ARCam: an FPGA-based Augmented Reality Framework

JOÃO PAULO LIMA
Germano Guimarães
Guilherme Silva

JOÃO MARCELO TEIXEIRA
Emanoel Xavier
Veronica Teichrieb
Judith Kelner

Petrópolis, May 2007

Motivation

- Many AR tracking techniques require image processing procedures
- Usually done by software
  - General purpose processing
  - Operating system overhead
  - Impact on frame rate and image resolution
  - Increase on clock frequency and power consumption

Motivation

- Solution: embedded image processing
  - Dedicated hardware
  - Better performance
  - Real parallelism
  - Low power consumption
  - High resolution images

Goals

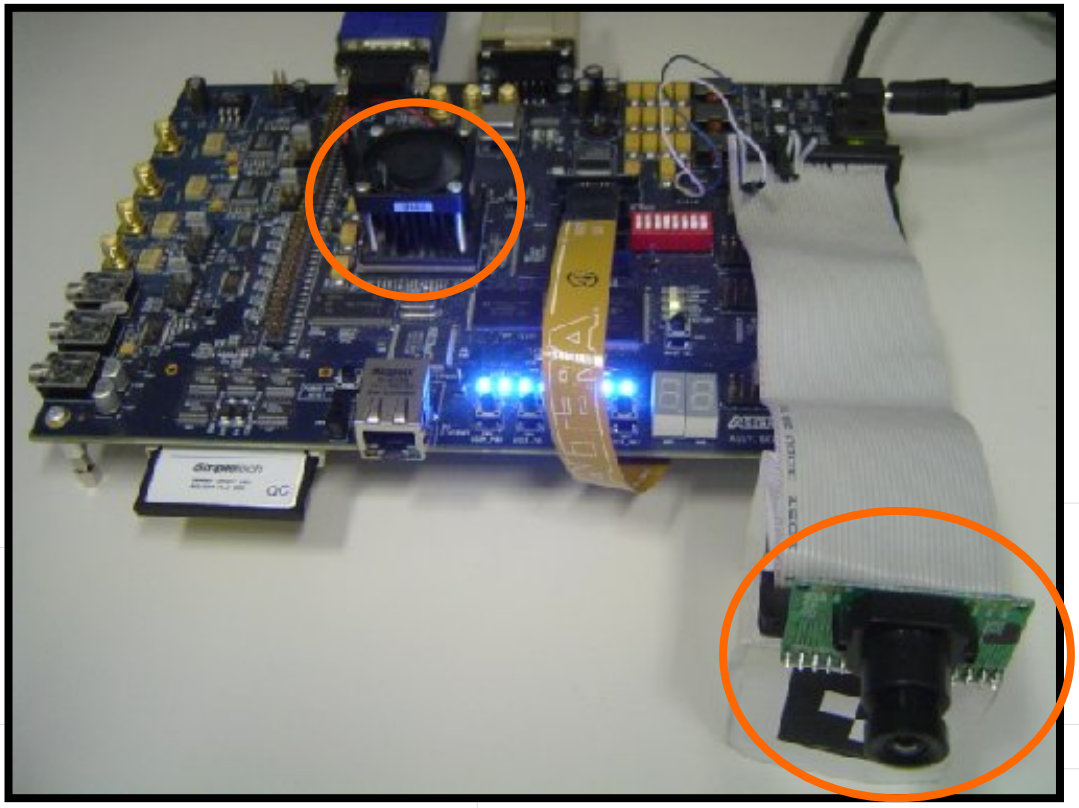
- ARCam: framework for the development of embedded AR systems
- Library of common AR functions
- Development model based on components

Related work

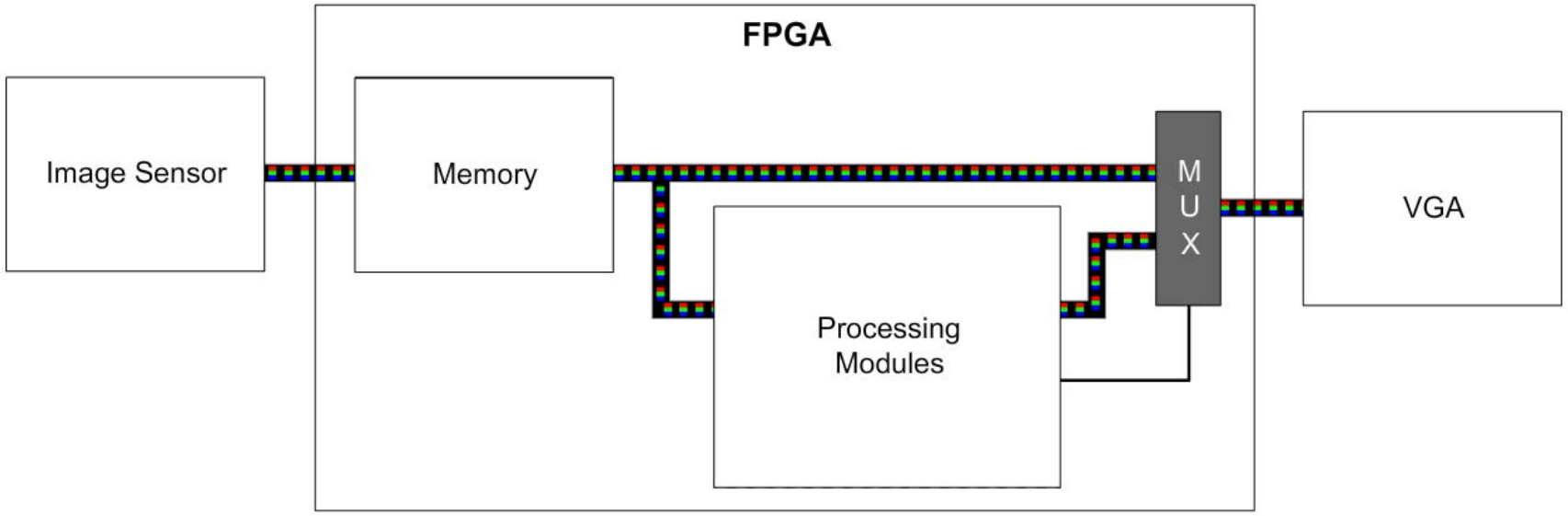
- It was not found any flexible embedded solution for AR applications
- Features of existing solutions
  - Rely on hybrid hardware-software approaches
  - Dedicated to specific applications
- ARCam contribution
  - Entirely hardware based
  - General component based framework for developing embedded AR applications

