

A Mobile Vision System with Reconfigurable Intelligent Agents

Yan Meng, *Member, IEEE*

Abstract — Performing face detection and tracking on a mobile robot in a dynamic environment is a challenging task with the real-time constraints. To realize a natural reactive behavior of the mobile robot, some efficient algorithms are applied for face detection and tracking due to the limited on-board computation power. Instead of pushing the limit of software development and the computational resources by implementing these algorithms on a software platform, a reconfigurable System-on-Chip (SoC) platform is adopted to achieve the higher real-time performance. A BDI intelligent agent model is proposed as a unified structure for both hardware and software. The real-world experimental results on a mobile vision system for face detection and tracking demonstrate the feasibility and efficiency of the reconfigurable intelligent agent model.

I. INTRODUCTION

EXTENSIVE research has been conducted on face detection and tracking in computer vision areas. However, most computer vision approaches have underlying assumptions, such as large memory, high computation power, static vision system, or off-line processing. When a vision system is on a mobile platform, there are some constraints which are specific compared to the static vision system. First, the system has to be highly robust to adapt to the changing illumination and environmental structure. Furthermore, the real-time processing is very important to realize a natural reactive behavior and prevent any serious damages. With the limited processing power and computational constraints of a mobile robot, it is difficult to apply those complicated algorithms to a vision system on a mobile platform. Therefore, a less general and complicated but simple and efficient algorithm for face detection and tracking is desirable. However, with the simplified approaches, we may have to compromise with the abated detection and tracking performance or the reduced image resolution.

Furthermore, a mobile robot system consists of not only a vision system but also the navigation and control systems. All of these tasks have to be processed on the on-board processor of a robot under the real-time constraints, which adds more burdens on the already limited computational power. A software solution to this kind of real-time multi-tasking environment leads to a very conservative computation solution even with a far more powerful processor.

Therefore, we turn our attention to hardware solutions. The hardware can respond to external inputs more efficiently since the multiple hardware units can all operate

in parallel, concurrent, and asynchronous, so that individual response times are much less variable and easier to guarantee, even as the number of tasks increases. Parallel hardware is not so affected by issues such as task swapping, scheduling, interrupt service, and critical sections, which complicate real-time software solutions. This feature makes the hardware platform appealing to real-time systems, such as a mobile vision system. However, these specific hardware devices are usually very expensive to develop and lack of flexibility.

Reconfigurable computing is intended to fill the gap between hardware and software, achieving potentially much higher performance than software, while maintaining a higher level of flexibility than hardware. Reconfigurable devices, including field-programmable gate arrays (FPGAs), contain an array of computational elements whose functionality is determined through multiple programmable configuration bits. The advances of FPGA technology make the hybrid hardware-software architecture of reconfigurable system-on-chip (rSoC) feasible in implementation, which can provide both runtime flexibility and real time performance for high-level applications. These two features are not simultaneously achievable by traditional pure software or hardware implementations.

Based on this rSoC platform, a Belief-Desire-Intention (BDI) agent model is proposed in this paper as a unified structure for both the hardware and software, which is intended to design high-level aspects of systems and to simplify the hardware/software partitioning and communication. The system is first decomposed into agents based on system specifications, where these agents can accomplish some specific tasks independently and can communicate with each other. These agents are then partitioned into software agents and hardware agents based on the heuristic approaches. On top of this unified agent-based representation, it is possible to provide a generic communication mechanism between the hardware and software agents.

Compared to the regular module-based architecture, the BDI agent model can provide more flexibility, intelligence, autonomy, and scalability. By applying this agent-based model into a rSoC platform, which consists of two soft core processors and dynamic reconfigurable hardware logic units on FPGA, the overall system flexibility and efficiency will be significantly improved, although some cost has to be paid on the reconfiguration procedure.

This intelligent agent-based model can be easily applied to a more general real-time embedded system with limited computation resources. This paper also serves as a case study demonstrating the practical implementation of developing an effective real-time mobile vision system.

Y. Meng is with the Department of Electrical and Computer Engineering, Stevens Institute of Technology, New Jersey 07030, USA (Phone: 201-216-5496; fax: 201-216-8246; email: ymeng1@stevens.edu).

