NEW TRENDS IN PROTOTYPING FOR ROBOTICS AND AUTOMATION

Kok-Meng Lee* and Tarek M. Sobh**

Guest Co-Editors

* George W. Woodruff School of Mechanical Engineering
Georgia Institute of Technology
Atlanta, GA 30332-0405
Tel: (404)894-7402; Fax: (404)894-9342
email: kokmeng.lee@me.gatech.edu

** School of Engineering and Design
University of Bridgeport
169 University Avenue Bridgeport, CT 06601, U.S.A.
Tel: (203)576-4116; Fax: (203)576-4766
e-mail: sobh@bridgeport.edu

Abstract

Recent developments in several emerging fields enable us to think of the fields of prototype for robotics and automation in new ways and to consider new applications. Today, besides introducing the intelligence directly into equipments/systems through embedded microcomputers and providing virtual prototyping through enhanced CAD/CAE facilities, information flow has been well regarded as an essential part of the integrated design approach whereby all members of the prototype development and manufacturing automation team can work closely together throughout the design and manufacturing cycle. This focused issue contains five papers: this overview paper, the second and third related to automation, and the last two on robotics. We emphasize on the following two subtopics in this paper: the first is an overview for the development of a theory for prototyping discrete event and hybrid systems; and the second highlights elements of prototyping a machine vision for real-time automation applications.

1. INTRODUCTION

During the last two decades, rapid advancements of computer, communication, and control technologies have greatly accelerated the efforts of developing novel prototypes and their cost-effective applications in automation. Today, besides introducing the intelligence directly into equipments/systems through embedded microcomputers and providing virtual prototyping through enhanced CAD/CAE facilities, information flow has been well regarded as an essential part of the integrated design approach whereby all members of the prototype development and manufacturing automation team can work closely together throughout the design and manufacturing cycle.

We decided at the outset of this focus issue on information flow and new trends in prototyping for robotics and automation. There are two broadly defined types of information essential to ensure the optimum performance of automation; namely, off-line knowledge databases and predictive models, and on-line sensing (or real-time feedback). The former provides a baseline of "in-advance" information but if routine deviations are greater than can be tolerated, the latter is needed to augment this baseline information for feedback to controllers. This focused issue contains five papers: this overview paper; the second and third papers are prototyping for automation and the last two are for robotics. We emphasize on two subtopics in this overview paper; namely, the development of prototyping discrete-event and hybrid systems and their applications, and prototyping machine vision for real-time automation applications. Following this overview paper, Riesenfeld *et al.* describe a method for automatically generating complete process plans, including CNC code, from a high-level shape feature part description. Their approach helps designers produce functioning machined parts from their designs, and minimizes the time required to design fixture details. The third paper presented by Khoshnevis *et al.* describes a layered fabrication technique based on a combination of an extrusion process and a filling process to automate contour crafting. Their computer-controlled contour-crafting technique, which has a potential to replace traditional plaster handwork and sculpting, uses computer control of troweling tools to mechanically create various surface shapes. Gosselin *et al.* present a rapid prototyping technology for design and fabrication of robotic mechanisms, a complex process involving geometric, kinematics, dynamics, and tolerance and stress analysis. Using a commercially available CAD package and a Fused Deposition Modeling rapid prototyping machine, several examples are given. In the last paper, Hagras *et al.* discuss the method of prototyping outdoor mobile robots using small lab-based indoor robots (that enable rapid on-line and autonomous learning of controllers) to transfer the learnt controllers from the indoor prototype to the outdoor vehicles.

The remainder of this overview paper focuses on two subtopics. The first is the development of a theory for prototyping discrete event and hybrid systems and its applications. Discrete Event Dynamic Systems (DEDS) are dynamic systems in which state transitions are caused by internal, discrete events in the system. DEDS are attracting considerable interest, and current applications are found in manufacturing systems, communications and air traffic systems, robotics, autonomous systems, and artificial intelligence. We also present an overview for the development of a simple graphical environment for simulating, analyzing, synthesizing, monitoring, and controlling discrete event and hybrid systems. The second is the prototyping machine vision for real-time automation applications. We discuss the problems associated with traditional machine vision systems for cost-effective real-time applications, novel alternative system design to overcome these problems, and the new trends of modern vision sensors.

Modern smart sensors provide the features of a traditional machine vision system at less than half of the usual price by eliminating the signal-conversion electronics, fixed-frame rates and limited gray-scale quantization. Camera, image-acquisition electronics, and computer are integrated into a single unit to allow dynamic access to the CCD without image float or flutter. We also present a physically accurate image synthesis method as a flexible, practical tool for examining a large number of hardware/software configuration combinations for a wide range of parts.

2. DISCRETE EVENT AND HYDRID SYSTEMS

The underlying mathematical representation of complex computer-controlled systems is still insufficient to create a set of models, which accurately capture the dynamics of the systems over the entire range of system operation. We remain in a situation where we must trade off the accuracy of our models with the manageability of the models. Closed-form solutions of mathematical models are almost exclusively limited to linear system models. Computer simulation of nonlinear and discrete-event models provide a means for off-line design of control systems. Guarantees of system performance are limited to those regions where robustness conditions apply. These conditions may not apply during startup and shutdown or during periods of anomalous operation.

