Towards intelligent autonomous Vision Systems

Smart Image Processing for Robotic Applications

Andreas Maeder – Hannes Bistry – Jianwei Zhang

TAMS

University of Hamburg
Faculty of Mathematics, Informatics and Natural Sciences
Department of Informatics
Technical Aspects of Multimodal Systems

IEEE International Conference on Robotics and Biomimetics
December 15. - 18. 2007 – Sanya, China

Outline

- Motivation
  - Introduction
  - Problem
  - Solution
- System design
- FPGA-based system
  - Components
  - Design flow
  - Results
- Smart-camera system
  - Components
  - Results
- Further development

Outline

Motivation
- Introduction
- Problem
- Solution

System design

FPGA-based system
- Components
- Design flow
- Results

Smart-camera system
- Components
- Results

Further development

Outline

- Motivation
  - Introduction
  - Problem
  - Solution
- System design
- FPGA-based system
  - Components
  - Design flow
  - Results
- Smart-camera system
  - Components
  - Results
- Further development

Outline

- Motivation
  - Introduction
  - Problem
  - Solution
- System design
- FPGA-based system
  - Components
  - Design flow
  - Results
- Smart-camera system
  - Components
  - Results
- Further development

Outline

- Motivation
  - Introduction
  - Problem
  - Solution
- System design
- FPGA-based system
  - Components
  - Design flow
  - Results
- Smart-camera system
  - Components
  - Results
- Further development

Introduction

New generation of service robots relies on environmental information for

- localization and navigation within the environment
- recognition and manipulation of objects
- interaction with users / other robots

Different sensors will be used – keywords:

- highly dynamic sensor information
- multiple modalities
- sensor fusion

⇒ Vision systems are a key technology

Introduction

New generation of service robots relies on environmental information for

- localization and navigation within the environment
- recognition and manipulation of objects
- interaction with users / other robots

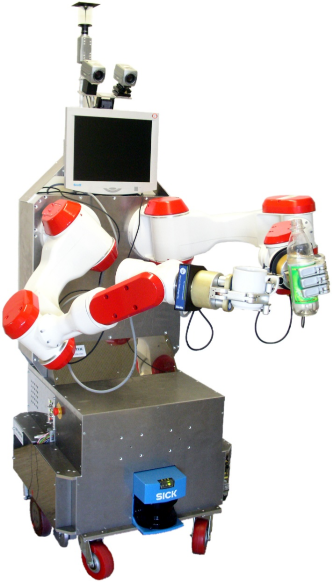
Different sensors will be used – keywords:

- highly dynamic sensor information
- multiple modalities
- sensor fusion

⇒ Vision systems are a key technology

Service Robot

- Cameras
  - omnidirectional
  - stereo-vision
  - hand
- Laser range finders
- Microphone(s)
- Force / Torque
  - arm
  - hand
- Gyro
- Position
  - platform
  - arm
  - hand
  - pan tilt unit



- Platform
  Neobotix MP-L655
- Arm
  Mitsubishi PA10-6C
- Hand
  Barrett BH8-262
- Pan tilt unit
- Speakers
- Face

...

Problem

Standard PC as control hardware

- limited processing resources
- real-time constraints could not be fulfilled

What makes the situation worse

- multiple sensor data streams
- several tasks to be done
  - input data processing
  - low-level control of the platform
  - high-level robotics: learning, planning, etc.

Problem

Standard PC as control hardware

- limited processing resources
- real-time constraints could not be fulfilled

What makes the situation worse

- multiple sensor data streams
- several tasks to be done
  - input data processing
  - low-level control of the platform
  - high-level robotics: learning, planning, etc.

