On the Architectural Design and Applications of CMOS Vision Systems

Carlos Mendoza Giraldo

IEEE CAS Workshop 2009
Rio de Janeiro, November 24
Belo Horizonte, November 26
Outline of the Talk

• Some Basic Concepts:
  ◊ Concept of Vision System
  ◊ Conventional Vision System Architecture
  ◊ Rationale for Using CMOS

• Architectures for Vision Systems
  ◊ Progressive Distributed Processing
  ◊ Bio-Inspired Architectures
  ◊ Shifting the Analogue-to-Digital Border

• Topographical Sensor-Processors
  ◊ Concept of Concurrent Sensory-Processing
  ◊ Functional Requirements
  ◊ Multi-Functional Pixel

• The Eye-RIS Vision System
  ◊ The Q-Eye and the Eye-RIS Architecture
  ◊ The Eye-RIS Family
  ◊ Programming Platform
  ◊ Sample Applications
  ◊ The Eye-RIS Image Processing Library
Some Basic Concepts

Concept of Vision System

Conventional Vision System Architecture

Rationale for Using CMOS
Some Basic Concepts

Concept of Vision System

Conventional Vision System Architecture

Rationale for Using CMOS
A Vision System analyzes a sequence of images and makes decisions.

Current Vision Systems:
- Bulky, power hungry
- Inefficient
Illustrating the Challenge of Speed I

Systems to close the **Perception-to-Action Loop at Thousands Images-per-Second**

Spinning at 300 codes/sec

1. Code under recognition
2. Start code
3. Spinning at 300 codes/sec
4. Code under recognition
Systems to close the Perception-to-Action Loop at Thousands Images-per-Second

Illustrating the Challenge of Speed II

Speed ~ 40 m/s

Laser system

Illumination system

Up to 3,000 frames per second
Illustrating the Challenge of Compactness

Compact systems that can be installed in small places and include all the needed functionality

3 persons detected  1 abandoned object  1 object removed
Systems capable of making **autonomous decisions for vision-based guidance**

Illustrating the Challenge of Autonomy

- Voltage regulator board
- Motor controller board
- Battery
Goals of Artificial Vision Systems

- **Perceive lightness and colour under various illuminations**
- **Detect intensity changes and perform 2-D segmentation**
- **Infer 3-D structures from stereo or motion images**
- **Organize surface and regions into objects of interest**
- **Generate description of objects and recognize them among a potentially large class of objects**
- **Make non-visual inferences about the scene based on visual processing (abstraction)**

[Emulate the capabilities of human visual systems]

[H.R. Myler, Fundamentals of Machine Vision]
Some Basic Concepts

Concept of Vision System

Conventional Vision System Architecture

Rationale for Using CMOS
The Vision Processing Chain

- Huge amount of data at Early Stages
- Most data is useless
Illustrating the Computational Demands

Per-Pixel 3x3 Linear Convolution

9 Products + 8 Sums = 17 Operations

If the image is $N \times M$:

$17 \times N \times M$ Oper.

If the algorithm contains $N_s$ convs.:

$17 \times N_s \times N \times M$ Oper.

At $F$ Frames per second:

$17 \times F \times N_s \times N \times M$ OPS

10 convolutions on 8-bit gray-scale VGA images at 100 FPS requires 5 GOPS

Memory operations must be accounted for as well

[Original from Gustavo Liñán]
Conventional Vision System Architecture

- Conventional systems are heterogeneous
- Sensors, Memory, Processors, Communications
- Involve multiple technologies

Data Flow:

1. CMOS / CCD sensor
2. High-performance A/D Converters
3. Advanced DSP & High Density Memory

Image Acquisition
Image Coding
Image Processing and Storage
Troubles of Conventional Vision Systems
Architectures

- **Physical separation:**
  - Sensors
  - Processors

- **Heterogeneous technologies**
  - CCD or CMOS for sensing
  - CMOS: FPGA, Power-PC, DSP, etc for processing

- **Bottlenecks at different stages**
  - Image coding
  - Image transmission
  - Image processing
Some Basic Concepts

Concept of Vision System

Conventional Vision System Architecture

Rationale for Using CMOS
CMOS Image Sensors have some important advantages that make them very suitable for Vision Systems

- Homogeneous systems
- Lower power consumption
- Low cost
- Random pixel access
- Allow additional circuitry in the pixel: can be used for processing!
The Concept of CMOS Camera On-Chip

- **On-chip Electronics**
- **Low-Cost**
- **Low-Power**
- **Random Access**...