Recently, attempts have been made to model low and high-level system changes in automated and semi-automatic systems as discrete event dynamic systems (DEDS). Several attempts to improve the modeling capabilities are focused on mapping the continuous world into a discrete one. However, repeated results are available which indicate that large interactive systems evolve into states where minor events can lead to a catastrophe. Discrete event systems (DES) have been used in many domains to model and control system state changes within a process. Some of the domains include the following: Manufacturing, Robotics, Autonomous Agent Modeling, Control Theory, Assembly and Planning, Concurrency Control, Distributed Systems, Hierarchical Control, Highway Traffic Control, Autonomous Observation Under Uncertainty, Operating Systems, Communication Protocols, Real-Time Systems, Scheduling, and Simulation.

A number of tools and modeling techniques are being used to model and control discrete event systems in the above domains. Some of the modeling strategies include: Timed, untimed and stochastic Petri Nets and State Automata, Markovian, Stochastic, and Perturbation models, State Machines, Hierarchical State Machines, Hybrid Systems Modeling, Probabilistic Modeling (Uncertainty Recovery and Representation), Queuing Theory, and Recursive Functions.

Sections 2 and 3 present a brief review for prototyping discrete event and hybrid systems, discuss some techniques used in the DEDS field, and present a simple software prototyping tool for representing hybrid DES.

2.1 Hybrid and Discrete Event Systems

Discrete event dynamic systems (DEDS) are dynamic systems in which state transitions are triggered by the occurrence of discrete events in the system. DEDS are suitable for representing hybrid systems, which contain one or more of the following characteristics:

- continuous domain,
- discrete domain,
- chaotic behavior, and
- symbolic parameters.

Some examples of DEDS are

Data Network: A ={send, receive, timeout, lost}
Shop with *k* jobs: A={admit_job, job_finished}
Electric Distribution: A= {normal, short_circuit, over_current}

There are several frameworks that can be used to model DEDS such as: finite automata, Petri nets, Markov chains, etc. Choosing one of these frameworks depends on the nature of the problem being modeled and the implementation techniques available to implement this model.

DEDS has been applied to model many real-time problems and has been involved in different types of applications. Some of these applications are:

- networks,
- manufacturing (sensing, inspection, and assembly),
- economy,
- robotics (co-operating agents),
- hghway traffic control, and
- operating systems.

For more details about DEDS applications see [1, 2, 4, 5, 6]. We believe that DEDS will have an important role in the development and improvement of many other applications in different disciplines.

2.2 Discrete Event Models

As mentioned before, there are several representations and frameworks used in DEDS modeling. Some of these frameworks are:

- automata (untimed, timed, temporal, stochastic),
- push-down automata, μ-recursive, and turing machines,
- Petri nets (timed, untimed),
- markov chains,
- queuing theory,
- min-max algebra,
- uncertainty modeling, and
- classical control.

These frameworks can be categorized in three different domains:

Timed vs. untimed models: the untimed models emphasize the "tate-event sequence" of a DEDS and ignore the holding time of each state, while in the timed models, "time" s an essential part of the model.

Deterministic vs. probabilistic models: deterministic models assume pre-knowledge of the sequence of events that will occur at any time, while probabilistic (stochastic) models associate probabilities with each event.

Computational model: which can be *logical models* in which the primary questions are of qualitative or logical nature, while *algebraic models* can capture the description of the trajectories in terms of a finite set of algebraic operations. Finally, *performance models* are formed in terms of continuous variables such as average throughput, waiting time, etc.

Figure 1 shows the different models of representing DEDS and their characteristics. More about DEDS models can be found in [3].

	Timed	Untimed
Logical	Temporal Logic Timed Petri Nets	Finite State Machines Petri Nets
Algebraic	Min-Max Algebra	Finitely recursive proc. Comm. sequential proc.
Performance	Markov Chains Queuing Networks GSMP/Simulation Stochastic Petri Nets	

Stochastic→ ←Nonstochastic

Figure 1 Different Models for DEDS

2.3 Evaluation of DEDS

The evaluation of each framework can be done in four dimensions:

- Descriptive power
  - Language complexity
  - Algebraic complexity,
- Implementation
- Performance evaluation
  - Logical correctness
  - Real-time requirements, and
- Applications

Language complexity is based on the formal theory of languages. Each FSM generates a language *L* which represents all possible traces of this FSM.

$$L(FSM) \subset L(Petrinets)$$

So, Petri nets is more *language complex* than FSM.

Algebraic complexity is based on the systems theory. We can consider any algebraic system as a set of models and a set of operators that map one or more model to another. For example, in transfer functions, addition and multiplication reflect serial and parallel systems.

Logical correctness is a desirable property of the traces generated by any DEDS model. For example, in the data network example, we must guarantee that each transmitted packet has been received correctly by the receiver.

