Design of a Wireless Vision Sensor for Object Tracking in Wireless Vision Sensor Networks

Mauricio Casarest  
*University of Nebraska-Lincoln*, mcasare1@bigred.unl.edu

Mehmet C. Vuran  
*University of Nebraska at Lincoln*, mcvuran@cse.unl.edu

Senem Velipasalar  
*University of Nebraska-Lincoln*, velipasa@engr.unl.edu

Follow this and additional works at: [http://digitalcommons.unl.edu/cseconfwork](http://digitalcommons.unl.edu/cseconfwork)

Part of the [Computer Sciences Commons](http://digitalcommons.unl.edu/csesc/)

This Article is brought to you for free and open access by the Computer Science and Engineering, Department of at DigitalCommons@University of Nebraska - Lincoln. It has been accepted for inclusion in CSE Conference and Workshop Papers by an authorized administrator of DigitalCommons@University of Nebraska - Lincoln.
DESIGN OF A WIRELESS VISION SENSOR FOR OBJECT TRACKING IN WIRELESS Vision Sensor NETWORKS

Mauricio Casares†, Mehmet Can Vurant†, Senem Velipasalar†

†Department of Electrical Engineering
University of Nebraska-Lincoln, Lincoln, NE, 68588
E-mail: mcasare1@bigred.unl.edu, velipasa@engr.unl.edu

‡Department of Computer Science and Engineering
University of Nebraska - Lincoln, Lincoln, NE, 68588
Tel: (402) 472-5019 Fax: (402) 472-7767
Email: mcvuran@cse.unl.edu

ABSTRACT
The integration of CMOS cameras with embedded processors and wireless communication devices has enabled the development of distributed wireless vision systems. Wireless Vision Sensor Networks (WVSNs), which consist of wirelessly connected embedded systems with vision and sensing capabilities, provide wide variety of application areas that have not been possible to realize with the wall-powered vision systems with wired links or scalar-data based wireless sensor networks. In this paper, the design of a middleware for a wireless vision sensor node is presented for the realization of WVSNs. The implemented wireless vision sensor node is tested through a simple vision application to study and analyze its capabilities, and determine the challenges in distributed vision applications through a wireless network of low-power embedded devices. The results of this paper highlight the practical concerns for the development of efficient image processing and communication solutions for WVSNs and emphasize the need for cross-layer solutions that unify these two so-far-independent research areas.

Index Terms— Wireless Vision Sensor Node, Wireless Vision Sensor Network (WVSN) middleware, cross-layer design, foreground detection, background subtraction, salient motion.

1. INTRODUCTION
The recent advances in the CMOS technology have enabled the development of single chip camera modules that could easily be embedded into inexpensive transceivers. Wireless Vision Sensor Networks (WVSNs), have recently been the focus of research in a wide variety of areas including digital signal processing, communication, networking, control, and statistics [2],[11],[18]. The realization of WVSNs enable a wide variety of application areas, which were not possible with the scalar data-based wireless sensor networks (WSNs) [1], by incorporating real-time multimedia data. Most of the existing multi-camera systems require camera nodes to be wall-powered, and have wired links, which cause significant costs and difficulty in deployment. The multi-hop and ad-hoc
wireless communication in WVSNs, on the other hand, will provide enhanced capabilities to distributed vision cameras in terms of ease of deployment, improved coverage and reduced cost.

As shown in Fig. 1, WVSNs are envisioned to consist of heterogeneous components including wireless vision sensor nodes and wireless sensor nodes described as follows:

- **Wireless vision sensor node**: Embedded with a CMOS camera, a microcontroller and a wireless transceiver, these nodes are deployed in the sensor field to provide increased coverage for various vision applications. Through wireless communication with other nodes, a vision sensor node can exchange data with others to perform distributed video processing tasks such as object tracking.

- **Wireless sensor node**: Embedded with scalar sensors such as temperature, humidity, and acceleration sensors, these nodes provide additional information regarding the sensor field. Furthermore, wireless sensor nodes are deployed in higher density to provide connectivity between vision sensors in the WVSN via multihop communication.

Vision applications need to process large amounts of data in real-time, which requires high processing power and memory at the vision sensor nodes. To reduce the communication burden and increase the lifetime of the network, in WVSN, wireless sensor nodes are employed for multi-hop communication as shown in Fig. 1. This requires close coupling between the embedded camera and the transceiver in the wireless vision sensor node. Transmission of multimedia content has long been associated with multimedia source coding techniques because of the large amounts of traffic generated by the multimedia sources such as cameras. As mentioned previously, video processing tasks such as background subtraction, forming of foreground blobs, and object tracking demand high processing power and memory for real-time operation. Consequently, the design of a wireless vision sensor node requires a cross-layer approach where both the video processing and communication components are jointly designed.