**PIXEL ARRAY**

**READ-OUT MODES**
- Progressive scan
- Window read-out
- Skip read-out (subsampling)

**Row Select Logic**

**Column Select Logic**

**Timing and Control**

**Analog Signal Processing**

**A/D Converters**

- Charge integration
- Gain
- Sample and hold
- Correlated-double-sampling
- FPN suppression

Digital Output

Column-parallel ADC or serial ADC

[E. Fossum, 1997]
All functions for smart imaging and vision systems on chip
Architectures for Vision Systems

Progressive Distributed Processing

Bio-Inspired Architectures

Shifting the Analogue-to-Digital Border
Architectures for Vision Systems

Progressive Distributed Processing

Bio-Inspired Architectures

Shifting the Analogue-to-Digital Border
Conceptual Architectural Choices

Conventional Architecture

- Large spatial resolution limited only by pixel SNR
- Large circuit operation predictability and robustness analog only at the ADCs
- Small spurious signal interactions
- Large flexibility and programmability:
  - mostly digital circuits and codes
- Small computational power and efficiency
- Large memory requirements
- Data transfer bottlenecks
## Conceptual Architectural Choices

### Linear Array of Digital Processors

- **Smaller spatial resolution**: trade-off with processing
- **Large circuit operation predictability and robustness**: analog only at the ADCs
- **Small spurious signal interactions**
- **Large flexibility and programmability**: mostly digital circuits and codes
- **Larger computational power and efficiency**
- **Smaller memory requirements**
- **Data transfer bottlenecks**

<table>
<thead>
<tr>
<th>Sensors + Analog Signal Read-out</th>
<th>A/D Conversion</th>
<th>Distributed Digital: LAP</th>
<th>Digital</th>
</tr>
</thead>
</table>

© ANAFOCUS 2009
Conceptual Architectural Choices

Linear Array of Analogue-Digital Processors

- Smaller spatial resolution: trade-off with processing
- Smaller circuit operation predictability and robustness: analog also for processing
- Larger spurious signal interactions
- Smaller flexibility and programmability: both analog and digital circuits
- Larger computational power and efficiency
- Smaller memory requirements
- Smaller transfer bottlenecks

Diagram:
- Sensors + Analog Signal Read-out
- Distributed Analog + A/D Conversion
- Distributed Digital: LAP
- Digital
Architectures for Vision Systems

Progressive Distributed Processing

Bio-Inspired Architectures

Shifting the Analogue-to-Digital Border
How Does The Retina Work?

- Sensors and processors are merged
- Processing and sensing are simultaneous
- Significant data compression is achieved
How Does The Retina Work? II

[Roska & Werblin 2001]
Conceptual Architectural Choices

Topographic ADCs + Digital Processing

- Smaller spatial resolution: trade-off with processing
- Larger system operation predictability and robustness: analog only for ADCs
- Smaller spurious signal interactions
- Larger flexibility and programmability:
  - only digital processing
- Functionality and efficiency compromised by ADC accuracy
- Large in-pixel memory requirements
Conceptual Architectural Choices

**Topographic Mixed-Signal Processing**

- Low spatial resolution: trade-off with processing
- Involved circuit operation predictability and robustness: analog also for processing
- Larger spurious signal interactions
- Involved flexibility and programability: both analog and digital circuits
- Large computational power and efficiency
- Small memory requirements
- Small transfer bottlenecks
Using the Bio-Inspiration Concept

Data Flow

CMOS / CCD sensor

Image Acquisition

High-performance A/D Converters

Image Coding

Advanced DSP & High Density Memory

Image Processing and Storage

Focal-Plane Sensor-Processor

Image Acquisition & Pre-processing

Low-Profile A/D Converters

Pre-processed Image Conversion

Simple DSP & Low Memory

Image Post-Processing and Storage

F

control signals

f << F

control signals

f

F

RAM

ADC

control signals

ADC

f

RAM

ADC

control signals

ADC

f

RAM
Architectures for Vision Systems

Progressive Distributed Processing

Bio-Inspired Architectures

Shifting the Analogue-to-Digital Border
Shifting the Analogue-Digital Border

<table>
<thead>
<tr>
<th>Task n°</th>
<th>Processing Task</th>
<th>Example</th>
<th>Amount of data</th>
</tr>
</thead>
<tbody>
<tr>
<td>①</td>
<td>Image Acquiring</td>
<td>Light sensing</td>
<td>b×N²</td>
</tr>
<tr>
<td>②</td>
<td>Noise reduction</td>
<td>Lowpass, median filtering...</td>
<td>b×N²</td>
</tr>
<tr>
<td>③</td>
<td>Segmentation</td>
<td>Edge detection</td>
<td>b×N²</td>
</tr>
<tr>
<td>④</td>
<td>Feature extraction</td>
<td>shape detection</td>
<td>N²</td>
</tr>
<tr>
<td>⑤</td>
<td>Classification</td>
<td>identification</td>
<td>N</td>
</tr>
</tbody>
</table>

**Sensors + Signal Conditioning**

**A/D Conversion**

Digital

N= image width & height
b= number of bits of resolution
Shifting the Analogue-Digital Border