Workaround

Sequentialization of tasks / use of subsystems

Example: tasks and their use of sensors

- Exploration, localization and movement of the robot
  - Laser range and odometric data
  - Omnidirectional vision system (localisation, not realtime)
- Arm operation and user interaction — immobile robot
  - Active stereo-vision system
- Grasp and arm approach — immobile robot
  - Hand camera

Solution

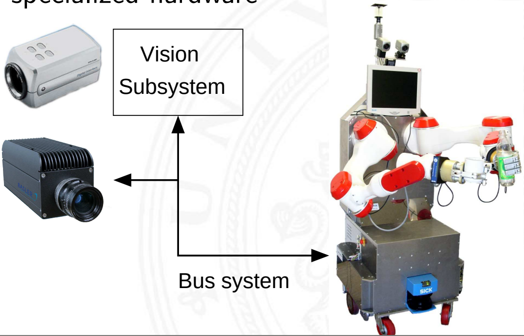
Distributed processing approach: Smart-Sensors / -Subsystems

- Vision
  - high data rates
  - computational complexity
- Vision processing hierarchy
  1. Image enhancement: normalization, calibration
  2. Preprocessing: filter and morphological operators
  3. Sensor data fusion: e.g. 3-D processing
  4. Feature extraction: segmentation, higher-level operators
  5. Classification and image analysis

$\Rightarrow$ Existing Smart-Cameras may handle 1. and 2.

Requirements

- Abstract from images, generate (less) feature data
- Implement direct control loops
- Flexibility $\Rightarrow$ embedded (software) system
- Performance $\Rightarrow$ specialized hardware



Workflow

Idea: "Smooth"-Transfer

1. Algorithm on Control Computer
2. Transferred to embedded processor or DSP
3. Hardware accelerated, when needed
   FPGA-based, for dynamic reconfiguration

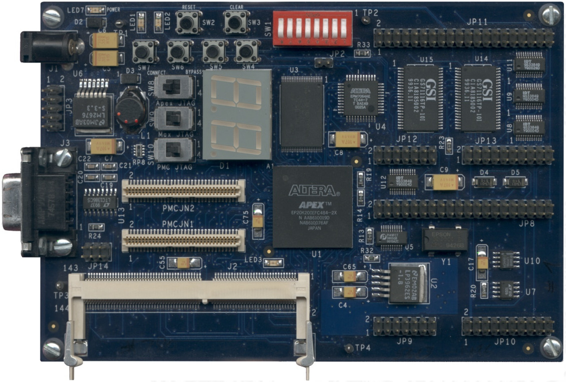
Two prototyping systems

- Evaluate specifics of architectural alternatives
- Define interfaces to software components of service robot
- Derive system requirements

Components

FPGA prototyping board

- Altera® NIOS
- realizes
  - hardware
  - CPU-core
    + software
- on board
  - interfaces
  - memory



Components (cont.)

FPGA prototyping board

- Programmable logic $\approx$ 500.000 gates
  1. user-defined hardware
  2. processor: 16/32-bit, widely configurable
  3. IP-components: RAM, FIFO...
- Toolchain
  1. VHDL code + Synthesis
  2. C code + Compiler (programming)
     Altera SOPC Builder (generation)
  3. IP generation/configuration program

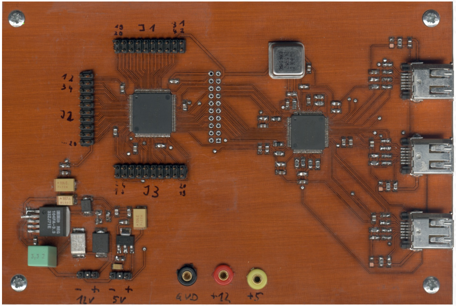
Components

FireWire board

- Implementation of the physical- and link-layer of IEEE 1394
- Standard components
- Memory-mapped interface
  32-bit data
  8-bit address

link layer controller TSB12LV01B

physical layer chip TSB41AB3



Design flow

Stepwise transition for existing / new algorithms

1. on PC on robot platform
2. software on embedded processor
3. –"– and special hardware for time-critical computing
4. dedicated hardware solution
   program control flow realized as state machines

Data-driven, hardware-implemented parts of the system

- link layer interface: response within one clock cycle (30 MHz)
- image processing pipelines running at full speed
- internal FIFOs, buffer data in between units
- SDRAM controller: additional image memory

Design flow

Stepwise transition for existing / new algorithms

1. on PC on robot platform
2. software on embedded processor
3. –"– and special hardware for time-critical computing
4. dedicated hardware solution
   program control flow realized as state machines

Data-driven, hardware-implemented parts of the system

- link layer interface: response within one clock cycle (30 MHz)
- image processing pipelines running at full speed
- internal FIFOs, buffer data in between units
- SDRAM controller: additional image memory

Design flow (cont.)