II. RELATED WORK

Most of the previous work on reconfigurable embedded platform for real-time systems is implemented on software only [12, 2]. The recent advent of dynamic reconfigurable FPGA platform attracts more attentions in large applications. Most of the real-time systems use FPGAs to speed up the portions of an application that fail to meet required real-time constraints. The approach of applying reconfigurable logic for data processing has been demonstrated in some areas such as video transmission, image-recognition and various pattern-matching operations [14] [15]. In [9], the system uses FPGAs that are dynamically reconfigured on programmable gate arrays at runtime to implement different functions.

As the soft processor cores for FPGAs become available in the rSoC platform, the hardware/software co-design issues becomes critical in the overall system design procedure. Some recent hardware/software co-design researches use reconfigurable processors as the platform for implementation [15, 3], where their architectures combine a reconfigurable functional unit with the microprocessor. Partitioning is a well-known NP-complete problem and many heuristics-based approaches have been proposed to guide the partitioning of a system into hardware and software components. In [21], the hardware/software partitioning problem is modeled as a Constraint Satisfaction Problem (CSP), and a genetic-based approach is proposed to solve the CSP to obtain the partitioning solution.

III. FACE DETECTION AND TRACKING ALGORITHMS

Some researches have been conducted for people detection systems using one single cue at the detection, such as motion [19], color [18], contour [5], or face's features (such as eyes, nose, and mouth, etc.) [20]. Due to the natural, complex, and dynamic working environment of a mobile robot, the robot vision system has to be highly robust and independent with the application scenario. Therefore, some researches proposed the combinations of multiple cues for people segmentation [8] [11] [22]. In this paper, we propose a detection method combining the color and motion cues.

First, skin-color classification is applied to find faces in images since it is independent from ego-motion of the camera systems. To represent skin color, the dichromatic r-g-color space ($r=R/(R+G+B)$, $g=G/(R+G+B)$), which is normalized in brightness and thus is widely independent from variations in luminance. To improve the system real-time performance, we have built up a look-up table with manually classified skin color pixels in the r-g-color space for 30 people under different illumination conditions.

Then, motion detection is applied to distinguish the people moving in the scene from the stationary background. For a stationary camera system, motion detection can be conducted by subtraction of two successive image frames. However, if the camera system is moving, the ego-motion has to be taken into account. The algorithm proposed in [17] is applied to estimate the compensation for the ego-

motion of the mobile robot. Assume the robot motion parameters can be obtained from the navigation and control systems. The next successive image can be predicted by using the epipolar transformation. The predicted image is matched with the actual image from the scene, and all non-matching areas represent the moving objects.

It may occur frequently that multiple Region of Interests (ROIs) are detected in one image, where a fusion algorithm is necessary to merge those ROIs. To simplify the merging procedure, we assume that there are relatively small changes in size and position of people from the successive image frames, in other words, the people is moving in a relatively slow speed. Then a probabilistic approach based on the color and motion cues as well as the geometric parameters of the object in the images is applied to select the most likely ROI.

The condensation tracking algorithm in [13] is adopted here for face tracking. The task of calculating the probability of the presence of a face for every pixel and tracking the resulting density function over time is solved by an approximation of the density function. Then, a recursive filter is applied to update the density function on each sample. Once a person is tracked, the samples of the condensation algorithm are concentrated on his/her face.

As we can see, it is extremely computation extensive for the image matching, face detection, and face tracking from the above approaches. To achieve the real-time performance without losing too much image resolution, an agent-based self-reconfigurable system-on-chip platform is proposed in this paper and described in the following sections.

IV. THE BDI AGENT MODEL

An *agent* is an independent processing entity that interacts with the external environment and the other agents to pursue its particular set of goals. The agent pursues its given goals adopting the appropriate plans, or intention, according to its current beliefs about the state of the world, so as to perform the role it has been assigned. Such an intelligent agent is generally referred to as a Belief-Desire-Intention (BDI) agent. In short, belief represents the agent's knowledge, desire represents the agent's goal, and intention lends deliberation to the agent. The agent control loop is: first determine current beliefs and goals, then find applicable plans (or intentions) and decide which plan to apply, and finally start executing the plan.