<table>
<thead>
<tr>
<th>Task n°</th>
<th>Processing Task</th>
<th>Example</th>
<th>Amount of data</th>
</tr>
</thead>
<tbody>
<tr>
<td>①</td>
<td>Image Acquiring</td>
<td>Light sensing</td>
<td>(b \times N^2)</td>
</tr>
<tr>
<td>②</td>
<td>Noise reduction</td>
<td>Lowpass, median filtering...</td>
<td>(b \times N^2)</td>
</tr>
<tr>
<td>③</td>
<td>Segmentation</td>
<td>Edge detection</td>
<td>(N^2)</td>
</tr>
<tr>
<td>④</td>
<td>Feature extraction</td>
<td>shape detection</td>
<td>(N)</td>
</tr>
<tr>
<td>⑤</td>
<td>Classification</td>
<td>identification</td>
<td>1</td>
</tr>
</tbody>
</table>

\(b = \)number of bits of resolution


A/D Conversion

Digital
Topographical Sensor-Processors

Concept of Concurrent Sensory-Processing

Functional requirements

Multi-Functional Pixel
Topographical Sensor-Processors

Concept of Concurrent Sensory-Processing

Functional requirements

Multi-Functional Pixel
Feature 1: Acquisition and processing are done on the focal plane
Feature 2: Multiple representations of a scene
Topographical Sensor-Processors

Concept of Concurrent Sensory-Processing

Functional requirements

Multi-Functional Pixel
**Pixel-Wise “Cosmetic Operations”:** Each pixel is transformed independently of its neighbors, but remains on the same location.
Pixel-Wise “Cosmetic Operations”: Each pixel is transformed independently of its neighbors, but remains on the same location.

**Generalized Convolutions**: Each pixel is transformed as a combination of the pixels within its neighborhood:

1) Linear Convolution Kernels
Pixel-Wise “Cosmetic Operations”: Each pixel is transformed independently of its neighbors, but remains on the same location.

Generalized Convolutions: Each pixel is transformed as a combination of the pixels within its neighborhood:

1) Linear Convolution Kernels

2) Morphological Operators

[Original from Gustavo Liñán]
Pixel-Wise “Cosmetic Operations”:

Each pixel is transformed independently of its neighbors, but remains on the same location.

Generalized Convolutions:

Each pixel is transformed as a combination of the pixels within its neighborhood:

1) Linear Convolution Kernels

2) Morphological Operators

3) Nonlinear Operations; ... anisotropic diffusion, median filtering, etc
**Pixel-Wise “Cosmetic Operations”:** Each pixel is transformed independently of its neighbors, but remain on the same location.

**Convolutions:** Each pixel is transformed as a combination of the pixels within its neighborhood:

**Movements:** Pixels are moved to a different position. Movements can be decomposed into shifts and rotations.
Pixel-Wise “Cosmetic Operations”: Each pixel is transformed independently of its neighbors, but remain on the same location.

Convolutions: Each pixel is transformed as a combination of the pixels within its neighborhood.

Movements: Pixels are moved to a different position. Movements can be decomposed into shifts and rotations.

Image-wise Operations: Pixels in different images in the same or different locations are combined (either linearly or non-linearly).

[Original from Gustavo Liñán]
Topographical Sensor-Processors

Concept of Concurrent Sensory-Processing

Functional requirements

Multi-Functional Pixel
Multi-Functional Pixel Example

Column

To neighbours

Multipliers

Registers

Optical Module

Input Block

Σ

From neighbours

DataBus

Global Control Bus

Local Control Bus

I/O

Fast Averag-

Binary Operator

Local Masks

Address Event

AE decoder
The more processing/storage you place in the pixel the lower will be the maximum achievable resolution for a given optical size.

Finding the **optimum partitioning** between pixel-wise, column-wise and system-level processing is key to devise an efficient processing sensor.
The Resolution Compromise

Size of Sensory Pixel

Size of Sensory-Processing Pixel

How Important is Resolution?
How Important is Resolution?

- Vision is possible with 25 x 25 “pixels” (within limited field of view)
- Text can be read at 200 words/min (300 words/min with normal vision)
- Students can navigate in complex environments (maze) with confidence

Resolution can be increased through proper design

- VGA and up to 1.3Mpixels resolution for Surveillance
- > 1Mpixels for Machine Vision and Automotive
The Eye-RIS Vision System

The Q-Eye and the Eye-RIS Architecture

Eye-RIS Family

Programming Platform

Sample Applications

The Eye-RIS Image Processing Library
The Eye-RIS Vision System

The Q-Eye and the Eye-RIS Architecture

Eye-RIS Family

Programming Platform

Sample Applications

The Eye-RIS Image Processing Library
## The Q-Eye Chip

<table>
<thead>
<tr>
<th>I/O control unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>I/O ADCs</td>
</tr>
<tr>
<td>I/O DACs</td>
</tr>
<tr>
<td>S&amp;H Banks</td>
</tr>
</tbody>
</table>