Bottom-up system development, increasing the data rate following the abstraction levels in image processing

1. Synchronization with the isochronous FireWire data
2. $+$ Asynchronous communication for control tasks
   $\Rightarrow$ Automatic focus
3. $+$ Generation of isochronous data streams
   $\Rightarrow$ Digital filter
4. $+$ Addition of image memory for extended capabilities
   $\Rightarrow$ Omnidirectional to panoramic image conversion

The FPGA prototype system is an excellent architectural workbench to evaluate design alternatives and to specify requirements of the overall system.

Design flow (cont.)

Bottom-up system development, increasing the data rate following the abstraction levels in image processing

1. Synchronization with the isochronous FireWire data
2. \+ Asynchronous communication for control tasks
   ⇒ Automatic focus
3. \+ Generation of isochronous data streams
   ⇒ Digital filter
4. \+ Addition of image memory for extended capabilities
   ⇒ Omnidirectional to panoramic image conversion

The FPGA prototype system is an excellent architectural workbench to evaluate design alternatives and to specify requirements of the overall system.

Statistics for several designs

	cells		[bit]		[MHz]
	logic	register	memory	pins	F_{max}
FireWire Control	392	208	2048		
Fifo 64×32	29	20	2048		
CPU nios32	3167	1359	38912		
AF-Pipeline	324	145	–		
ShiftReg 2×128×16	18	14	4096		
Automatic Focus	3930	1746	47104	136	49.39
FireWire Control IO	681	299	2048		
Fifo 256×32	33	23	8448		
Fifo 512×32	39	29	16384		
Processing omni	1521	344	–		
Sine ROM (2×)	–	–	6656		
SDRAM Control	2287	920	2048		
Image Conversion	7798	3019	75008	231	47.63

Automatic focus for active stereo-vision system

Problem

- Algorithm on service robot causes 30 % cpu load

⇒ focus fixed to arm's length, no adjustment!

⇒ working with defocused camera data voids calibration efforts!

Solution: Simple control application

1. Subsystem analyzes isochronous image data
2. Computes measurement for focus quality
3. Implements algorithm optimizing this focus value
4. Sends asynchronous commands controlling the cameras

Automatic focus for active stereo-vision system

Problem

- Algorithm on service robot causes 30 % cpu load

⇒ focus fixed to arm's length, no adjustment!

⇒ working with defocused camera data voids calibration efforts!

Solution: Simple control application

1. Subsystem analyzes isochronous image data
2. Computes measurement for focus quality
3. Implements algorithm optimizing this focus value
4. Sends asynchronous commands controlling the cameras

Automatic focus (cont.)

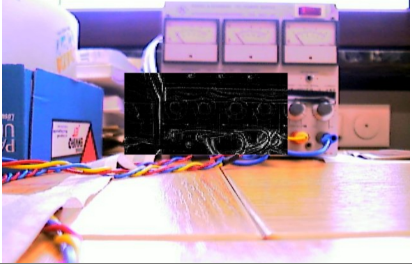
Focus Algorithm

- Measure: Sum up Laplace filtered image of central region
- *Global* interval search for maximum
- *Local* optimization adapts focus to changing scenes
- Dynamic threshold for transition between operation modes

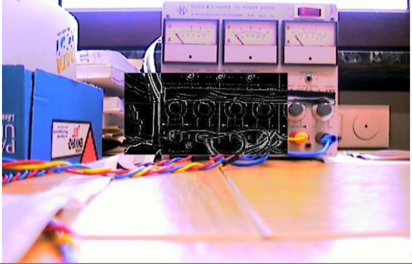
Automatic focus (cont.)