In the agent model, as shown in Fig. 1, beliefs and desires influence each other reciprocally. Furthermore, beliefs and desires both influence intentions. The model includes three external ports: inter-agent communication, control, and input/output. The control port is for the agent to synchronize with the system. The input/output port is used to send and receive information to and from the host environment. The inter-agent communication port allows the agents to send/receive information with other agents and cooperate with others.

The BDI agent is capable to provide the following features which are impossible for the regular module-based architecture.

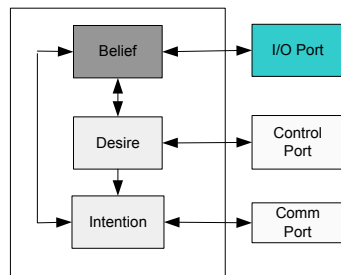


Fig. 1. The BDI agent model

- **Autonomy.** Once launched with the information describing the bounds and limitations of their tasks, BDI agents are able to operate independently of and unaided by their use.
- **Social ability.** To effect changes or interrogate their environment, BDI agents possess the ability to communicate with the outside world through their communication ports.
- **Reactivity.** BDI agents are able to perceive their environment and respond to changes to it in a timely fashion.
- **Proactivity.** To help BDI agents to be adaptive to new situations, they are able to exhibit proactivity, that is, the ability to effect actions that achieve their goals by taking the initiative.

Some agent-oriented implementations have been developed, such as JACK [1] and distributed multi-agent reasoning system (dMARS) [6]. In our systems, the software agent follows the similar idea of JACK. The BDI agents with beliefs, desires, intentions, and plan library are created. Events alert the agent to changes in the world or in its internal state. When an event is raised, the agent looks through its plan library and finds a plan that is relevant to the event and its current beliefs. It then creates an instance of this plan and tries to execute it, however if it fails, another plan is tried until all relevant plans are exhausted.

The hardware agent implementation is similar to the software agent except that in software, all the parameters of the agents can be transmitted by function calls in C/C++, while in hardware we have to specifically define all of the hardware agent entities, their associated ports and parameters in VHDL. The hardware agent interface should be simple for customization to any specific applications without significant rework.

V. HARDWARE/SOFTWARE CO-DESIGN

Usually a complex real-time system, such as a mobile robot with real-time vision systems, can be constructed as a set of independent and cooperating agents, where each agent owns its own intention and pursues its own sets of goals. Each agent can communicate with each other through their inter-agent communication ports, interact with external environment through their I/O ports, and control signals through control ports. Each agent may include multiple

tasks, where these tasks may or may not be independent to each other. To achieve high performance of overall system, effective hardware/software co-design and communication between agents is required.

A. Hardware/software Co-design

Hardware/software co-design provides a way of customizing the hardware and software architectures to complement one another in ways which improve system functionality, performance, reliability, survivability, and cost/effectiveness. To this end, first issue to be solved is the hardware/software partitioning. It is desirable to propose a method which can automatically partition the agents into hardware and software on the fly based on the task parameters on each agent, such as execution time, release time, and deadline. However, automatically partitioning is a challenge problem not only because it is a well-known NP-complete problem, but also due to the lack of efficient and accurate performance profiling tools. Furthermore, the dynamic partitioning is complex since it has to handle some sporadic tasks, for example, the environment changes or a new obstacle is detected. To simplify the problem, an off-line heuristic partitioning method is applied in this paper.

In general, the following partitioning policies are required to follow. First, more regularly structured agents that have highly repetitive and extensive time consuming operations are suitable for implementation in reconfigurable hardware, whereas the more complex and irregularly structured agents should be programmed in software. Second, due to the different implementation platforms of hardware and software, the agents who have heavy communication or dependencies should be all partitioned either to hardware or software to minimize the communication costs. Third, the agents with hard deadlines are distributed to hardware, while the agents with soft deadlines may be implemented in software.

B. Multi-agent Communication Mechanism

Various agent communication languages have been developed, such as Knowledge Query Manipulation Language (KQML) [1], and Foundation for Intelligent Physical Agent (FIPA) [6]. But they have high overheads in terms of memory usage and computation time that are not always acceptable for real-time systems. Numerous proprietary methods also have been developed, but more and more often they encounter maintenance, porting, and enhancement issues.