ARCam development environment

- Digital image sensor
- Altera Stratix II FPGA

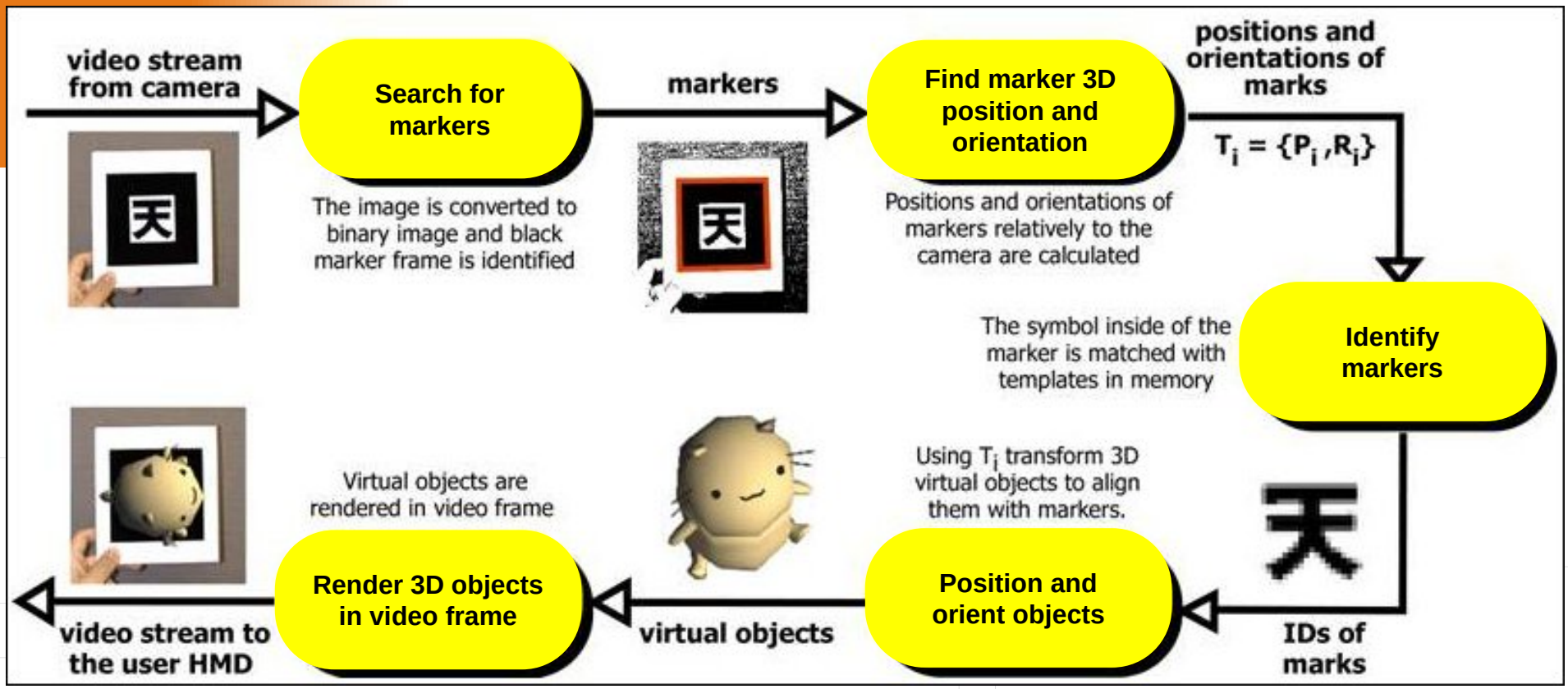


Framework architecture



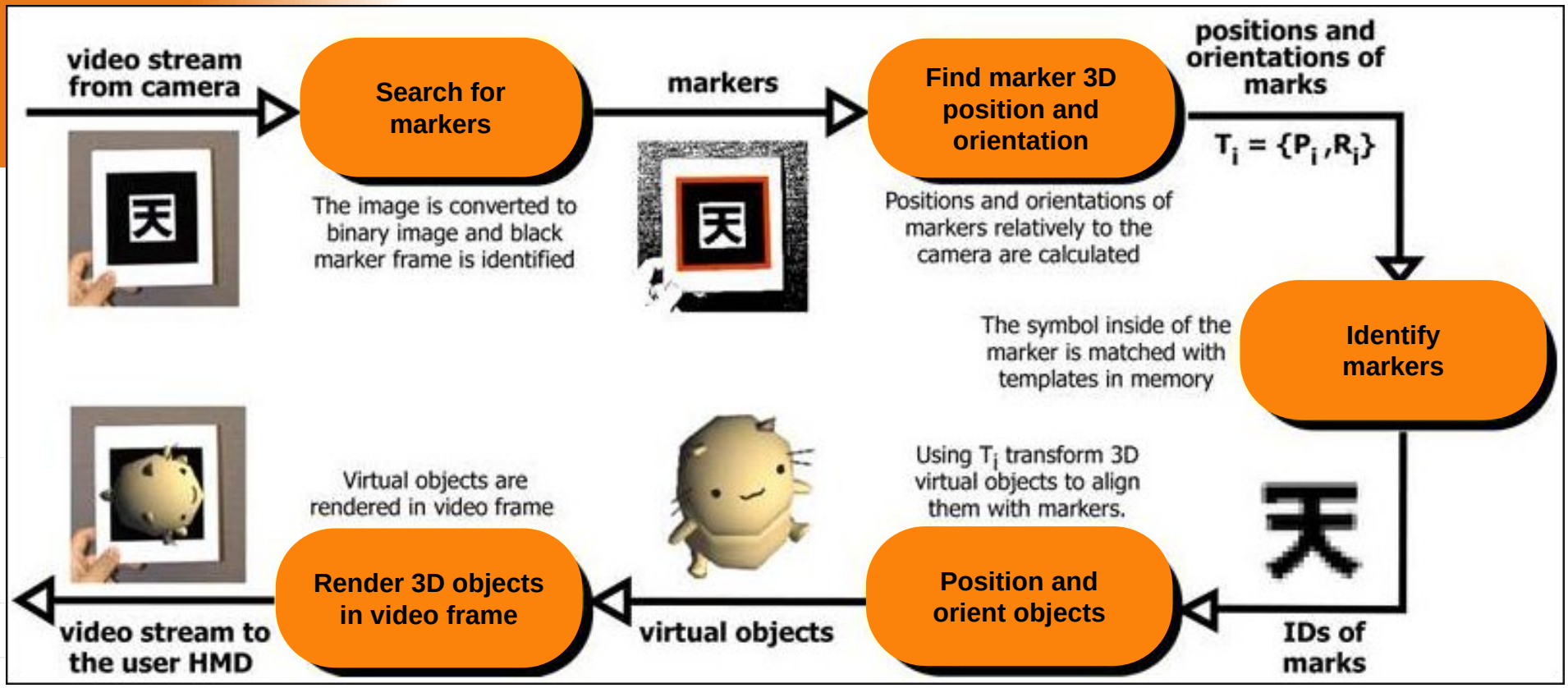
ARCam architecture

Framework architecture



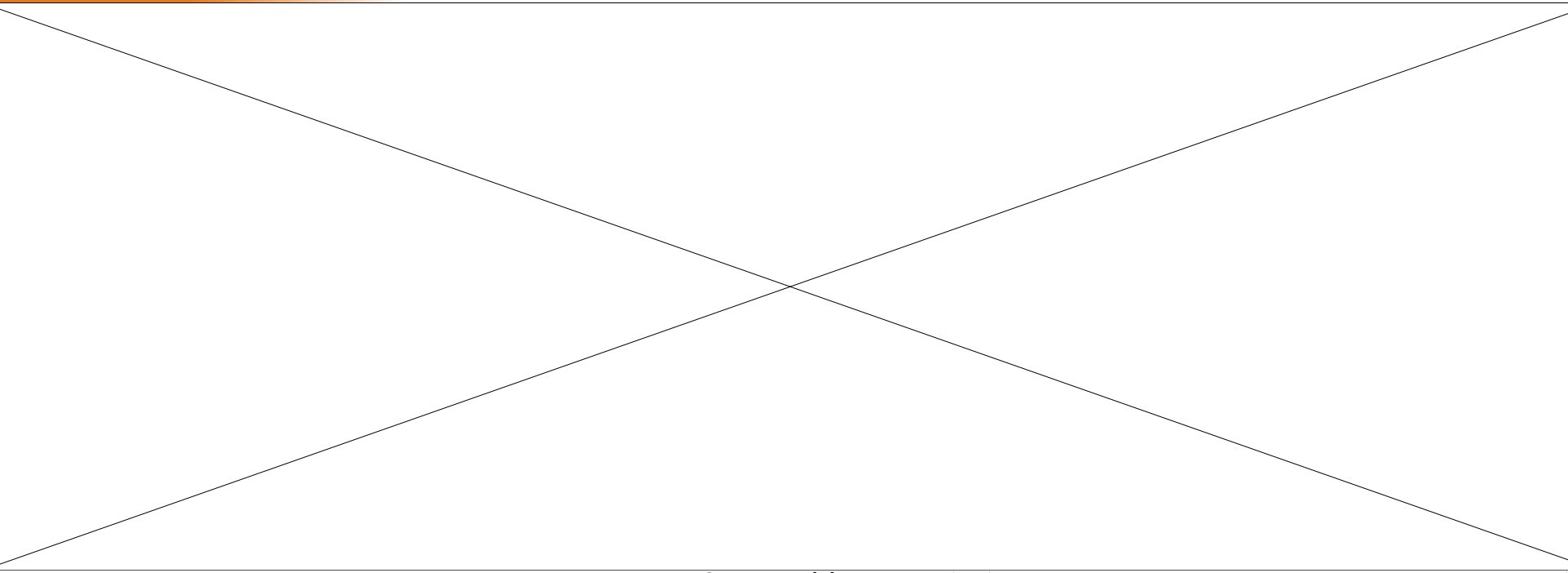
ARToolKit Pipeline

Framework architecture



ARCam architecture

Framework architecture



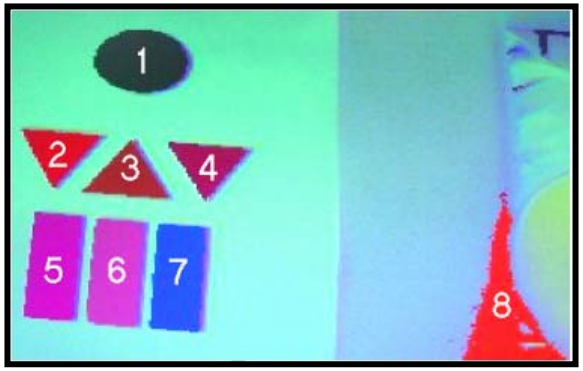
ARCam architecture

Implemented components

- Image binarization and gray scaling
- Labeling
- Generic convolution
- Mean filter
- Edge detection
- Centroid estimation
- Quad detection
- Hardwire