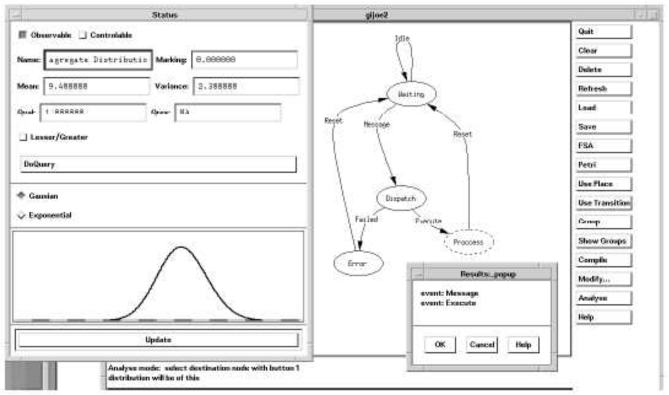
Real-time requirement is a desirable property of the real-time response of the actual system. It is necessary to embed the DEDS model in a real-time environment.

3. A SIMPLE PROTOTYPING TOOL

We have built a software environment to aid in the design, analysis and simulation of Discrete Event and Hybrid Systems. The environment allows the user to build a system using either Finite State Machines or Petri-Nets. The environment runs under X/Motif and supports a graphical DES (Discrete Event System) hybrid controller, simulator, and analysis framework. The framework allows for the control, simulation and monitoring of dynamic systems that exhibit a combination of symbolic, continuous, discrete, and chaotic behaviors, and include stochastic timing descriptions (for events, states, and computation time), probabilistic transitions, controllability and observability definitions, temporal, timed, state space, Petri-nets, and recursive representations, analysis, and synthesis algorithms.

The environment allows not only the graphical construction and mathematical analysis of various timing paths and control structures, but also produces C code to be used as a controller for the system under consideration.

Using the environment is fairly simple. For finite state machines the designer uses the mouse to place states (represented by ovals) and connect them with events (represented by arrows). Transitions and states can be added, moved and deleted easily. Figure 2 is an example of a simple stochastically timed FSM, containing 4 states and 5 events.



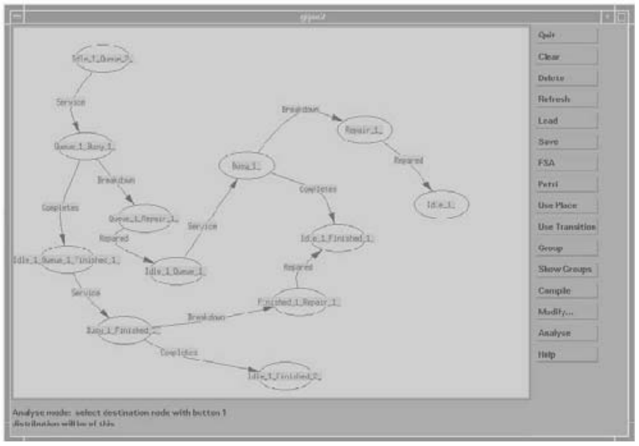
Name: agregate Distributio
Marking: 0.000000
Mean: 9.400000
Variance: 2.300000
Qual: 1.000000
Qpar: NA

event: Message
event: Execute

Analyse mode: select destination node with button 1
distribution will be of this

Figure 2 A Stochastically timed FSM window during analysis

The probabilities on the events (that is, which path to navigate in the automaton) are designated using the marked field in the status dialog box. The different timings (on event and state times) and distribution function type, mean and variance can be assigned through the status dialog box too. The allowable distributions are currently restricted to Gaussian and exponential functions, but can be easily extended to arbitrary discrete or continuous distributions. A window shows the distribution function at a state or event, and also allows the user to do queries: for example, queries on whether a path time probability is greater or less than a given time, or combined timing distributions to reach a goal state through various paths, etc. The dialog box allows the user to perform queries of various kinds. The currently selected state/event is drawn with a dashed line, and the information in the status window pertains to it. Optimizing paths based on stochastic timing can also be performed; in that case, windows will pop out with the event path, and the status window will have the combined distribution function. Figure 3 presents an automaton model in the environment. The environment also produces C code for controlling the system under consideration. In our PN model we have extended the definition of stochastic timed Petri Nets, to have additional timings. Our model has three times associated with it, a place time, a delay time, and an event time (see Figure 4). The place time is a time where the token is held back and delays the enabling of the transition. This represents the computation time of that place. The delay time is a time associated with the input arcs to a transition. It represents the time to leave the corresponding place. The event time is analogous to the single time in stochastic timed Petri Nets which is called *firing time*. We believe that this lends to a more intuitive representation of the times and simplifies the modeling task since it captures more details than the original timed Petri net model.



Analyse mode: select destination node with button 1
distribution will be of this

Figure 3 A snap shot of the FSM environment

We can define the new model as:

$$PN = (P,T,A,W,x_0)$$

where

P = set of places with associated random variables;

T = set of transitions;

A = $A_{in} \cup A_{out}$ with

- A_{in} set of elements from {P x T} with associated random variables;
- A_{out} set of elements from {T x P};

W = a weight function, $w{:}A \rightarrow \{1,2,3,\ldots\}$; and

x_0 is an initial marking.

The environment for Petri Nets is similar. Circles, transitions by ellipses, and arcs by arrows represent places graphically. As mentioned above, there are three locations where one can place timing information: on the events- the event time, which is the time the actual event takes; place time- when a token is moved through a transition firing there is a place time which hides the token until it has expired; and the final time is a delay time which comes into effect when a transition fires- it is the time for the event to reach the transition. The event time will not start until all of its input tokens delay time has expired. Figure 5 depicts a snap shot of the Petri Net environment in action.

