active badge location system", ACM Transactions on Information Systems, 10(1): pp.91-102, January 1992
- [12] Yasuyuki Sumi, T. Matsuguchi, S. Ito, Sidney Fels and Kenji Mase "Collaborative capturing of interactions by multiple sensors", UbiComp'03, pp. 193-194. October 2003.
- [13] Yasuyuki Sumi, Sadanoro Ito, Sidney Fels, et al. "Collaborative capturing and interpretation of interactions", Proceedings of Pervasive 2004 Workshop on Memory and Sharing of Experiences, pp. 1-7. April 2004.