A novel protocol for multi-agent communication has been proposed in our previous work [16], which provides (1) on-demand message queue (ODMP) protocol to satisfy the real-time constraints and asynchronous properties; (2) provide transport independence, allowing distributed agents to effectively exchange data over any transport, eliminating custom requirements for communication hardware. Therefore, it is portable to different agent-based real-time systems without significant modification to the both

software and hardware. The only parts need to be customized are adding different transport-dependant adapters to new systems.

VI. A RECONFIGURABLE PLATFORM

Since Vertex Pro-II FPGA is adopted as our protocol platform, the internal reconfiguration access port (ICAP) in Vertex Pro II FPGA makes the self-reconfiguration possible. A self-reconfiguration framework which has been proposed in our previous work [16] is adopted here. The goal of this framework is to provide a flexible method for implementing field self-reconfiguration while minimizing both system hardware and required configuration data, and reducing the reconfiguration time. The fixed logic includes common fixed logic circuits, ICAP, configuration controller, multiple dual-port block RAMs (BRAMs), embedded processors, and external memory. The embedded processor provides intelligent control of device reconfiguration at runtime. The embedded processor communicates with FPGA logic through IBM-designed CoreConnect bus architecture. The DDR external memory is connected to the processor through process local bus (PLB bus).

The fixed logic part is located the right-most columns because ICAP in Vertex II Pro FPGA is located at the lower-right corner while the reconfigurable logic part is located on the left-side columns. The configuration controller uses the ICAP to reconfigure the reconfigurable logic parts. The dual-port characteristics allow the BRAM to be accessed from different clock domains on each side. In this self-reconfiguration platform, one port is accessed from the configuration controller, while the other port is accessed from the reconfiguration logic part. This makes the BRAMs act like a shared memory between the processor and reconfiguration logic parts, which can be further applied as buffers to reduce the reconfiguration latency.

Usually, there are two trigger conditions for the reconfiguration, one is from bottom-up, and the other is from top-down. The bottom-up method is through event interrupts invoked by the sensor agents. The top-down method is that the processor makes the decision due to the input information from sensor management agent. Both methods are applied in our platform.

It is worth noting that the multi-agent communication mechanism can easily take advantage of the shared memory feature provided in self-reconfigurable platform. Due to the dual-port characteristics of the BRAM, the software agents can send messages to BRAM through configuration controller, while the hardware agents can access their own message queues directly from the same BRAM.

To take full advantage of the self-reconfigurable platform of hardware, the following approaches are proposed to improve the efficiency of image processing procedure for the face detection and tracking so that a real-time performance can be achieved.

First, image matching is critical for a mobile vision system since face tracking can only be conducted through multiple views. It is possible to make the image loading and image

matching to execute at the same time. This parallel scheme can reduce the implementation time dramatically since the image loading from external memory to reconfigurable hardware is usually time consuming. Fig. 2 shows this architecture where two buffers created, one is for loading and one for matching. When the image is loaded onto the buffer from I/O, the other buffer is applied for high-speed matching algorithm. When the image loading is finished, the control logic will check the status of another buffer. If the matching is done, the two buffers exchange operations. Otherwise, the control logic will inform the loading buffer to wait till the other buffer finishes the matching, and then exchange the operations.

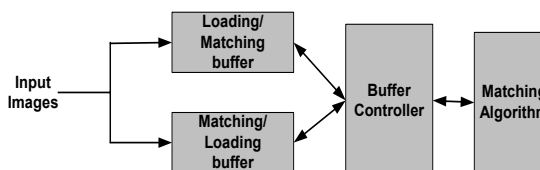


Fig. 2. A reconfigurable architecture for image matching

Second, traversing the image is necessary to identify the ROIs during the face detection. To speed up this traversing procedure, each loaded image can be divided into several regions, where each region can be assigned to different FPGA logic block. A separate control logic is responsible for whole image information integration. By paralleling the searching procedure, the overall speed of the image processing can be significantly reduced. Since the FPGA blocks are limited for each SoC platform, the dynamic reconfiguration is conducted so that the new image regions will be continuously loaded into each block after the previous block has been processed.

If some of the regions finish their traversing faster than others, it is possible that new image regions can be loaded in the finished regions to start processing. Therefore, a pipeline scheme is proposed to improve the processing efficiency by using the FPGA logic blocks greedily.