The system generates C code for the user hybrid system, so one can simulate and control an actual system using the code. The C code is currently generated for FSMs (soon code will be generated for PN's too). A Petri Net will be converted to a FSM before code is generated; all of the timing is then placed on the events. The user has to select the initial state, and provide the function for simulating/generating the events; the code will keep track of the elapsed simulated time, and will return when it reaches a state with no transitions.

The environment allows conversion back and forth between the FSM and PN models. Conversion to a Petri Net is straight forward, but one loses the event probabilities. The only thing that's needed is to create a transition for every event. Conversion from a Petri Net to a FSM is only possible if the PN is *k*-bounded, which means no place can ever have more than *k* tokens. The system generates a state for every possible marking of that net. The states are represented as the marking; the events are just the transitions. The three "times" are pushed into the events. The system convolves the maximum of the input delays with the event, and the maximum of the place times. The maximum function is a standard convolution, except that the maximum is used instead of multiplication.

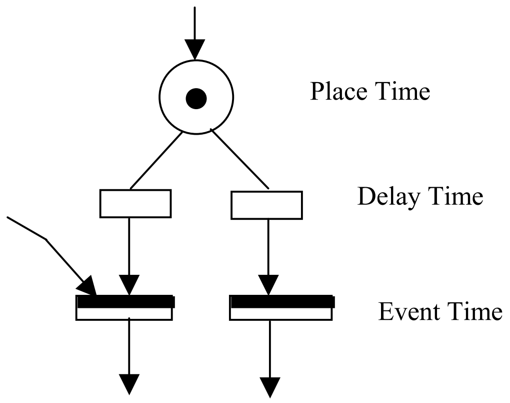
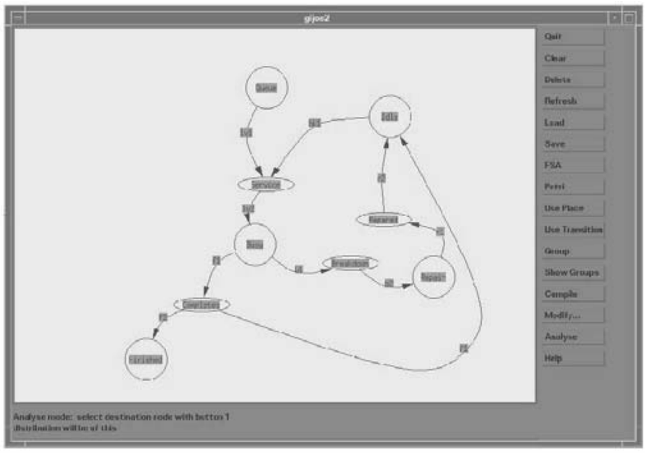


Figure 4 The proposed three time zones for a timed Petri Net



Analyse mode: select destination node with button 1
distribution will be of this

Figure 5 A snap shot of the Petri Net environment

The algorithm for generating all of the markings starts with some initial marking, and then goes through all of the possible transitions. If it can fire, the firing is simulated, and the new marking is inserted into the set of states. If it is already represented, the transition is kept; otherwise the transition is kept and recursion is done with the new marking. This process is repeated until no transitions can be fired.

Our system can serve as a simple graphical simulator, analyzer, synthesizer, monitor, and controller for hybrid systems models using either Petri Nets or FSMs high-level frameworks.

4. TRENDS IN MACHINE VISION SYSTEM DESIGN

The predictive model provides a baseline of "in-advance" information, but if routine deviations are greater than can be tolerated, sensors are needed to augment this baseline information for feedback to controllers. In existing systems, estimates of the impact of sensing systems on process performance indicate as much as a six fold increase in effective operation speed [7]. A general review of different sensors for robotics and automation can be found in [8]. One of the major contributions of information technologies to sensors was the idea of digitized output, which removed analog variation from the outputs. A good illustrative example is machine vision, which grows from a standard composite video signal that the television industry uses, to a more general-purpose sensor with on-board intelligence.

However, although it has been well recognized in the past three decades that vision can add considerably to flexibility by simplifying grippers, component feeders, and location tooling, and can reduce the engineering time required to implement it, the capabilities of commercial vision systems for use in part verification, kitting, and presentation for robotic assembly are still very limited. Until the late 1980's, most of the vision systems employed a camera that outputs a video signal limited by the traditional TV standard (typically 30 frames per second specified by the RS170 established in the 1950's) and an object-dependent structured illumination. For use as a robot vision system, a frame grabber board and a high performance host computer must accompany the video camera. The conventional vision approach generally discards color information and requires a substantial amount of memory and data communication time, and sophisticated vision interpretation. Variations in surface reflectance, coupled with algorithm computational demands, often make the conventional approach too expensive, unreliable, and slow. In addition, the conventional vision approach, which attempts to emulate human eyes and brain, does not necessarily yield the accurate data required by the robots.