## Processing & Sensing Array

- **UMC 0.18µm CMOS 1P5M (1.8V/3.3V) - mixed-signal.**
- **High-performance Smart Image Sensor**
  - 176 x 144 grey-scale pixels with 29.1µm pitch
  - High-speed global electronic shutter. Programmable exposure time (controlling step-down to 20ns)
  - Approximated sensitivity of 3V/lux-sec at 550nm
  - 7 +1 (two banks) high-retention analogue and 7 binary memories
  - Analogue multiplexer for image shifting
  - Analogue MAC unit
  - Programmable, 3 x 3 neighbourhood pattern matching with 1/0/d.n.c.pattern definition (fast morphological functions)
- **On-chip bank of high-speed ADCs and DACs.**
- **Multiple I/Os and high-speed communication ports**
Configurable system which provides an efficient solution for low-to-medium resolution and high frame rates applications

**AnaFocus**

Q-Eye
Mixed-signal focal-plane early processor

Data - Memory (SRAM)

32bit RISC Microprocessor

Program - Memory (SRAM)

Ethernet / PCI / USB 2.0

Off-chip memory Controller

High-speed Bus

Low-speed Bus

Multi-channel DMA controller

Timing, PLL, General I/O, …

Bus Bridge

Ultra-high frame rate: up to 8,000fps@QCIF image resolution

Ultra-low power consumption: < 10mW@30fps@QCIF image resolution

© ANAFOCUS 2009
The Eye-RIS Vision System is made up of two processors:

- **SIS Q-Eye (Smart Image Sensor Q-Eye)**
  - Image acquisition
  - Pre-processing tasks
  - Input are images
  - Output are some particular characteristics of those images

- **Altera’s Nios II**
  - Input is information about the images
  - Post-processing tasks
  - Output are decisions based on that information
The structure of a an Eye-RIS Application follows scheme of a tipical vision application

1. There is a main loop
2. Initialization (Both processors)
3. Optical acquisition (SIS Q-Eye)
4. Preprocessing (SIS Q-Eye)
5. Postprocessing (SIS Q-Eye and maybe Nios II)
6. Decision making and actuation (Nios II)

- The Nios II uses the SIS Q-Eye to accomplish the corresponding tasks

```c
int main()
{
    // Init
    ...
    while(1)
    {
        // Optical acquisition
        ...
        // preProcessing
        ...
        // postProcessing
        ...
        // Decision making
        // and actuation
        ...
    }
    return 1;
}
```
The execution flow between the two processors is the key point to understand how the Eye-RIS Vision System works.

- Execution is split between both processors.
- Nios II leads the execution -> the execution starts in the “main” function.
- Nios II decides when to execute a section in the SIS Q-Eye code using function `Section_execute`.

```c
... Section_execute(S1);
...
Section_execute(S3);
...

void section S1 ()
{
  ...
}
...
void section S2 ()
{
  ...
}
...
void section S3 ()
{
  ...
}
```

Execution in:
- Nios II
- SIS Q-Eye
Both processors can work in parallel mode, executing its respective codes at the same time

- Replace `Section_execute` for `Section_executeAsync` and `Section_wait`
- `Section_executeAsync` launches the execution of a CFPP code section, but returns immediately, without waiting for the section execution to end
- `Section_wait` waits for the section execution to end. It must be called to synchronize Nios II and SIS Q-Eye
Sharing data is the way the Nios II and the SIS Q-Eye can work together to accomplish the tasks

There are three ways to share data:

1. sections arguments and return value
2. CFPP global variables
3. Sharing images: the Eye-RIS Image Memory
CFPP sections can have arguments and a return value

- Declare the section to receive the arguments and return a value:

```cpp
int section SampleSection (int arg)
{
    // Section code
    ...
    return retval;
}
```

- Add the parameters and read the return value in the call to `Section_execute` or `Section_executeAsync`:

```cpp
int main ()
{
    int par;
    int result;
    ...
    result = Section_execute(SampleSection, par);
    ...
}
```
**CFPP global variables can be accessed from the C/C++ code**

- Declare the global variable in CFPP code:

  ```
  int g_a;
  time g_t;
  ...
  void section SampleSection ()
  {
    ...
  }
  ```

- Declare the global variables as extern and use them in C/C++ code:

  ```
  extern fpp_int g_a;
  extern fpp_time g_t;
  int main ()
  {
    int a = g_a;
    g_a = 3;
    fpp_time t = FPPTime_read(g_t);
    FPPTime_write(g_t, 20000);
  }
  ```

  - **fpp_int** variables can be used directly
  - **fpp_time** variables must be accessed using macros
Images are the most important information that's exchanged in the system

- The Eye-RIS Image Memory is shared between the SIS Q-Eye and the Nios II processor
- The Nios II processor operates on the images in the Image Memory
- SIS Q-Eye does not operate on images in the Image Memory, only loads/downloads images and points coordinates from/to it
The Image memory is an important part of the system

- Is statically allocated in pages, one for each image
- Divided in two areas:
  1. 440 grey level images
  2. 512 binary images
- Images are referenced by its page index
The Eye-RIS Vision System

The Q-Eye and the Eye-RIS Architecture

Eye-RIS Family

Programming Platform

Sample Applications

The Eye-RIS Image Processing Library
Main Features

Q-Eye focal-plane processor
- 176 x 144 spatial resolution
- Monochrome image sensor with 3.2V/(lux∙sec) sensitivity
- Maximum frame rate (sensing + processing) of over 8,000fps
- Advanced pixel architecture combining image acquisition, image processing and storage:
  - Multi-mode image sensing, analogue & binary memories, analogue multiplexor for image shifting, analogue MAC unit, programmable LUT, resistive grid for controllable smoothing...