VII. AGENT-BASED SYSTEM ARCHITECTURE OF A MOBILE VISION SYSTEM

The overall agent-based architecture for a vision system on a mobile robot system is shown in Fig. 3, where rectangles represent hardware agents and ovals represent software agents. Since robot path planning agent and robot/human interface agents are complex and not critical in real-time reaction, they are partitioning into the software agents, where all of other time-critical and extensive time consuming agents, such as image capture, image processing, face detection, face tracking, camera control, laser and sonar sensor, sensor fusion, and robot actuator control, are assigned to hardware agents. Unlike the other hardware agents which are located on the reconfigurable logic blocks, the reconfiguration control agent locates on the fixed logic block, which needs to control the reconfiguration procedure of the other hardware agents.

First, the camera captures the image from the environment, and sends the images to the image filtering agent. After the color calibration agent finishes its processing to remove the illumination affects, the sensor fusion agent combines the image information with the laser agent and sonar agent. The output of the sensor fusion agent can serve for face detection agent, face tracking agent, obstacle avoidance agent, and robot motion planning agent. The face detection and tracking agent will control the movement of the PTZ camera to keep the target inside its field of view. It is worth noting that at any given time, the obstacle avoidance agent and motion planning agent can work in a complementary mode, which means that only one agent can work at a time. If there is obstacles are detected, obstacle avoidance agent works, otherwise, motion planning agent works, then the motion control signal will be sent to robot actuator agent by the working agent. Since the face detection and face tracking agents are all configured in the hardware FPGA, they can be implemented concurrently, even with the obstacle avoidance agent, which means that the vision system can keep detecting and tracking a face while the robot is avoiding some dynamic obstacles on its way. The proposed reconfigurable schemes are applied to the face detection agent and the face tracking agent to improve the efficiency. This concurrent feature improves the system responsiveness to the dynamic environments significantly.

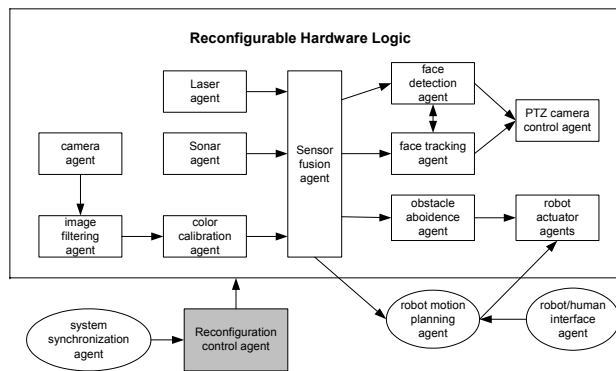


Fig.3. Agent-based architecture for a vision system on a mobile robot

VIII. EXPERIMENTAL RESULTS

The Xilinx ML310 embedded development platform, as shown in Fig. 4 (a), is installed to a mobile robot, Pioneer 3DX, as an on-board processor. ML310 offers designers a Virtex-II Pro XC2VP30-based embedded platform. It provides 30k logic cells, over 2,400kb BRAM, and dual PPC405 processors, as well as onboard Ethernet MAC/PHY, 256M DDR memory, multiple PCI slots, and standard PC I/O ports. A Pioneer 3DX mobile robot, as shown in Fig. 4(b), have a Pan-tilt-zoom Canon camera, laser range finder, and sonar sensors on board.

By using ML310 embedded platform, hardware agents are configured as fixed or reconfigurable FPGA logic circuits in VHDL, where bus macros are used as fixed data

paths for signals going between reconfigurable logic parts and other logic parts (fixed or reconfigurable). Software agents are implemented on the embedded soft cores PowerPC 405 in C/C++.

The laser and sonar data are fused to obtain the distance of people or obstacles around the robot. A spatial low pass filtering of adjacent measurements and a temporal low pass filtering of successive measurements is applied to reduce the influence of noise.