4.1 Alternative Vision System Architecture

To overcome the problems associated with the traditional video-based vision system, several vision systems were designed for robotic applications. Among these is a Flexible Integrated Vision System (FIVS) developed at Georgia Tech in the late 1980's [9], which offers performance and cost advantages by integrating the imaging sensor, control, illumination, direct digitization, computation, and data communication in a single unit. By eliminating the host computer and frame grabber, the camera is no longer restricted by the RS-170 standard and thus frame rates higher than 30 fps can be achieved.

Architecture

As shown in Figure 6, the central control unit of the flexible integrated vision system is a microprocessor-based control board. The design is to have all of the real-time processing performed using the microprocessor control board without relying on any other system or computer. The prototype of FIVS is shown in Figure 7.

On-board processor

The DSP-based control board is designed to communicate with several option boards in parallel to tailor the system for a number of applications. Each of these option boards is controlled independently by a programmable logic device (PLD), which receives a peripheral select signal, a read/write signal, and an address signal from the microprocessor control board.

Typical examples of the option boards for the FIVS are the digital video head, a real-time video record/display/playback board, and an expandable memory board.

Camera

The video head consists of an *m x n* CCD array, the output of which is conditioned by a high bandwidth amplification circuitry. The output is then sampled by a "flash" analog-to-digital converter (ADC). The DSP-based control board provides a direct software control of the CCD array scanning and integration time, the intensity of the collocated illumination, and the real-time execution of a user-selectable vision algorithm imbedded in the EEPROM. In operation, the PLD decodes the control signals to initiate row shifts and column shifts in response to commands from the DSP-based control board. Particular row shifts and column shifts enables retrieving only a specific relevant area from an image. The PLD also provides control signals to ADC for performing the analog-to-digital conversion synchronized with row shifts, and enables the video buffer when the DSP reads or writes data to the VRAM.

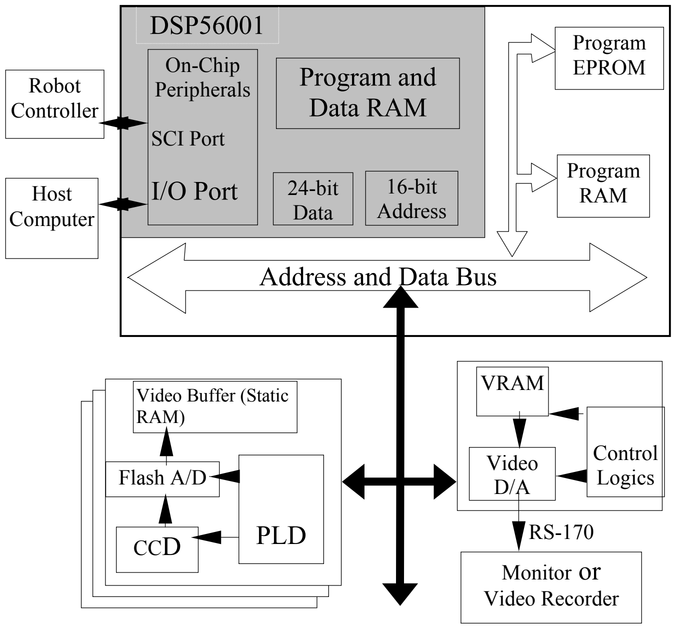


Figure 6 Schematic of a flexible integrated vision system

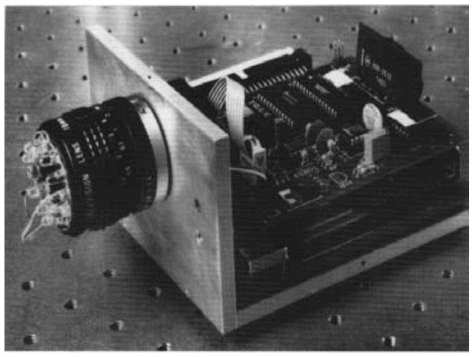


Figure 7 FIVS and its collocated illumination system

Imbedded software

The vision system imbedded software gives users the flexibility to control the CCD array scanning and integration time and the intensity of the illumination. With the CCD under software control, partial frames can be "captured" instead of the customary full frame, reducing the cycle time required to capture and process an image. The ability to shift out partial frames is ideal for high speed tracking applications where the approximate location is known from a prior image. By reducing the time to capture an image, the effective frame rate is increased. For example, shifting out 1/4 of an image can increase the frame rate up to 480 fps, not including the time required for illumination and image processing. This frame rate is 16 times the rate achievable from the RS-170 standard.

4.2 The New Trends

Unlike conventional RS170-based systems which require pixel data to be stored in a video buffer before processing of pixel data can commence, the FIVS design provides an option to completely bypass the video buffer and thus offers a means to process and/or to store the digitized pixel data by directly transferring the ADC output to the DSP. For real-time vision-based object tracking and motion control system applications, the scheme represents a significant saving in time and video buffer size required for processing an image. As an illustration, consider an image array of *m x n* pixels. The time needed to store the entire image (with no computation) in a memory at K MHz is (*m* x *n*)/K μs and requires (*man*) bytes of memory. Typical array size of a CCD ranges from 200x160 to 4096x4096 of pixels. The corresponding video buffer and time required simply to store the entire image at a clock rate of 10 MHz would range from 32K bytes to 16M bytes and 3.2 ms to 1600 ms respectively! Clearly, the option to completely bypass the video buffer offers a potentially useful solution to eliminate the frame storage prerequisite which is often required in conventional vision systems. Furthermore, this scheme completely eliminates the special hardware needed in acquiring the digitized pixel data for storage.