Digital control & post-processing
- ALTERA NIOS-II 32-bit RISC microprocessor
- 1.17 DMIPS/MHz performance at 70MHz operation frequency
- 32Mb SDRAM for program and image/data storage

Communications
- SPI port, UART, 2xPWM ports and GPIOs, USB 2.0, and GigE
- JTAG Controller
- 1.5W typical power consumption

Eye-RIS ADK (Application Development Kit)
- Application Development Kit including: Project builder, C compiler, assembler, linker, and source-code debugger.
- Image-processing library including basic routines such as: point-to-point operations, spatial filtering operations, morphological operations and statistical operations
Main Features

Q-Eye Smart Image Sensor
- 176 x 144 spatial resolution
- Monochrome image sensor with 3.2V/(lux∙sec) sensitivity
- Maximum frame rate (sensing + processing) of over 8,000fps
- Advanced pixel architecture combining image acquisition, image processing and storage

Digital control & post-processing
- ALTERA NIOS-II 32-bit RISC microprocessor
- 1.17 DMIPS/MHz performance at 70MHz operation frequency
- 64 + 64Mb DDR2 for program and image/data storage (expandable up to 256 + 256Mb)
- Multi Layer Perceptron
- Digital Image Coprocessor

Communications
- UART, 2xPWM ports and GPIOs and GigE
- JTAG Controller
- 1.5W typical power consumption

Eye-RIS ADK (Application Development Kit)
- Application Development Kit including: Project builder, C compiler, assembler, linker, and source-code debugger.
- Image-processing library including basic routines such as: point-to-point operations, spatial filtering operations, morphological operations and statistical operations

Existing standard products: Eye-RIS™ v1.3
Main Features

**On-chip Smart Image Sensor**
- 176 x 144 spatial resolution
- Monochrome image sensor with 1V/(lux∙sec) sensitivity
- Maximum frame rate (sensing + processing) of over 8,000fps
- Advanced pixel architecture combining image acquisition, image processing and storage:
  - Multi-mode image sensing, analogue & binary memories, analogue multiplexor for image shifting, analogue MAC unit, programmable LUT, resistive grid for controllable smoothing...

**On-chip digital control & post-processing**
- NIOS-II microprocessor with 1.17 DMIPS/MHz at 100MHz
- On-chip 128kB image storage, 256kB program and data memory. External SRAM interface up to 2GB.
- 8MB of Flash EPCS for program and data storage.

**On-chip communication I/Os**
- SPI port, UART, 2xPWM ports and GPIOs, USB 2.0 interface, and
- JTAG Controller
- <700mW typical power consumption

**Eye-RIS ADK (Application Development Kit)**
- *Application Development Kit* including: Project builder, C compiler, assembler, linker, and source-code debugger.
- *Image-processing library* including basic routines such as: point-to-point operations, spatial filtering operations, morphological operations and statistical operations

Existing standard products: **Eye-RIS™ v2.1**
Existing standard products: **Eye-RIS™ v2.1**

Chip Layout

Q-Eye Focal-plane processor

NIOS-II uP

Memory

Memory

Chip photography

System view
The Eye-RIS Vision System

The Q-Eye and the Eye-RIS Architecture

Eye-RIS Family

Programming Platform

Sample Applications

The Eye-RIS Image Processing Library
Eye-RIS vision systems are programmed using the Eye-RIS ADK (Application Development Kit), a complete development environment based on the open and extensible “eclipse” platform.

ADK includes all the necessary tools to develop and debug applications on Eye-RIS vision systems:

- Project creation wizards
- Project management
- Built-in editors
- Output consoles
- Real-time image viewer
- Launcher
- Debugger
- On-line help

Programming Platform: Eye-RIS ADK
The creation of Eye-RIS applications follows a typical developing scheme:

1. **Create a Project**
2. **Write the Application code**
3. **Compile the Project**
4. **Execute and/or debug the Project**
5. **Make it a stand-alone application**
Running a project means downloading it to the Eye-RIS Vision System and launching it

- To run a project, right click on the project and select “Run as -> Eye-RIS Application”
- The “.elf” file is loaded into the Eye-RIS memory and executed
Each processor is programmed in a different language

- SIS Q-Eye™ is programmed using CFPP code
- Nios II is programmed using C/C++ code
The CFPP code is a proprietary language developed in Anafocus to program the SIS Q-Eye

Its main characteristics are:

1. C like syntax for a quick learning-curve

2. Special types: time, lam, template, mpattern, etc...
   - This types are built in types that represent Q-Eye specific parameters

3. Structured in functions and sections. A section is a particular type of function that can be called from Nios code

4. There are no arrays, although some special types may look like arrays

5. There are no multiplications nor divisions
The Eye-RIS ADK provides a set of libraries to enhance the development of Eye-RIS applications