While the necessity of a wireless vision sensor node capable of low-power and light-weight video processing, and wireless communication is clear, the recent work focuses on certain aspects of this paradigm. The Cyclops platform is presented in [15], which is developed as a sister-board for the Mica2 and MicaZ sensor boards and provides limited processing capabilities with a 7.3 MHz processor. The MeshEye platform [7] integrates three multi-resolution CMOS cameras to provide a surveillance camera module, which has a processor with 50 MHz speed. However, because of their complex local communications between devices, MeshEye can only support low frame rates in the order of 3 frames per second which can not be considered as a real time application.

On the other hand, Panoptes platform is developed to host a 206 MHz processor, WLAN card, and a webcam [6]. Panoptes enables more sophisticated processes to be performed in this high-end architecture leading to a high energy consumption of the node and reducing the lifetime of the wireless application. Furthermore, a smart camera architecture is developed from common off-the-shelf components including a CMOS image sensor, multiple stacked DSPs, and a GSM/GPRS communication to reach a high processing power with onboard memory of 784 MB in [4]. While this high-end platform also provides sufficient capabilities for image processing, it is expected to result in a large power consumption. Moreover, the use of a cellular network communication module significantly increases the energy consumption for communication and limits the applicability of the module for low-cost, high density applications.

A vision-enabled wireless sensor network, FireFly Mosaic, is presented in [8], and an assisted living application is demonstrated. It is stated that multiple people in the environment cause problems with activity transition probabilities, which suggests that multi-object tracking cannot be performed with this architecture. Our aim is to reach energy consumption values in terms of several watts while performing robust video processing tasks and multi-hop communication. It is clear that a low-power but yet powerful vision sensor architecture is yet to be developed for WVSNs.

In this paper, the design of a middleware for a wireless vision sensor node is presented for the realization of WVSNs. The vision sensor node consists of two common off-the-shelf components: CMUcam3 CMOS camera module [21] and a SunSPOT sensor module [22]. The integration of these two components are performed both in hardware and software through the developed middleware. Moreover, simple motion detection and location extraction are performed on the vision sensor node (on the processor of the CMUcam3) to study and test the capabilities of our vision sensor node, and determine the challenges in distributed vision applications through a wireless network of low-power embedded devices. The results of this paper highlight the practical concerns for the development of efficient video processing and communication solutions, and emphasize the need for cross-layer solutions that unify these two so-far-independent research areas.

The rest of the paper is organized as follows: In Section 2, the properties of the common off-the-shelf components used in our wireless vision sensor node are described. The hardware and software architecture as well as the middleware are presented in Section 3. The test application is described in Section 4. The experiment results are discussed in Section 5, and the future work is summarized in Section 6.
2. COMPONENTS OF THE WIRELESS VISION SENSOR NODE

Vision applications need to process large amounts of data in real-time, which requires high processing power and memory at the vision sensor node. In order to address this issue, we employ two efficient off-the-shelf components, namely CMUcam3 and SunSPOT, and make use of the 512 Kb RAM provided by the SunSPOT instead of just using it as a communication entity. Using off-the-shelf components also supports fast prototype implementation through middleware development. Next, we describe each individual component and its functionalities in the wireless vision sensor node. A summary of the hardware specifications of these components is presented in Table 1.

• CMUcam3: CMUcam3 is a fully programmable embedded vision sensor [21]. It is equipped with C libraries, which provide software for embedded vision systems. The software runs on the CMUcam3 hardware platform, which is a NXP LPC2106-based board with an onboard CMOS camera and FIFO, MMC interface, and serial UARTs.

• SunSPOT: Sun Small Programmable Object Technology (SPOT) is a recent wireless sensor platform developed by Sun Microsystems, Inc., which consists of a ZigBee-capable wireless transceiver, a microprocessor, accelerometer, temperature sensor, light sensor and several general purpose I/O ports [22]. The SunSPOT runs on a Java Virtual Machine (VM), which provides basic operating system functionality. The SunSPOTs use a fully capable Java ME implementation, called Squawk that supports CLDC 1.1 and MIDP 1.0, plus provides basic OS functionality. The VM executes directly out of flash memory. All the device drivers are also written in Java.