Applications

With the on-board intelligence, computer controlled machine vision systems have found a number of real-time applications, where the accuracy of image gray-scale pixel values far outweighs image. Some of these examples are robotic part pickup [10], motion tracking [11], three degrees-of-freedom orientation sensing [12], servo-track-writing in hard disk drive manufacturing [13], disassembly automation [14], and haptic sensor [15].

The advance in direct-digital machine vision will continue to lead to new ways of addressing industrial automation problems that were difficult (if not impossible to solve), particularly for the traditional industries. One such example is an on-going development of a high-speed live-bird handling system for poultry processing applications, where machine vision has played a significant role in automating the process of transferring live birds from a moving conveyor to shackles, typically at a line speed of 3 birds/second. Live-bird handling problems have been found difficult because the birds tend to flail about when they are caught. Non-evasive techniques must be developed along with the study of stimulus environments to promote behavior compliance, the study of the role of visual responsiveness, and the evaluation of vision acuity in different spectral environments. Often such real-time control application requires a stringent combination of structured illumination, reflectance, and imaging sensor.

Structured illumination and reflectance of a machine vision system could play a significant role in the live-bird handling application [16], where the bird's posture is determined for real-time manipulation of the bird's legs. In order to keep the bird from flailing and its presentation uniform as the bird enters the grasper, retro-reflective sensing technique [10] has been used to obtain a snap shot of the moving bird. The structured illumination system consists of a low-intensity spectrally filtered illumination that insensitive to the bird, and a vision system that captures a snap shot of the bird against a retro-reflective background. Figure 8 shows an image of a bird on a conveyor moving at 0.5m/s toward the grasper. The image of he bird was captured against a 580-85 Black Scotchlite retro-reflective background with a low-intensity illumination filtered with a Roscolux full-blue filter since birds' are insensitive to blue light (or low 400nm wavelengths).

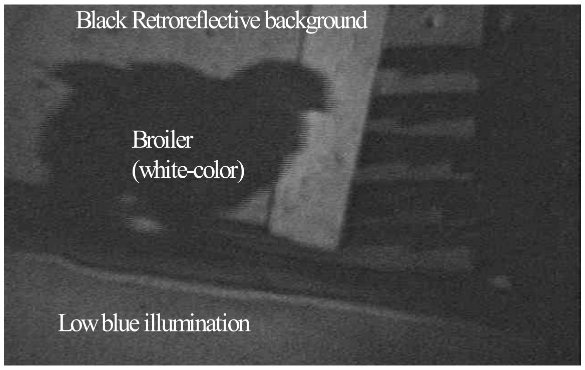


Figure 8: An example with structured illumination/reflectance (snap shot of a live bird)

Advances in New Imaging Sensors

In the early 1990's, CMOS sensors emerged as low-cost, low-power alternatives to CCD. The principle architectural solutions, which enabled the high data throughput, are the effective integration of pixel readout processing, ADC conversion, and the high-speed dual-port RAM in one single chip. The core of the CMOS sensor designs is an *m* x *n* photodiode active pixel array, which is accessed in row-wise fashion and readout into a column ADC's in parallel. With the addition of a column of dual-port SRAM's, the readout of the digital data can be done during the A/D conversion of the next row. The sensor has an on-chip digital block, which runs the row processing, ADC conversion, and readout and allows flexibility in selecting rows and columns as well as defining the start time for row processing or read. The use of parallel pixel readout and digitizing, as well as easy ways of multiplexing/de-multiplexing data- techniques required for high-speed large format sensors- is challenging CCD technology in mainstream applications. Today, CMOS sensors, for example the 1024x1024 CMOS Active Pixel Sensor [17], has the potential to achieve very high output data rate over 500MB/second and a low power dissipation of 350mW at a clock rate of 66MHz.

Attempts to emulate human visual perception have led to the development of high dynamic-range color (HDRC) imaging systems [18]. The power of human visual perception lies in its very high dynamic range, its robust object detection due to high and constant contrast resolution in both bright and dark regions of a scene. Natural photoreceptors like those in our eye have a logarithmic response that detects very fine absolute steps in the dark or shade while they limit their response to larger absolute steps at high intensities. HDRC CMOS pixel generates an output voltage equivalent to the log of the local optical intensity. As a result, the high dynamic-range of the HDRC camera outperforms the digital CCD camera that has reached its limits in spite of its advantage in resolution and all its post-processing power.

Computing and Integration

The recent introduction of microprocessors with large internal caches and high-performance external memory interfaces make it practical to design high-performance imaging systems with balanced computational and memory bandwidths. Sgro *et al* [19] presents a framework that allows a developer to choose a microprocessor system that offers the performance and scalability often required by a real-time vision application. Using the component inspection application as an example, they demonstrate that coprocessor-based solutions with local memory architects allow throughput to scale linearly as the number of processors increase. For demanding vision applications, especially those that require future expansion, the most practical solution remains a co-processor board that is more scalable, has higher throughput, and ultimately is cheaper than the native solution.