**SIS Q-Eye (CFPP code)**
- Image Processing Library (IPL)

**Nios II (C/C++ code)**
- Extended Image Processing Library (Extended IPL)
- Eye-RIS Basic Library (EBL)

![Diagram showing the libraries provided by SIS Q-Eye and Nios II](image-url)
The IPL allows the user to take advantage of the sensing and processing capabilities of the SIS Q-Eye

It is focused on images and its main uses are:
- Image acquisition
- Filters
- Arithmetical operations
- Morphological and logic operations
- Coordinates extraction

The Extended IPL combines the SIS Q-Eye with the computing potential of the Nios II to extend the capabilities of the IPL

Its main uses are:
- Blobs processing
- Coordinates management
- Smart sensing
The Eye-RIS Basic Library eases typical Nios II tasks in Eye-RIS applications

Its main uses are:

- Control of the SIS Q-Eye: Section_execute, FPPTime_write, ...
- Displaying of images in the PC: Image_display, ...
- Loading/storing images from/on the PC: Image_write, Image_read
- Printing messages on the PC: printMessage
- Text input: scanMessage, ...
- Interrupts: Irq_enable, ...
- Timers: Timer_start, Timer_measure
The Eye-RIS Vision System

The Q-Eye and the Eye-RIS Architecture

Eye-RIS Family

Programming Platform

Sample Applications

The Eye-RIS Image Processing Library
Sample Applications: High Speed Multitracking

High speed Multitracking

For downloading demo video please go to http://www.anafocus.com/videos/mt.mov
High speed Multitracking

Image acquisition | Low-pass filtering | adaptive thresholding | motion estimation | object tracking | Loop control | position prediction | coordinates translation | actuation on the laser system.

Sample Applications: High Speed Multitracking II
Data Matrix Code (DMC) decoding

High-speed processing of 3,000 images-per-second to decode up to 200 data-matrix codes per second

For downloading demo video please go to http://www.anafocus.com/videos/dmc.mov
Sample Applications: DMC II

Data Matrix Code (DMC) decoding

Q-Eye

- Image Acquisition
- Image preprocessing
  - Code detected?
    - Yes: Alternating pattern
    - No: Continue
- Extreme points

NIOS

- Extract Data
- Decode
- ECC
- Synchronization

Sample Applications: DMC II

Data Matrix Code (DMC) decoding
High Dynamic Range: Well-Adjustment

Sample Applications: HDR I

For downloading demo video please go to http://www.anafocus.com/videos/mt.mov
Sample Applications: HDR II

High Dynamic Range: Multiple images average

Simple exposure-time control

HDR algorithm

Saturation due to excess of light
Lose of information due to lack of time
Sample Applications: Part Finding

- All steps of processing chain are performed in the Q-Eye
- Amount of data to be processed by the host digital processor is very small
Sample Applications: Human Computer Interface

Interactive Human Computer Interface

- Shoot (in game)
- Drag & drop
- Hand tracking
- Hand gesture recognition

Track body:
- Jump
- Crouch
- Move left
- Move right

Starting menu:
- 1. Play game
- 2. Image explorer

Sample Applications: Human Computer Interface

- Hand gesture recognition
- Body motion recognition
- Body gesture recognition

© ANAFOCUS 2009
**Smart Surveillance**

*Access control and network of intelligent surveillance cameras*

- 98% accuracy in access control under a wide range of situations
- Automatic and efficient adaptation to illumination conditions
- Capability to detect changes in the background
- Equipped with GigE connection to create networks with distributed intelligence

For downloading demo video please go to [http://www.anafocus.com/videos/surveillance.mov](http://www.anafocus.com/videos/surveillance.mov)
Sample Applications: Smart Surveillance II

Smart Surveillance

Q-Eye

1. Init
2. Background acquisition
3. Image acquisition
4. Band-pass filtering
5. Motion detection
6. Background subtraction
   - Threshold
   - Morphological Filtering
7. People tracking
8. Objects detection
9. Threshold
10. Morphological Filtering
11. Show results

NIOS

Sample Applications: Smart Surveillance II
**Autonomous Vehicle**

*Systems capable of making autonomous decisions for vision-based guidance*

Image acquisition | Low-pass filtering | thresholding | skeletonization | morphological filtering | signal recognition | road-lanes recognition | actuation on the robot wheels.
Sample Applications: Autonomous Vehicle II

Autonomous Vehicle

- Q-Eye
  - Adjust exposure time
  - Optical acquisition
  - Adaptive Threshold
  - Skeleton
  - Corners detection
  - Morphological filtering

- NIOS
  - Obstacle detected?
    - Yes: Stop
    - No: SIGNAL RECOGNITION
      - Signal Detected?
        - Yes: Signal analysis
        - No: Road detection
          - Perspective transform
          - Track algorithm and speed control