Implemented components

- Image binarization and gray scaling
- Labeling
- Generic convolution
- Mean filter
- Edge detection
- Centroid estimation
- Quad detection
- Hardwire



Implemented components

- Image binarization and gray scaling
- Labeling
- Generic convolution
- Mean filter
- Edge detection
- Centroid estimation
- Quad detection
- Hardwire

N_{11}	N_{12}	N_{13}
N_{21}	P_{in}	N_{23}
N_{31}	N_{32}	N_{33}

C_{11}	C_{12}	C_{13}
C_{21}	C_{22}	C_{23}
C_{31}	C_{32}	C_{33}

$$P_{out} = C_{11} \times N_{11} + C_{12} \times N_{12} + C_{13} \times N_{13} + C_{21} \times N_{21} + C_{22} \times P_{in} + C_{23} \times N_{23} + C_{31} \times N_{31} + C_{32} \times N_{32} + C_{33} \times N_{33}$$

Implemented components

- Image binarization and gray scaling
- Labeling
- Generic convolution
- Mean filter
- Edge detection
- Centroid estimation
- Quad detection
- Hardwire

Implemented components

- Image binarization and gray scaling
- Labeling
- Generic convolution
- Mean filter
- Edge detection
- Centroid estimation
- Quad detection
- Hardwire

Implemented components

- Image binarization and gray scaling
- Labeling
- Generic convolution
- Mean filter
- Edge detection
- Centroid estimation
- Quad detection
- Hardwire

Implemented components

- Image binarization and gray scaling
- Labeling
- Generic convolution
- Mean filter
- Edge detection
- Centroid estimation
- Quad detection
- Hardwire

Implemented components

- Image binarization and gray scaling
- Labeling
- Generic convolution
- Mean filter
- Edge detection
- Centroid estimation
- Quad detection
- Hardwire

Performance analysis

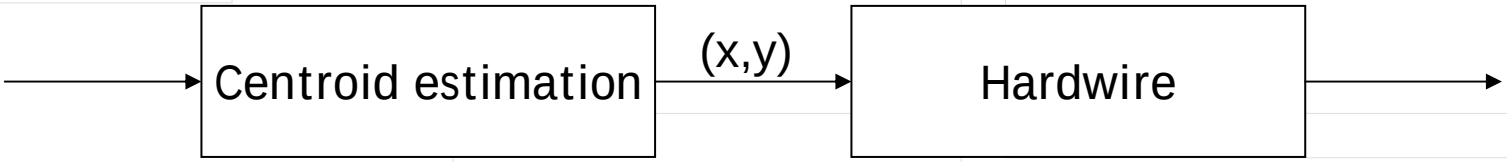
	Ratio (100MHz)
Process	**sw/hw**
Binarization	18.89
Gray Scale	16.10
3x3 Filter	30,428.57
Mean Filter	1,751.15
Edge Detection	1,951.06
Labeling	3,623.64
Centroid	5.26
QuadDetection	27.63

Results

- Two case studies
  - Pong
    - Prototype AR application rapidly created
    - Do not worry about modularization
  - Object recognition
    - Make use of the componentized design model

Results :: Pong

Results :: Object recognition



Lessons learned

- Software to hardware translation
- Recursion to iteration
- New possible optimizations

Conclusions

- An architecture was implemented to support the development of AR embedded solutions
- A pre-existent infrastructure makes the development of hardware based AR applications easier and faster
- Performance obtained from the hardware implementation was shown to be satisfactory

Future work

- Performance analysis
- Finish QuadDetector
- Hardwire extension
  - Z-buffer
  - Textures
- Creation of an authoring tool for hardware based AR applications
- More complex AR studies
- Different AR approaches
  - Markerless AR
- External memory access