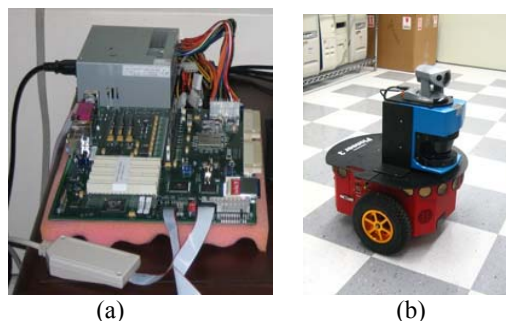


Fig. 4. (a) ML310 embedded platform; (b) Pioneer 3DX robot

In a dynamic environment, it is hard to compare the face recognition parameters due to the high dependence of the algorithm on the scene. To evaluate the proposed agent-based platform, the proposed face detection and tracking algorithm is implemented on the Xilinx ML310 embedded development platform in an indoor office environment. When the robot is moving, the processing frequency of 15 frames per second at an image resolution of 320x240 pixels is conducted in the experiment. Usually it is difficult for a moving robot to detect a fast moving object. Therefore, we assume that the velocity of the people's motion is up to 0.5m/s. A selected 1-minute sample image sequence is selected from the indoor office, where one person is moving around and facing to the vision system on the moving robot. There are some obstacles dynamically sitting on the way between the vision system and the person. Before the robot detects the face and tracks the face, since the robot has no idea where the person is, therefore, random movements are initiated until the vision system detects the face, then the robot moves towards the face while controlling the PTZ camera to tracks the face afterwards.

The detection rate with a selected sample image sequence is 85.54%. The Table I shows the resource usages of the FPGA hardware by implementing the face detection and face tracking algorithm proposed in Section III.

To evaluate how much performance to be gained by the hardware/software co-design platform compared to the software only platform, software implementation is also conducted on the on-board 1.6MHz COBRA computer for face detection and tracking.

The detection time is defined as the time difference from the initial time to the time when the face is detected. The experimental results are shown in Fig. 5. The average detection time for software-only is 33.05msec, while the average time for hardware/software co-design approach is

8.375msec. It is obviously that the co-design approach can improve the system performance significantly in its responsiveness and fault tolerance for real-time systems, especially in the situations where the environments are dynamically distributed by some obstacles.

TABLE I
RESOURCE USAGES OF HARDWARE AGENTS

Resources	Utilization
I/Os	43.41%
Function Generators	81.427
CLB slides	92.32%

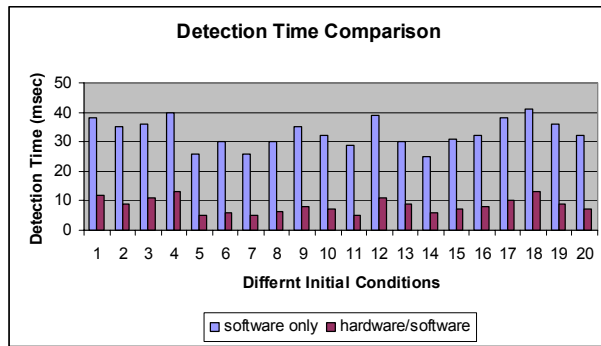


Fig. 5: Detection time comparison

IX. CONCLUSION

This paper proposes an intelligent agent based reconfigurable architecture for a real-time mobile vision system, where a case study of face detection and tracking is conducted on a Virtex II Pro FPGA platform to evaluate the proposed approach. The unified BDI agent structure significantly simplifies the hardware/software partitioning and communication. Some reconfiguration schemes are proposed to improve the hardware efficiency on a self-reconfiguration platform. Some efficient face recognition algorithms are implemented on the proposed rSoC platform. Experimental results of applying the agent-based architecture to a real-time face detection and tracking task on a real-world mobile vision system show that the proposed architecture is feasible and much more efficient compared to the software only approach.

The proposed BDI agent-based architecture provides a very flexible platform for the future extension, such as learning capability and proactivity to adapt to the dynamic environments. Our future research will focus on dynamic hardware/software partitioning and scheduling instead of just using an off-line heuristic method. In addition, we will also investigate the learning capability of the BDI agent architecture to make the overall system be more robust and adaptive to the environmental changes.

REFERENCES

[1] Agent-Oriented Software Pty Ltd, JACK Manual (v4.1), www.agent-oriented.com, 2003.

[2] J. Almeida, M. Wegdam, M. Sinderen, and L. Nieuwenhuis. Transparent Dynamic Reconfigurable for CORBA. *Proceeding of the 3rd International Symposium on Distributed Objects & Applications (DOA 2001)*, Sept. 17-20, 2001, Rome, Italy.