The integrated CMUcam3 and SunSPOT constitutes our wireless vision sensor node. We performed a test application on this node as described in Section 4.

3. DESIGN AND IMPLEMENTATION

In this section, the hardware architecture and the middleware implementation of the wireless vision sensor node is
presented. First, in Section 3.1, we describe the hardware interconnection-between the CMUcam3 and the SunSPOT. Then, in Section 3.2, the middleware implementation, which governs the serial communication between the two components as well as the wireless communication between a wireless vision sensor node and a wireless sensor node, is presented.

3.1. Hardware Implementation

The hardware integration of the SunSPOT and the CMUcam3 are performed through the I/O connector pins of each platform as shown in Fig. 2. SunSPOT is equipped with a 20-pin expansion connector for external data transfer. A 5V level is generated on the sensor board (left side of Fig. 2) which means that the Serial communication between both devices (SunSPOT - CMUcam3) has to be in a TTL platform.

The corresponding inter-board connector for the CMUcam3 is also depicted on the right side of Fig. 2. The CMUcam3 has two ports for serial communication through either RS232 (-12 to 12 Volts Level) or TTL (0 to 5 Volts Level). Since SunSPOT also supports TTL, the TTL serial port corresponding to the CMUcam3 is used as shown in Fig. 2. Note that there is a Serial Bypass Jumper which has to be taken off the moment the Camera is attached to the SunSPOT to allow a TTL platform. However, this jumper is placed back whenever the microcontroller of the CMUcam3 needs to be programmed.

Finally, the hardware interconnection between SunSPOT and CmuCam3 is depicted in Fig. 2. The pin number 4 (D0) in the SunSPOT’s interconnection board (left side of Fig. 2) corresponds to the Serial Reception Pin (RX) while the pin number 6 (D1) corresponds to Serial Transmission Pin (TX). On the other hand, the pin Logic Out STX from CMUcam3 depicted in right side of Fig. 2 is the Serial Transmission Pin in the CMUcam3, while Logic In SRT pin from CMUcam3 shown in Fig. 2 corresponds to the Serial Reception Pin in the CMUcam3. Furthermore, the Transmission pin (STX) from the cmucam3 has to be connected to the Reception pin (D0) in the SunSPOT. Reciprocally, the Reception pin (SRX) from the CMUcam3 has to be connected to the Transmission pin (D1) in the SunSPOT.

3.2. Middleware

In this section, the description of the developed middleware, which supports the data transfer between CMUcam3 and SunSPOT as well as the wireless communication between the wireless vision sensor node and the Base station is provided. The developed middleware interacts between application programs APIs which can use the serial port as an object to be shared between the SunSPOT and the CMUcam3 software components. Next, the components of the middleware are described.

J2ME \(^1\) uses a Generic Connection Framework (GCF) to create connections (such as HTTP, datagram, or streams) and perform I/O [22]. To perform wireless communication, the J2ME provides two GCF-based radio communication protocols, Radio Stream and Radio Datagram. Both of those protocols perform a reliable point-to-point communication between two SunSPOTs.

A Radio Stream Connection java class is used to send and receive data between the WVSN and the Base Station in which a Radio Stream protocol was used. The radio protocol is a socket-like peer-to-peer protocol that provides reliable, buffered stream-based I/O between two devices. More specifically, this protocol opens a specific channel between the two devices using their MAC addresses. Moreover, the same Port Number is shared and the destination addresses correspond to both of the elements that are going to be involved in the communication (SunSPOT and Base Station MAC addresses). Once the connection is established, each communication end (WVS and Base Station) can obtain streams to use to send and receive data.

It is important to note that SunSPOTs are configured in such a way that every data transmission between the two devices involves waiting for an acknowledgment, which is a function of the underlying MAC layer. The NoAckException is thrown if this acknowledgment doesn’t arrive, despite retrying.

**Serial Communication** between the SunSPOT and CMUcam3 is a major component of the middleware implemented on WVSN. Although two components are based on different software platforms, i.e., JAVA and C respectively, if the correct parameters are set in both of the devices, the serial communication can be performed. To provide communication between the SunSPOT with the CMUcam3, the Standard Serial Protocol (UART) is used. If the amount of data is considerably large, then, a flow control algorithm will be implemented to avoid overflow. Although the maximum communication rate of the CMUcam3 is 115,200 bauds, it was set to 38400 bauds since this is the maximum rate that SunSPOT can support. In addition, since the data transmission corresponds to a single integer, which is the centroid

---

\(^1\)J2ME refers to Java 2 Micro Edition (J2ME) which is Sun’s version of Java aimed at machines with limited hardware resources such as PDAs, cell phones, and other consumer electronic and embedded devices.
of the moving object, we decided to constantly check the buffer on the SunSPOT to see if the CMUcam3 has sent any information.