Finally, one other potential impact is the influx of low-cost USB and Firewire cameras into the lucrative consumer market that drives the development of the USB [20] and IEEE 1394 [21] (commonly known as Firewire) communication standards. The USB standard was designed to replace typical parallel and serial I/O ports (such as RS232) and has been widely accepted by the PC industry. Future USB 2.0 is expected to have a speed over 120-240 Mbps. The IEEE 1394 standard was designed as a high-speed bus with digital video as its target application. The bus currently runs at speeds up to 400 Mbps and expects to exceed 1600 Mbps in the near future.

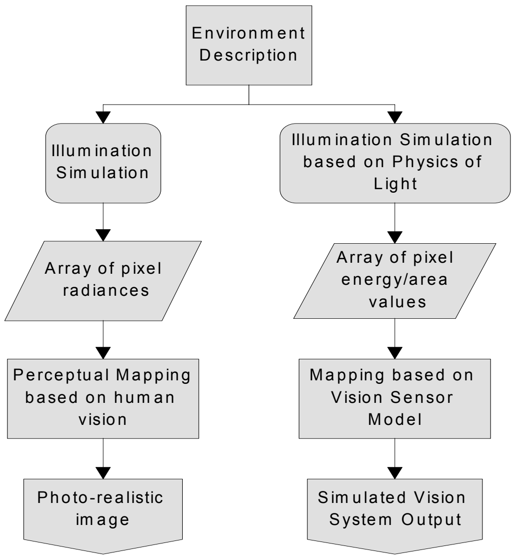
5. PROTOTYPING MACHINE VISION DESIGN

Imaging sensors are characterized by their specific bandwidths or wavelengths of light, which maximize the sensor response and will provide it an optimum operating environment. It is desired that the photo-detector respond only to the light from the illumination source structured for the object but not that of ambient lighting. Synthetic images [22] can efficiently be used to study the effects of illumination and vision system design parameters on image accuracy, providing insight into the accuracy and efficiency of image-processing algorithms in determining part location and orientation for specific applications, as well as reducing the number of hardware prototype configurations to be built and evaluated. Figure 9 compares the processes used to generate synthetic images for (a) photo-realistic and (b) physically-accurate synthetic images for vision system applications.

As shown in Figure 9, an accurate mathematical model is needed to describe the physical scene and the vision system used to capture that scene. This model is used to simulate scene illumination, which is represented as an array of [pixel] radiances. This array of radiances is then converted to energy/area values, which are transformed by a mapping based on a model of the system sensor and how it converts incident light energy into gray-scale values.

The physically accurate synthetic image is simulated in a two-step process. In the first step, *RADIANCE*, a freely-distributed software package from the Lighting Systems Research Group of the Lawrence Berkeley Laboratory, is used to solve the radiative heat transfer equation. In the second step, the sensor model for the computer vision system is modeled using a power law [10].

Figure 10 quantitatively compares various methods of generating synthetic images, where synthetic images of the retroreflective background were generated and compared to a captured image of the retroreflective field (Figure 10). As seen in Figure 10(a), a CAD-generated image assuming an ideal diffuse surface results in an image that is nearly black. Figure 10(b) illustrates *RADIANCE's* ability to model the retroreflective background; however, the illuminated area is too small and too sharply defined. Incorporation of the finite aperture (Figure 10(d)) results in an image with a more acceptable transition between the illuminated and non-illuminated areas, but the illuminated area is still too small. The importance of accurate source emission distribution modeling is shown in Figure 10(c). Other detailed illustrative examples of using physically-accurate synthetic images can be found in [23].


Environment Description
Illumination Simulation
Illumination Simulation based on Physics of Light
Array of pixel radiances
Array of pixel energy/area values
Perceptual Mapping based on human vision
Mapping based on Vision Sensor Model
Photo-realistic image
Simulated Vision System Output

(a) Photo-realistic (b) Physically-accurate

Figure 9 Model of the synthetic imaging process

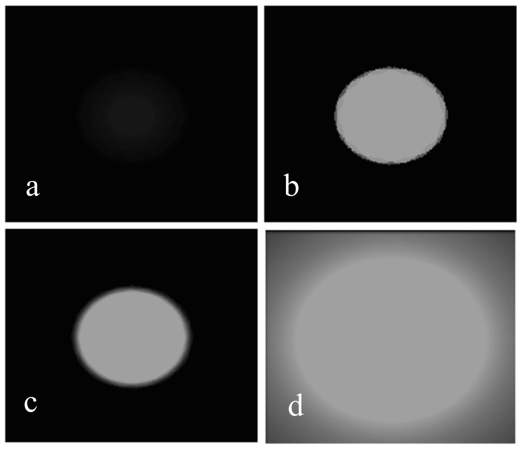


Figure 10. Synthetic images of retroreflective field

The benefits of this realistic image synthesis are three-fold. First, it provides a rational basis for designing the hardware and software components of a machine vision system. Secondly, it provides a standard platform for comparing algorithms and predicting the optimal algorithm (and optimal performance) for a specific application. Additionally, it provides an opportunity to perform an in-depth study of the factors that can significantly degrade the performance of image-processing algorithms and aid in the determination of critical design parameters. A third benefit is the ultimate development of a well-designed CAD-tool which utilizes physically-accurate synthetic images to accurately and inexpensively predict the performance of a proposed vision system design prior to implementation or the construction of a prototype, minimizing the need to build and test a large number of hardware configurations. Such a tool also allows necessary changes in part design to be made earlier in the design phase, significantly reducing implementation time and improving industrial reliability.