miva: Constructing a Wearable Platform Prototype

Virtual Reality and Multimedia Research Group

GRVM

{jmxnt, ds2, gsm, lhcbc, vt, jk} @cin.ufpe.br

João Marcelo Teixeira
DALITON SILVA
Guilherme Moura

Luiz Henrique Costa
Veronica Teichrieb
Judith Kelner

Centro de Informática U·F·P·E

UFPE

Petrópolis, May 2007

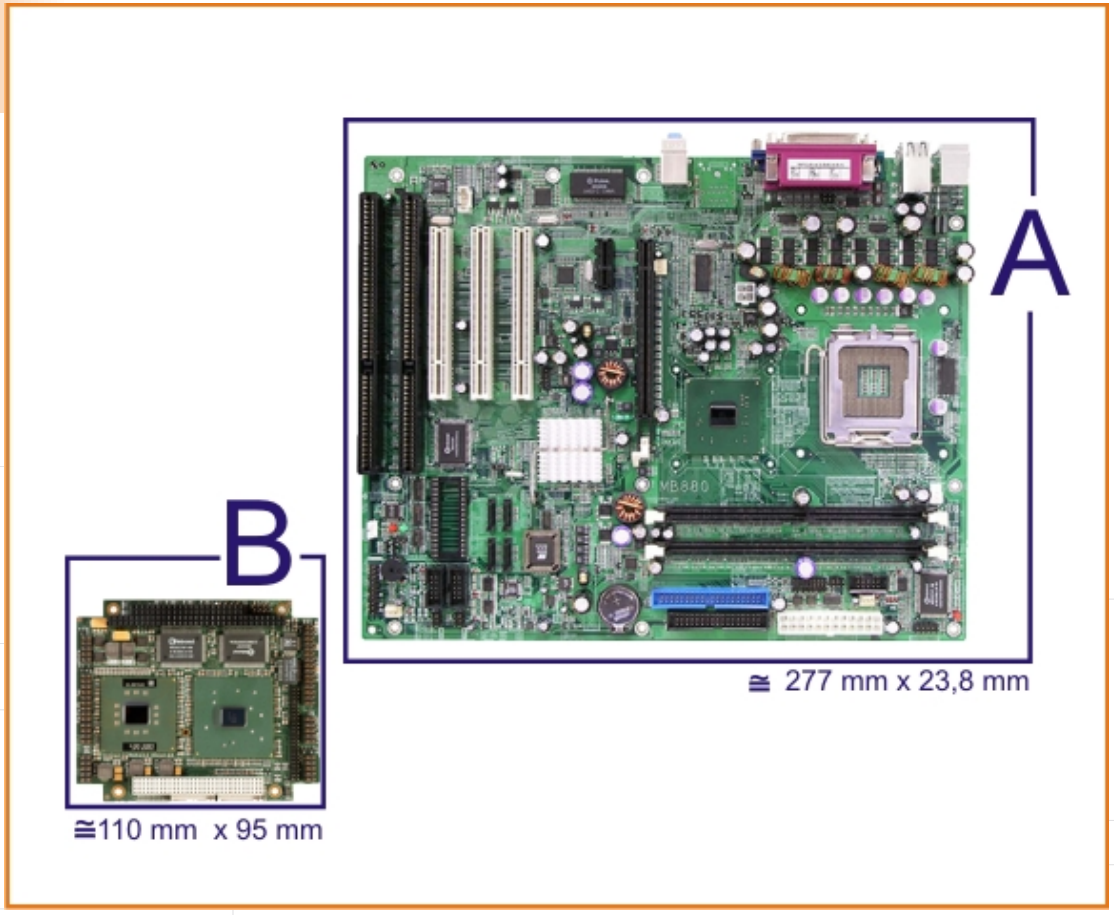
Introduction

- Virtual and augmented reality applications hosting
- Mobile and autonomous execution

Introduction

- High performance hardware requirement
  - 3D applications
  - Visual quality/response time result in high cost!
- Minimum intervention on user's mobility
  - Wearable computer

miva Prototype



miva Prototype

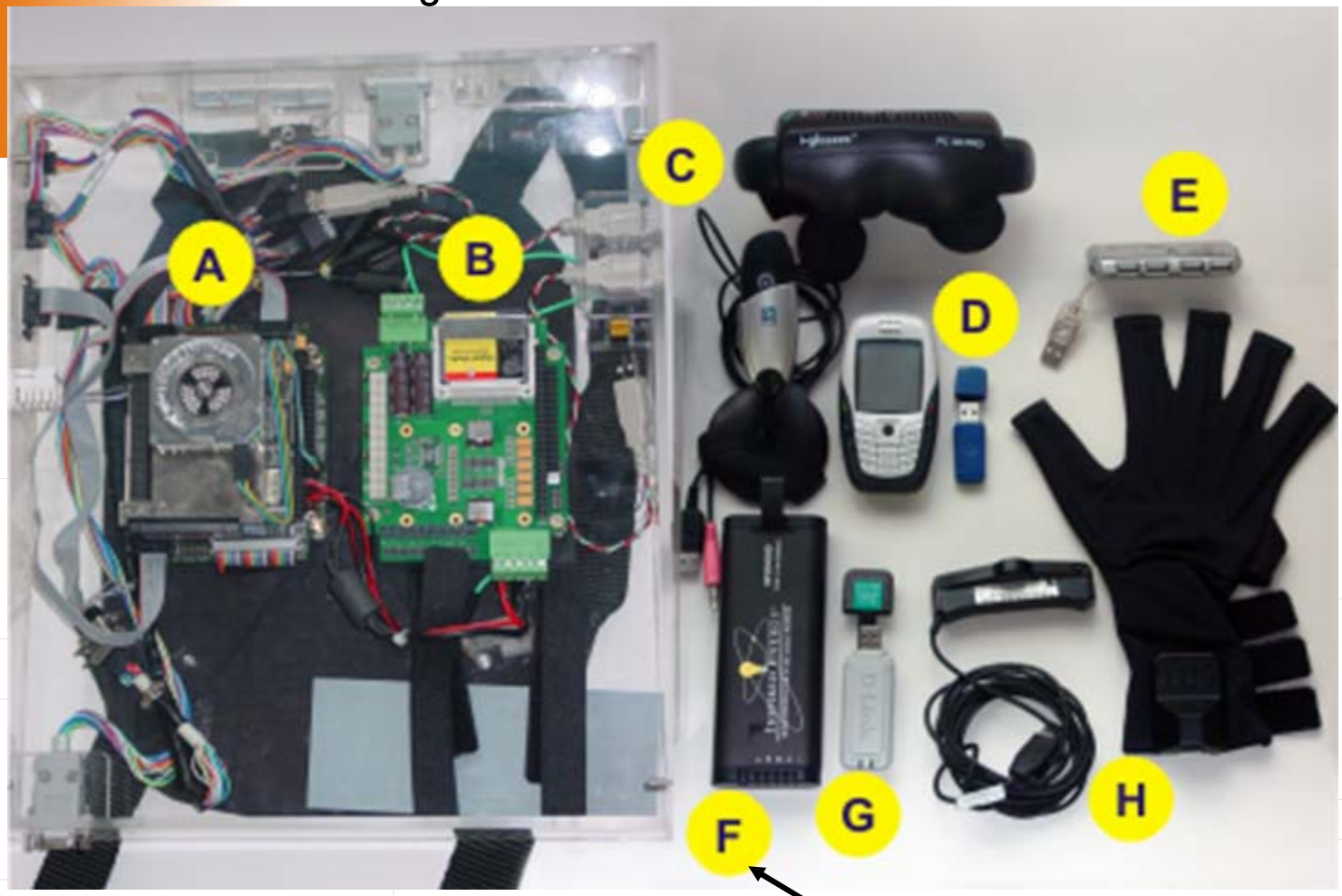
- Intel Pentium-M,1.4GHz
- 512MB DDR (up to 1024MB)
- 855GME Intel, 16-64MB
- Intel 10/100 Mbit/s