In Fig 3, the required tasks for the Wireless Vision Sensor is shown. At the beginning of each process, both the SunSPOT and the CMUcam3 are initialized. Each component on both boards related to the specific process is initialized. Then, two different processes run in parallel over two different platforms Java VM and C. The SunSPOT first looks for the Base Station to see if it is its radio range. Since, the WVS is aware of the MAC address of the Base Station, it will keep trying until the connection is possible or a timer expires. If the Base Station is in its range a handshake is performed by opening a channel between the devices involved in the communication. The Radio I/O stream is then used as a buffer to receive the information from the CMUcam3 and transmit this information to the Base Station. Parallel to this process, in the CMUcam3, the background subtraction is performed in which the segmentation is done by applying temporal differencing. After performing the segmentation, the centroid and the location of the object is estimated. This process is repeated for each frame to provide real-time feedback and prevent losing the moving object.

Once the centroid is found, the middleware notifies the SunSPOT and initiates communication between the CMUcam3 and the SunSPOT. The data that is to be sent is encapsulated to a serial configuration and sent to the SunSpot, which relays this information to the Base Station through the wireless channel.

The resulting wireless vision sensor node is shown in Fig.
4(a) and Fig. 4(b). It can be seen that the wireless vision sensor node has a small form factor compared to the state-of-the-art smart camera solutions and provides high processing as well as wireless communication capabilities.

4. TEST APPLICATION

In order to study and test the capabilities of our wireless vision sensor node, and determine the challenges in distributed vision applications through a wireless network of low-power embedded devices, we performed simple motion detection and location extraction on the vision sensor node, more specifically on the processor of the CMUcam3. Figure 5 shows the processes that are performed in the wireless vision sensor node.

First, temporal difference between consecutive frames is found, and the centroid of the detected motion pixels is computed. Currently, it is assumed that there is only one object in the environment to simulate a tracking application, and to use the centroid locations as the tracking outputs. The CMUcam3 places the centroid location into a buffer. The SunSPOT reads this location from the buffer, and sends it wirelessly to the base station, which is connected to a PC. Figure 6 shows two consecutive frames and the detected centroid location that is broadcasted by the SunSPOT.

Although many methods have been proposed for automatic moving detection or foreground detection [3] [9], [10], [12], [13], [14], [16], [17], [20], much less attention has been paid to the memory requirements and computational complexity of the algorithms. For instance, the memory requirements of the state of the art background subtractions algorithms exceed the 64Kb RAM provided by the CMUcam3. We designed a light-weight foreground detection algorithm [5], and started to import it to the processor of the CMUcam3.

5. RESULTS AND CONCLUSIONS

In this paper, we focused on the design of a wireless vision sensor node (WVSN) to be used in a wireless sensor network. We built the WVSN by integrating a CMUcam3, which as discussed in Sections 5 and 6.

This application showed the necessity of communicating between the CMUcam3 and the SUNSpot and using the SUNSpot for data processing as well instead of just as a communication entity.
Frame (t-1)  Frame (t)  Centroid

Fig. 6. Detected Centroid

This is a fully programmable embedded vision sensor, with a SunSPOT, and by implementing a middleware for communication and process distribution between the two components. We also tried to import a background subtraction algorithm to the processor of the CMUcam3. This whole process presented us with some challenges as described below:

- **Limited Processing Power and Memory**: The memory requirements of the state of the art background subtraction algorithms are much higher than the memory provided by the CMUcam3, which uses the NXP LPC2106 microcontroller. The LPC2106 is a 32-bit, 60 MHz ARM7TDMI with 64 Kb of built-in RAM. As expected, 64 Kb was not enough to import a sophisticated foreground detection code, [10]. We started to implement a light-weight foreground detection algorithm [5] on the CMUcam3 processor, and this algorithm is expected to use up all the limited memory. However, foreground detection by itself is not enough to track objects of interest, since it only gives us foreground pixels. Some morphological operations are needed to clean the output of the foreground detection algorithm. After this step, the remaining pixels need to be grouped into different objects to form foreground blobs. Then, corresponding blobs need to be found in consecutive frames to track the objects. All these steps require additional memory. In order to address this issue, a more sophisticated middleware application will be built as discussed in more detail in Section 6. After performing foreground detection, the remaining processing jobs will be delegated to the Sun SPOT, which has a 512Kb of RAM. Thus, Sun SPOT will be used for processing video data instead of just working as a communication entity.