- ROAD TRACKING

Sample Applications: Autonomous Vehicle II
The Eye-RIS Vision System

The Q-Eye and the Eye-RIS Architecture

Eye-RIS Family

Programming Platform

Sample Applications

The Eye-RIS Image Processing Library
Image codification inside the SIS Q-Eye takes sign into account
**IPL**

*Image acquisition functions:*

Sense_acquire
Sense_start
Sense_readSensor
Sense_end
Sense_noiseRemovalAndScale

**Parameters:**
Destination LAM
Exposure Time
Gain

**Beware:**
Dark Current Noise
MAC Errors
IPL

Image Moving Functions

Move_loadImage
Move_downloadImage

Move_moveImage
Move_shiftImage

Parameters:
Origin/Destination LAM
Origin/Destination Digital Memory
Number Of Shifted Pixels

Masks
Contour Conditions

Beware:
LAM Read Errors
Analog Image Shifter Errors

The Eye-RIS IPL: Image Moving
**IPL**

**Filters**

- Filter_gaussian
- Filter_convolution

- Filter_diffusion
- Filter_directional_diffusion

- Filter_average
- Filter_laplace
- Filter_sobel
- Filter_sharpen

**Parameters:**
- Origin/Destination LAM
- Saturation
- Mode
- Masks

**Beware:**
- LAM Read Errors
- MAC Errors
- Resistive Grid Errors
- Analog Image Shifter Errors

**Two approaches:**
- LPF + MAC operation.
- Convolution.

Sign info in horizontal Sobel

---

**Diagram:**

- Optical sensor
- Resistive grid
- LAMs
- Analog I/O
- MAC
- Analog Image shifter
- Analog-to-binary converter
- Binary-to-analog converter
- LLU
- HitAndMiss
- LDMs
- Digital I/O
- Active pixel I/O
- Binary Image shifter
**IPL Arithmetic Operations**

- Arith_setImageValue
- Arith_imageMean
- Arith_macOperation

**MAC Operation**

\[ V_{out} = V_{\text{offset}} + \text{Factor}(V_1 - V_2) \]

Factor = 1/2, 1, 2

**Parameters:**
- Origin/Destination LAM Masks

**Beware:**
- LAM Read Errors
- MAC Errors
- Signal Range
**IPL**

*Arithmetic Operations Examples*

\[
\text{Result} = \frac{1}{4} \times (V_1 + V_2 + V_3 + V_4)
\]
**IPL**

**Arithmetic Operations**

Arith_invert  
Arith_scale  
Arith_posDifference  
Arith_negDifference  
Arith_signedDifference

Arith_absDifference  
Arith_add

**Parameters:**  
Origin/Destination LAM  
Saturation

**Beware:**  
LAM Read Errors  
MAC Errors  
Signal Range
IPL
Arithmetic Operations Examples

Arith\_signedDifference

Arith\_negDifference

Arith\_absDifference

Arith\_posDifference

Arith\_add
The Eye-RIS IPL: Thresholding

IPL Threshold

Thresh\_threshold

Parameters:
Origin/Destination LAM
Threshold -> Local / Global

Beware:
LAM Read Errors
MAC Error

Local threshold

Global threshold

Optical sensor
Resistive grid
LAMs
Analog I/O
MAC
Analog Image shifter

Analog-to-binary converter

Binary-to-analog converter

LLU
HitAndMiss
LDMs
Digital I/O
Active pixel I/O
Binary Image shifter
**IPL**

**Binary Basics**

- Logic_not
- Logic_and
- Logic_or
- Logic_xor
- Logic_nand
- Logic_nor
- Logic_nxor
- Logic_andNot
- Logic_orNot
- Logic_notAnd
- Logic_notOr

**Morph_hitAndMiss**

**Parameters:**
- Origin/Destination LDM
- B/W/DNC
- Contour Conditions

**Hit & Miss Pattern [DNC, W, B, W, W, B, B, B, B]**

![Diagram](image)
The Eye-RIS IPL: Morphological Operations I

**IPL**

**Morphological operations**

- Morph_erode
- Morph_dilate
- Morph_open
- Morph_close
- Morph_thin
- Morph_thicken
- Morph_centroid
- Morph_skeleton
- Morph_convexHull
- Morph_removeSinglePoints
- Morph_endPoints
- Morph_prune
- Morph_skeletonJoints
- Morph_skiz

**Parameters:**
- Origin/Destination LDM
- Connectivity / SE
- Contour Conditions
- Number of iterations -> UNC

![Morphological Operations Diagram](image-url)
**IPL**

**Morphological operations**

Morph_floodFill
Morph_fillHoles
Morph_extractHoles
Morph_eraseBorderBlobs

**Parameters:**
Origin/Destination LDM
Connectivity
Contour Conditions

FloodFill with connectivity 8

FillHoles with connectivity 4

The Eye-RIS IPL: Morphological Operations II
IPL

Active Points functions

ActivePoints_load
ActivePoints_download
ActivePoints_activeRegion
ActivePoints_activityDetection