miva Prototype

- 4 USB V2.0 ports
- Stereo analog output and SPDif audio input, mic and SPDif
- COM1/COM2 RS232
- 5Volts DC 1.5Ah

miva Prototype

3.4kilograms



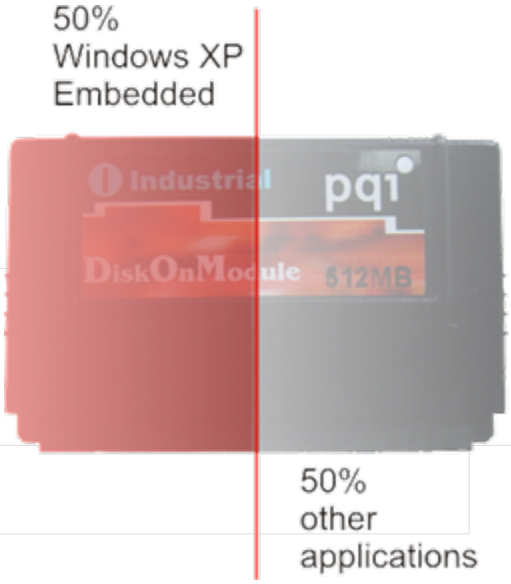
36cm

31cm

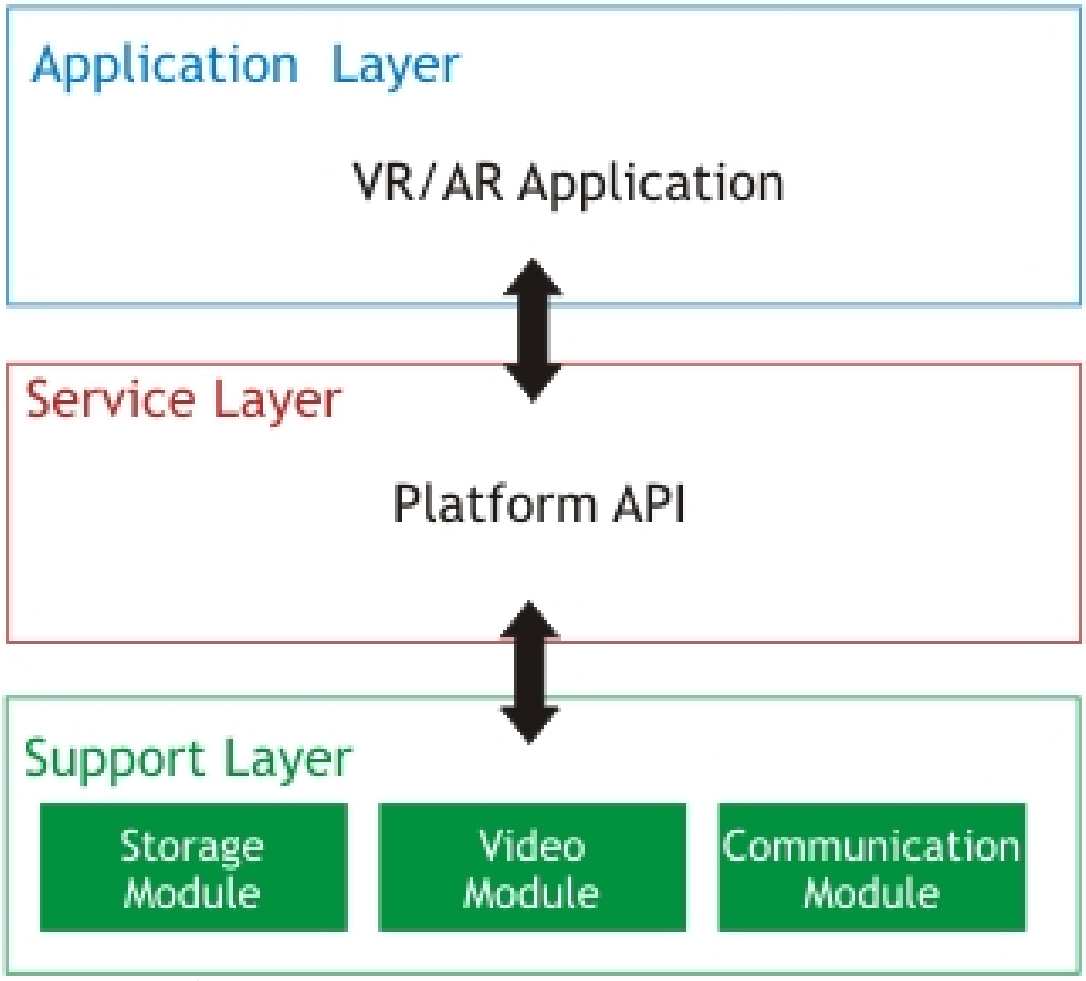
4hours

miva Prototype

- Software setup
  - Windows XP Embedded
    - Reduced storage space required
    - High performance