[3] M. Baleani, F. Gennari, Y. Jiang, Y. Patel, R. K. Brayton, and A. Sangiovanni-Vincentelli. HW/SW Partitioning and Code Generation of Embedded Control Applications on a Reconfigurable Architecture Platform. In *Proceedings of the Tenth International Symposium on Hardware/Software Codesign*, May 2002.

[4] B. Blodget, P. James-Roxby, E. Keller, S. McMillan, P. Sundararajan, "A self-reconfiguring platform", *Proceeding of the 13th International Conference on Field Programmable Logic and Applications (FPL'03)*, 2003, pp. 565-574.

[5] J. Davis and V. Sharma, "Robust Detection of People in Thermal Imaging," 17th International Conference on Pattern Recognition (ICPR'04), 2004, pp. 713-716.

[6] M. d'Inverno, D. Kinny, M. Luck, and M. Wooldridge, "A formal specification of Dmars", *Proceedings of the 4th International Workshop on Agent Theories, Architecture, and Language*, vol. 1365 of LNAI, Berlin, 1998, pp. 155-176.

[7] S. E. Fahlman, and C. Lebiere, "The cascade-correlation learning architecture", *Advances in Neural Information Processing Systems*, 2, 1990, pp. 524-532.

[8] S. Feyrer and A. Zell, "Detection, tracking, and pursuit of humans with an autonomous mobile robot", *Proceedings of the International Conference on Intelligent Robots and Systems (IROS'99)*, 1999, pp. 864-869.

[9] J. Fleischmann, K. Buchenrieder, and R. Kress, "Codesign of Embedded Systems Based on Java and Reconfigurable Hardware Components", *Design Automation and Test in Europe*, March 1999.

[10] R. Fong, S. Harper, P. Athanas, "A versatile framework for FPGA field updates: an application of partial self-reconfiguration", *Proceedings of the 14th IEEE International Workshop on Rapid Systems Prototyping (RSP'03)*, 2003.

[11] B. Froba and C. Kublbeck, "Face detection and tracking using edge orientation information", *SPIE Visual Communications and Image Processing*, 2001, pp. 583-594.

[12] V. Gafni. Robots: a real-time systems architectural style. In *Proc 7th European software engineering conference*, Toulouse, 1999, pp. 57-74.

[13] M. Isard and A. Blake, "Condensation - conditional density propagation for visual tracking", *International Journal on Computer Vision*, 29(1), 1998, pp. 5-28.

[14] A. Johnson and K. Nackenzie, "Pattern Matching in Reconfigurable Logic for Packet Classification," *Proceedings of the 2001 international conference on Compilers, architecture, and synthesis for embedded systems*, pp. 126-130.

[15] Y. Li, T. Callahan, E. Darnell, R. Harr, U. Kurkure, and J. Stockwood. Hardware-Software Co-Design of Embedded Reconfigurable Architectures. *Design Automation Conference*, June 2000.

[16] Y. Meng, "An Agent-Based Reconfigurable System-on-Chip Architecture for Real-time Systems". *The 2nd International Conference on Embedded Software and Systems (ICCESS 2005)*, Xian, China, December 16-18, 2005.

[17] D. Murray and A. Basu, "Motion Tracking with an Active Camera", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 16, no.5, pp. 449-459, 1994.

[18] Y. Raja, S. J. McKenna, and S. Gong, "Tracking and Segmenting People in Varying Lighting Conditions," in *Proc. 3rd Int. Conf. on Automatic Face and Gesture Recognition*, pp. 228-233, 1998.

[19] K. Rohr, "Towards Model-based Recognition of Human Movements in Image Sequences", *CVGIP: Image Understanding*, vol. 59, no.1, 1994, pp.94-115.

[20] H. A. Rowley, S. Baluja, and T. Kanade, "Neural Network-based Face Detection", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1), 1998, pp.23-38.

[21] D. Saha, R. S. Mitra, and A. Basu. Hardware/software Partitioning using Genetic Algorithm. In *Proc. of 10th Intern. Conference on VLSI Design*, 1997, pp.155-160.

[22] P. Viola and M. Jones, "Robust real-time object detection", *Proceedings of the Second International Workshop on Statistical and Computational Theories of Vision*, 2001.