Focus Algorithm

- Measure: Sum up Laplace filtered image of central region
- *Global* interval search for maximum
- *Local* optimization adapts focus to changing scenes
- Dynamic threshold for transition between operation modes



Sum: 255.960 — Defocused: -10 cm



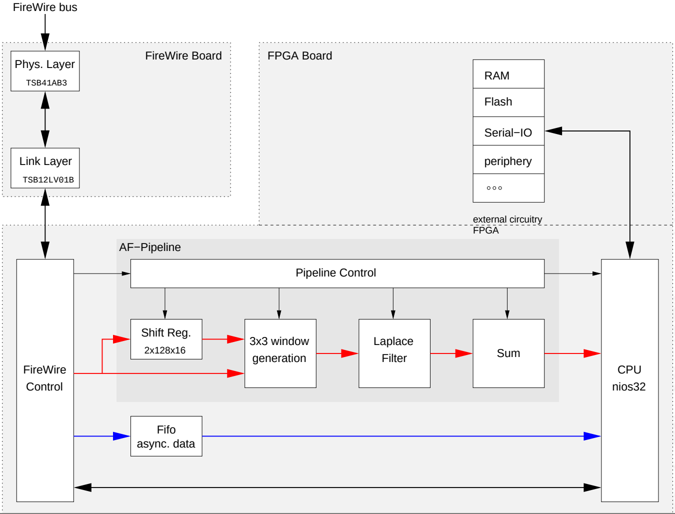
Sum: 465.479 — Focus optimized

Automatic focus (cont.)

Implementation details

- YUV 4:2:2 video data format:
  two adjacent pixels processed per cycle
- Hardware computation of focus quality — Pipeline
  - 3×3 window generation
  - Laplace filter, simple coefficients (+1, -4)
  - Sum up filtered values
- Control algorithm on embedded cpu
  - Code could be transferred without changes
  - Interrupt-driven

Automatic focus (cont.)



Digital filter

- Extend the architecture to *send* isochronous FireWire data
- Bottleneck: interface to link layer circuit
  - Proper scheduling between send and receive requests
  - Buffers are added (IP-Components)
- Datapath implementation depends on
  - the window size of the filter, determines # buffered lines
  - fixed or variable coefficients
  - the arithmetic used

⇒ FPGA prototype board has the flexibility to tune these parameters for optimal performance

Digital filter

- Extend the architecture to *send* isochronous FireWire data
- Bottleneck: interface to link layer circuit
  - Proper scheduling between send and receive requests
  - Buffers are added (IP-Components)
- Datapath implementation depends on
  - the window size of the filter, determines # buffered lines
  - fixed or variable coefficients
  - the arithmetic used

⇒ FPGA prototype board has the flexibility to tune these parameters for optimal performance

Digital filter

- Extend the architecture to *send* isochronous FireWire data
- Bottleneck: interface to link layer circuit
  - Proper scheduling between send and receive requests
  - Buffers are added (IP-Components)
- Datapath implementation depends on
  - the window size of the filter, determines # buffered lines
  - fixed or variable coefficients
  - the arithmetic used

⇒ FPGA prototype board has the flexibility to tune these parameters for optimal performance

Digital filter

- Extend the architecture to *send* isochronous FireWire data
- Bottleneck: interface to link layer circuit
  - Proper scheduling between send and receive requests
  - Buffers are added (IP-Components)
- Datapath implementation depends on
  - the window size of the filter, determines # buffered lines
  - fixed or variable coefficients
  - the arithmetic used

⇒ FPGA prototype board has the flexibility to tune these parameters for optimal performance

Omnidirectional to panoramic image conversion

Example

Omnidirectional image conversion (cont.)

Algorithm

- Store omnidirectional image: $Q_{x,y}$
- The panoramic image $P_{x,y}$ computes as

$$Q_x = P_y \cdot sin(a \cdot P_x) + x_0 \quad (1)$$

$$Q_y = P_y \cdot cos(a \cdot P_x) + y_0 \quad (2)$$

- Interpolate between adjacent pixels Q

Omnidirectional image conversion (cont.)

Additional hardware requirements

- Large images $\approx$ 2.4 MB + memory for computed image
  - $\Rightarrow$ external memory (SODIMM connector)
  - $\Rightarrow$ DRAM controller

- different images handled by the hardware
  1. Omnidirectional image: input from FireWire bus
  2. Omnidirectional image: output during computation
  3. Panoramic image: input from computation
  4. Panoramic image: output to FireWire bus
  - $\Rightarrow$ address switching techniques to partition the memory

- Sine/Cosine computation
  - $\Rightarrow$ Look-up table ROM (IP-component)

Omnidirectional image conversion (cont.)