- **Heterogeneity**: Another challenge was the different programming languages used by the CMUcam3 and the Sun SPOT, which use C and Java, respectively. Currently, we place the output obtained in the CMUcam3 into a buffer from where the SUN Spot retrieves it. This also introduces a potential overhead problem if large amounts of data is placed in the buffer.

Since our WVSN is a combination of two embedded systems with two different processors and different hardware configuration, a middleware layer is required between these components to exchange data. This handshaking procedure was introducing considerable overhead during the transmission causing the drop of frames. The SunSPOTs are by default configured in a way that every data transmission between the two devices involves waiting for an acknowledgement. In our experiments, we transmitted the centroid of an object, detected on the CMUcam3, to the Sun SPOT. First, a temporal difference method is applied to detect motion pixels, and then the centroid of the detected pixels is found. The frame rate of this process was 12.5 frames per second.

As stated previously, we will use the additional memory provided in the SunSPOT to perform parts of the steps required for object tracking. Yet, this is expected to introduce some overhead causing the frame rate to decrease. Thus, it is important to carefully decide what data to transfer and how often to transfer it to the Sun SPOT.

- **Wireless Communication**: The limited bandwidth, which is 250 kbps for the SunSPOTs, restrict the amount of information that can be transmitted between vision sensors. Hence, transmitting whole frames between sensor nodes is both time consuming and may violate the real-time requirements of distributed vision applications. Moreover, sending whole frames between vision sensors may not actually be necessary for most distributed image applications. Consequently, the communication protocols need to be tailored to the requirements of image processing algorithm necessitating a cross-layer approach for efficient operation.
6. FUTURE WORK

One of the major tasks will be sharing the processing tasks between the CMUcam3 and the Sun SPOT. Since the SUNSPOT is capable of performing higher amount of processing and has a much larger RAM(512Kb) compared to the CMUcam3 (64Kb), the image processing tasks can be distributed between two platforms to enable more sophisticated operation and more tasks may be assigned to the Sun SPOT. We will apply the same ideas and the structure described above. After importing our light-weight foreground detection algorithm to the CMUcam3, we will send the salient foreground pixels to the Sun SPOT. This will provide advantages in terms of overhead compared to transferring the whole frame. Furthermore, the communication protocol on SunSPOT will be tailored to the communication requirements of the object tracking application.

As stated in Section 1, in our experiments, the temporal difference is found, and an integer corresponding to the centroid of the detected object is sent to the SUNSPOT module, which relay this information wirelessly. The frame rate of this procedure is 12.5 frames per second. When we start to send more data, it will be necessary to analyze the tradeoffs between accuracy and real-time processing as well as multi-hop communication, providing a detailed report of the overhead caused by data transfer between the CMUcam3 and the Sun SPOT. Then, we will implement a more sophisticated detection and tracking algorithm on our WVSN. Figure 7 depicts what is planned to be implemented on the vision node.

Since one visual sensor has a limited field of view, it cannot provide enough coverage. Many applications, including surveillance, require larger coverage and better accuracy. Using multiple vision sensors provides robustness and fault tolerance against issues such as target occlusion and sensor failures.

However, if many of this WVSNs are used, a sophisticated communication protocol needs to be designed to reach the goal of real-time processing by conserving energy and with an acceptable accuracy. Conserving energy in a wireless sensor network is crucial, since individual nodes are not going to be wall-powered. Energy concern is even more pronounced when the sensor being used is a vision sensor (a camera). Since every node is planned to run on batteries, direct communication (with a single hop) is not the best option to establish communication between WVSNs. Hence, we will place multiple simple nodes, consisting just one Sun SPOT (see Fig. 1) between the WVSNs to accomplish a wireless multi-hop communication. Multi-hop communication is expected to increase the overall lifetime of the wireless vision sensor network. The field of view information obtained from the vision nodes, and the proximity of cameras will be used to decide the optimal path or route to be taken to perform the communication between nodes.

7. REFERENCES


[22] SunSpot, http://www.sunspotworld.com