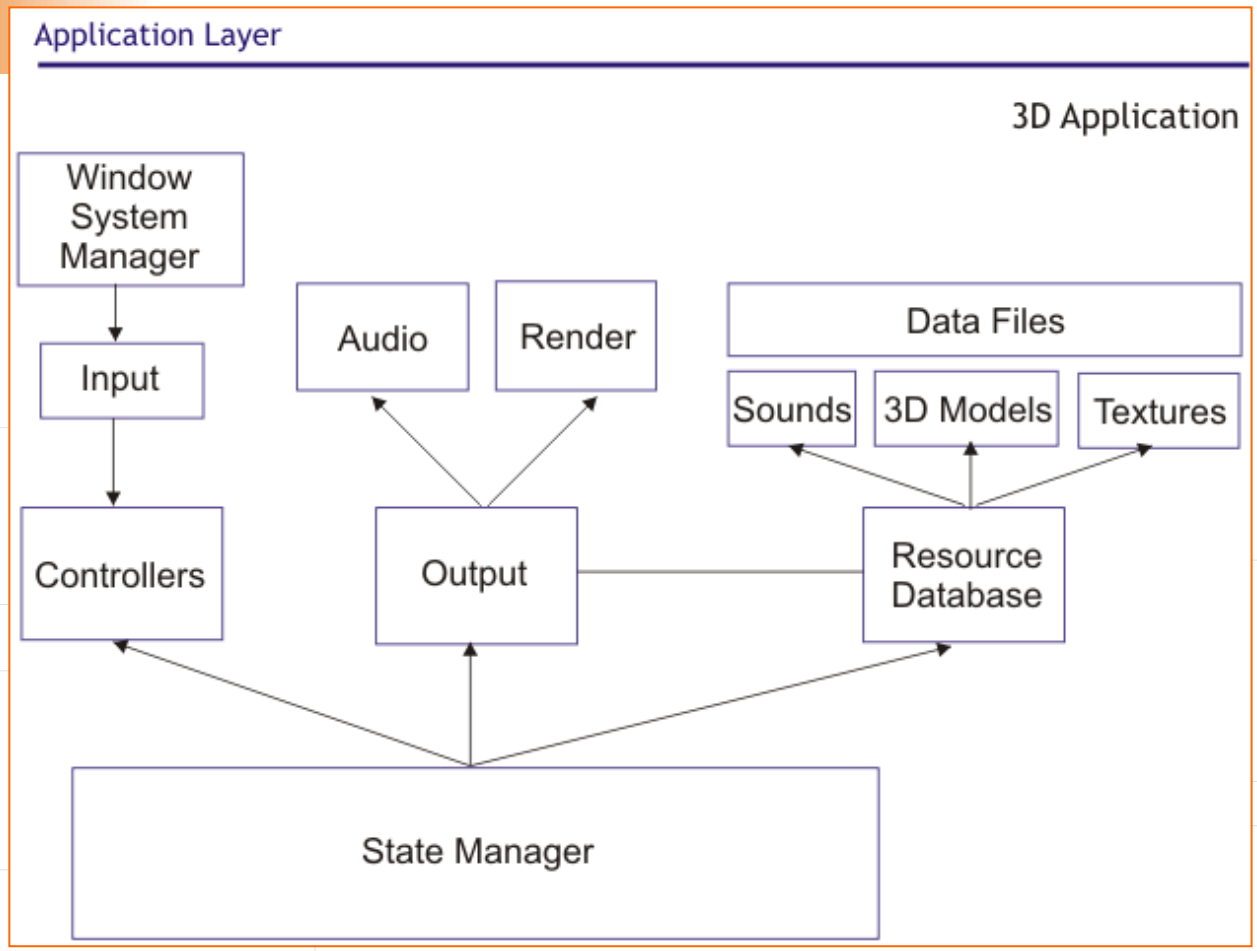


Software Architecture



Software Architecture

- Application layer modules

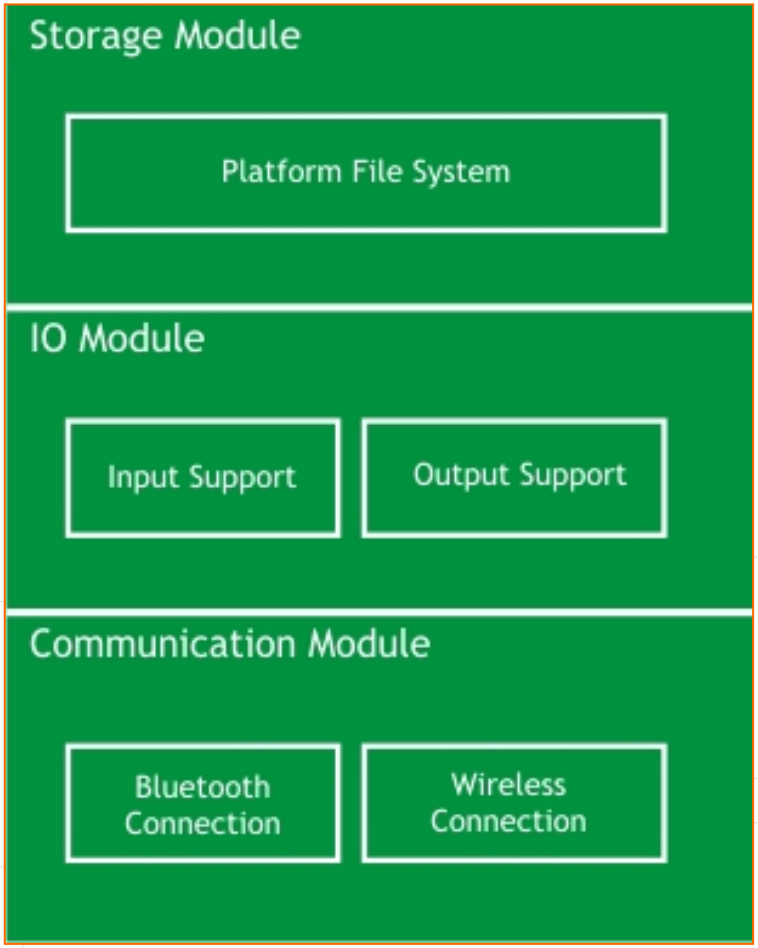


Software Architecture

- Service layer
  - Hardware abstraction API
  - Can be customized and incremented by user
  - Provides a high level abstraction for communication and persistence services