Parameters:
Origin/Destination LDM
Origin/Destination Digital Memory
**EIPL**

**Advanced Image Acquisition functions**

- Sense_acquire (lam destination, time exposureTime, int gain)
- Sense_adaptiveAcquire (lam destination, time initTime, int gain, int optimalMeanValue, int numberOfIterations)

- Sense_hdrAcquire(lam destination, fpp_time tExp, int percentage[6])
- Sense_hdrAdaptiveAcquire(lam destination, fpp_time initTime, int optimalMeanValue, int numberOfIterations, int percentage[6])
**EIPL**

**Active Points functions**

**IPL ActivePoints_download**
- `ActivePoints_readData` (int imageNumber, int firstPoint, int numberOfPoints, char *x, char *y)

**IPL ActivePoints_load**
- `ActivePoints_writeOnePoint` (int imageNumber, int pointIndex, char x, char y)
- `ActivePoints_writeData` (int imageNumber, int firstPoint, int numberOfPoints, char *x, char *y, int totalNumPoints)
- `ActivePoints_getPointer` (int imageNumber)

**IPL ActivePoints_activeRegion**
- `ActivePoints_getBoundingBox` (int imageNumber, char *xMin, char *yMin, char *xMax, char *yMax)
- `ActivePoints_contactPoints` (ldm orig, char *bb_xMin, char *bb_xMax, char *bb_yMin, char *bb_yMax, char *xUp0, char *xUp1, char *xDown0, char *xDown1, char *yLeft0, char *yLeft1, char *yRight0, char *yRight1)
- `int Counting_whitePointsAuto` (ldm orig)
- `int Counting_whitePointsFixed` (ldm orig, unsigned short exponent)
- `int Counting_whitePointsSeed` (ldm orig, unsigned short exponent, unsigned short numberOfIterations)
**EIPL**

**Blob Handling Library**

- Blobs_configure
- Blobs_get
- Blobs_recall
- Blobs_selectFeatures
- Blobs_removeFeatures
- Blobs_getFeatures
- Blobs_contactPoints
- Blobs_select
- Blobs_reconstruct
- Blobs_sortByFeature
- Blobs_erase
- Blobs_free
**EIPL**

**Blob Handling Library**

**Features**

- **Area**
- **Perimeter**
- **Bounding box**
- **Bounding Box Area**
- **Contact Points**
- **Extent** = Area/BB_Area
- **First Point**
- **Bounding Box Width and Height**
- **Convex Perimeter**
- **Convex Area**
- **Convexity** = Convex Perimeter/Perimeter
- **Compactness** = Perimeter^2/(4 π Area)
- **Number of Holes**
- **Euler Number** = Number of Blobs – Number of Holes
- **Length and Breadth**
  - Perimeter = 2(Length + Breadth)
  - Area = Length x Breadth
- **Elongation** = Length/Breadth

- **Intercept 0** [B F]
- **Intercept 45**
- **Intercept 90**
- **Intercept 135**
- **Intercept 180** [F,B]
- **Intercept 225**
- **Intercept 270**
- **Intercept 315**

**Centroid**

**Ellipse Major Axis**

\[
\sqrt{\frac{p^2}{2\pi}} + \frac{2a}{\pi} + \sqrt{\frac{p^2}{2\pi} - \frac{2a}{\pi}}
\]

**Ellipse Minor Axis**

\[
\sqrt{\frac{p^2}{2\pi} + \frac{2a}{\pi}} - \sqrt{\frac{p^2}{2\pi} - \frac{2a}{\pi}}
\]
EIPL

Blob Handling Library
Features Example

```c
Blobs_configure(&blobsExample, CONNECT_8, BINARY, numberOfBlobs);

Blobs_selectFeatures(&blobsExample, numberOfFeatures, AREA, PERIMETER, NUMBER_OF_HOLES);

Blobs_get(&blobsExample, LDM_2);

Blobs_getFeatures(&blobsExample, ALL_FEATURES);

Blobs_free(&blobsExample);
```
**EIPL**

**Blob Handling Library Example**

```c
Blobs_configure(&blobsExample, CONNECT_4, BINARY, 8);

Blobs_selectFeatures(&blobsExample, numberOfFeatures, AREA, PERIMETER);

GetBlobs(&blobsExample, LDM_2);

Blobs_getFeatures(&blobsExample, 1, AREA);

Blobs_select(&blobsExample, EXCLUDE, AREA, LESS, AREA_MIN, 0);

Blobs_getFeatures(&blobsExample, PERIMETER);

Blobs_sortByFeature(&blobsExample, PERIMETER, DESCENDING, orden);

Blobs_recall(&blobsExample, LDM_3, orden[k]+1);

Blobs_free(&blobsExample);
```
Conclusions

➔ We can take advantage of nature through understanding

 Parallel and concurrent sensory-processing

 Multi-scale representation

➔ Smart CMOS sensors are not common yet

A real challenge for engineering !!

➔ Close interaction and collaboration between analog and digital circuitry is needed for efficient VSoC design.

➔ VSoCs are still like science fiction toys for industry