Additional hardware requirements

- Large images ≈ 2.4 MB + memory for computed image
  - ⇒ external memory (SODIMM connector)
  - ⇒ DRAM controller

- different images handled by the hardware
  1. Omnidirectional image: input from FireWire bus
  2. Omnidirectional image: output during computation
  3. Panoramic image: input from computation
  4. Panoramic image: output to FireWire bus

  ⇒ address switching techniques to partition the memory

- Sine/Cosine computation
  - ⇒ Look-up table ROM (IP-component)

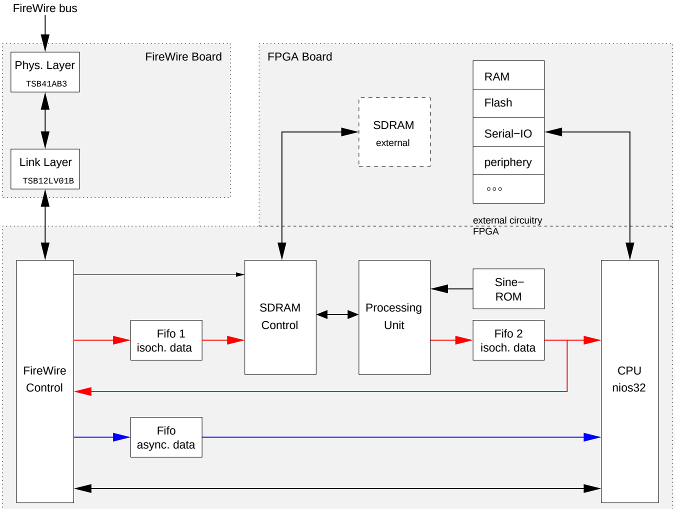
Omnidirectional image conversion (cont.)

Additional hardware requirements

- Large images $\approx$ 2.4 MB + memory for computed image
  - $\Rightarrow$ external memory (SODIMM connector)
  - $\Rightarrow$ DRAM controller
- different images handled by the hardware
  1. Omnidirectional image: input from FireWire bus
  2. Omnidirectional image: output during computation
  3. Panoramic image: input from computation
  4. Panoramic image: output to FireWire bus

  $\Rightarrow$ address switching techniques to partition the memory
- Sine/Cosine computation
  - $\Rightarrow$ Look-up table ROM (IP-component)

Omnidirectional image conversion (cont.)



Components

Hardware

Smart-Camera: Basler eXcite exA1390-19c

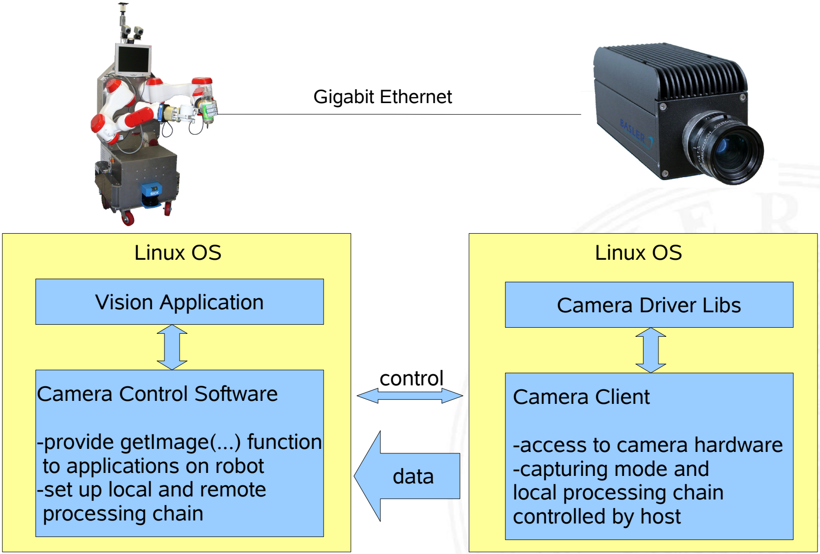
- Colour camera
  - 1388 × 1038 pixel
  - 19 frames/sec.
- Embedded computer
  - Linux OS
  - 1 MHz MIPS processor
  - 128 MB RAM, 128 MB Flash
  - Gigabit ethernet