Software Architecture

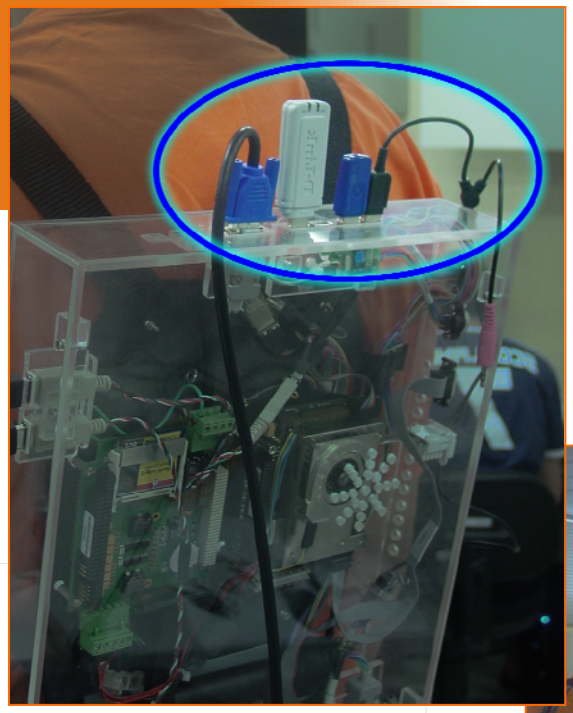
- Support layer modules



Potential Applications

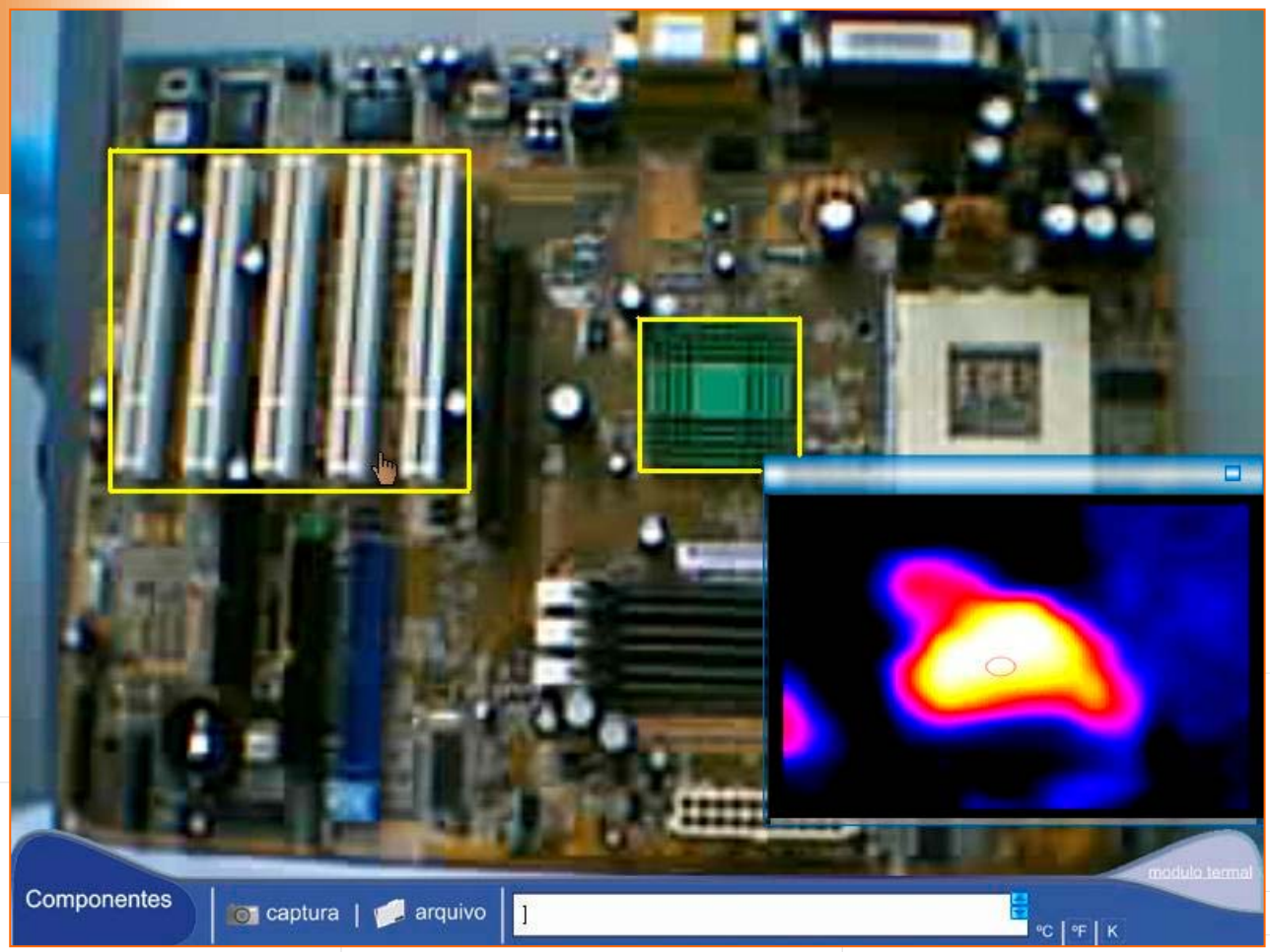
mivaDesk

Potential Applications



mivaDesk

Potential Applications



mivaTherm

Conclusions and Future Work

- RV/RA platform
- Easily extensible
- Developed without previous project design

Conclusions and Future Work

- miva platform physical evaluation
  - Size must be reduced
  - A more usable container will be developed
  - Use of more accurate pointer devices (data glove and tracker)