5. CONCLUSIONS

We have presented trends of prototyping design and automation with an emphasis on the following two subtopics. The first is a brief review of Discrete Event and Hybrid Systems prototyping. A simple software environment system was developed for simulating, analyzing, synthesizing, monitoring, and controlling discrete event and hybrid systems.

The second is a review and the trends on prototyping real-time machine vision system design. Specifically, we present an alternative system design to overcome those problems associated with a traditional video-based vision system, which is a physically-accurate image synthesis method as a flexible, practical tool for examining a large number of hardware/software configuration combinations for a wide range of parts.

REFERENCES

[1] Balemi, S. Control of Discrete Event Systems: Theory and Application. Ph.D. thesis, Automatic Control Laboratory, Swiss Federal Inst. of Tech. (ETH), Zurich, Switzerland, May 1992.

[2] Ho, Y. Discrete Event Dynamical System and its Application to Manufacturing. In IFAC Cong. Proc. (1981).

[3] Y. C. Ho and X.-R. Cao, Perturbation Analysis of Discrete Event Dynamic Systems. Boston: Kluwer Academic Publishers, 1991.

[4] G. J. Olsder, “Applications of the theory of discrete event systems to array processors and scheduling in public transportation," in Proc. of 28th IEEE Conf. on Decision and Control, Dec. 1989.

[5] Sobh, T., Bajcsy, R., and James, J. Visual observation for hybrid intelligent control implementation. In 31st IEEE CDC (Tucson, AZ, USA, December 1992), IEEE.

[6] Sobh, T., Jaynes, C., and Henderson, T. A discrete event framework for intelligent inspection. In Proc. 1993 IEEE Int. Conf. Robotics and Automation (May 1993). Atlanta, GA.

[7] Eversheim, W., 1991. "Strategies and Tools meeting Future Challenges in Manufacturing," Int. Conf. on Appl. of Manufacturiing Technology, SME, Apr. 17-19, Washington, D. C.

[8] Lee, K.-M., “Sensors And Actuators,” Section in Chapter on Robotics, CRC Handbook of Mechanical Engineering, CRC Press, 1998.

[9] Lee, K.-M. and Blenis, R., "Design Concept And Prototype Development Of A Flexible Integrated Vision System," Journal Of Robotic Systems, 11(5), 387-398, (1994).

[10] Lee, K.-M., "Design Concept Of An Integrated Vision System For Cost-Effective Flexible Part-Feeding Applications," ASME Trans. Journal Of Engineering For Industry, Vol. 116, pp. 421-428, November 1994.

[11] Lee, K.-M. and Qian, Y.-F., "Development Of A Flexible Intelligent Control For Target Pursuit," ASME Trans. Journal of Manufacturing Science and Engineering, Vol. 120, August 1998, pp. 640-647.

[12] Lee, K.-M., Meyouhas, G., and Blenis, R., "A Machine-Vision-Based Wrist Sensor For Direct Measurement Of Three Degrees-Of-Freedom Orientation," Mechatronics, Vol. 3, No. 5, pp. 571-587, 1993.

[13] Garner,H. K-M. Lee, L. Guo, "Design Analysis of a Grating Interferometer Sensor for HDD Servo-Track Writing," IEEE/ASME AIM'99 Proceedings, Atlanta, GA Sept. 18-23, 1999.

[14] Presti, Peter, "The Haptic Lens – a Tactile Sensor," IEEE/ASME AIM'99 Proceedings, Atlanta, GA September 18-23, 1999.

[15] Lee, K.-M. and Bailey-Vankuren, M., “Supervisory Control of an Automated Disassembly Workcell Based Blocking Topology,” Proceedings of the 1997 IEEE International Conference on Robotics and Automation, Albuquerque, New Mexico, April 20-25, 1997.

[16] K.-M. Lee, “Design Criteria for Developing an Automated Live-Bird Transfer System,” IEEE Int. Conf. on Robotics and Automation, San Francisco, April 22-28, 2000.

[17] Zhou, Z., B. Pain, E. R. Fossum, "CMOS Active Pixel Sensor with On-Chip Successive Approximation Analog-to-Digital Converter," IEEE Trans. on Electron Devices, pp. 1759-1763, Oct. 1997.

[18] Seger, U., H.-G. Graf, Landgraf, “Vision Assistance in Scenes with Extreme Contrast,” IEEE Micro 13(1), February 1993, S. 50-56.

[19] Sgro, J., P. Stanton, and J. Delfino, “Evaluation Framework for Scalable Microprocessor Vision Systems in an industrial Inspection Application,” Alrcon, Inc.

[20] “USB Press,” http://www.usb.org/press, June 1999.

[21] Intel, “1394 Technology,” http://developer.intel.com/technology/1394, June 1999.

[22] Parker, J. and K.-M. Lee, “Physically-Accurate Synthetic Images for Machine Vision Design,” ASME Trans. Journal of Manufacturing Science and Engineering, November, 1999.