Components

Software environment

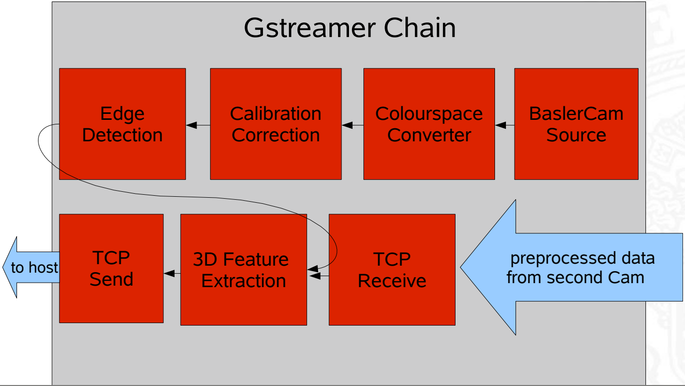
GStreamer framework

- Open source multimedia framework
- Predefined data handling functions
  - format conversion
  - resizing
  - encoding
- Timing issues
- Network data transmission
- Plugin mechanism for *user expansions*
  - Integration of image processing libraries, e.g. OpenCV


Gigabit Ethernet
Linux OS
Vision Application
Camera Control Software
-provide getImage(...) function
to applications on robot
-set up local and remote
processing chain
control
data
Linux OS
Camera Driver Libs
Camera Client
-access to camera hardware
-capturing mode and
local processing chain
controlled by host

System integration

- Transfer image processing tasks from Control PC to camera
- GStreamer for data transport (network transparent)
- sample setup computing depth information with two cameras



Results

A novel architecture must provide

- the flexibility of a software system and
- the performance of signal processors or dedicated hardware data paths,
- integrated into a framework, which allows the implementation of vision algorithms independently from the underlying hardware resources ⇒ Software- and Hardware-Libraries.

⇒ Further development:

Integrate of the features of the two prototyping systems

Results

A novel architecture must provide

- the flexibility of a software system and
- the performance of signal processors or dedicated hardware data paths,
- integrated into a framework, which allows the implementation of vision algorithms independently from the underlying hardware resources ⇒ Software- and Hardware-Libraries.

⇒ Further development:

Integrate of the features of the two prototyping systems

Results

A novel architecture must provide

- the flexibility of a software system and
- the performance of signal processors or dedicated hardware data paths,
- integrated into a framework, which allows the implementation of vision algorithms independently from the underlying hardware resources ⇒ Software- and Hardware-Libraries.

⇒ Further development:

Integrate of the features of the two prototyping systems

Results

A novel architecture must provide

- the flexibility of a software system and
- the performance of signal processors or dedicated hardware data paths,
- integrated into a framework, which allows the implementation of vision algorithms independently from the underlying hardware resources ⇒ Software- and Hardware-Libraries.

⇒ Further development:

Integrate of the features of the two prototyping systems

Results

A novel architecture must provide

- the flexibility of a software system and
- the performance of signal processors or dedicated hardware data paths,
- integrated into a framework, which allows the implementation of vision algorithms independently from the underlying hardware resources ⇒ Software- and Hardware-Libraries.

⇒ Further development:

Integrate of the features of the two prototyping systems

Conclusion

- Both hard- and software architectures for a flexible and extendible vision subsystem have been presented
- Experimental workbenches for system architectures
- FPGA-based system
  - Automatic focus for active stereo-vision system
  - Digital filter
  - Omnidirectional to panoramic image conversion
  - ⇒ HW-/SW-CoDesign strategy
- Smart-camera system
  - Streaming media framework integration (GStreamer)
  - Using optimized image processing library (OpenCV)
  - ⇒ Software architecture

Thank you for your attention...

Andreas Maeder – Hannes Bistry – Jianwei Zhang
maeder | bistry | zhang @informatik.uni-hamburg.de

TAMS

University of Hamburg
Faculty of Mathematics, Informatics and Natural Sciences
Department of Informatics
Technical Aspects of Multimodal Systems