Pivoting Object Tracking System

CSEE 4840 Embedded System Design, Spring 2009

Damian Ancukiewicz
Arjun Roy
Baolin Shao
Jinglin Shen

Outline

- Project overview
- Key points in different components
- Experience and lessons learned

System overview

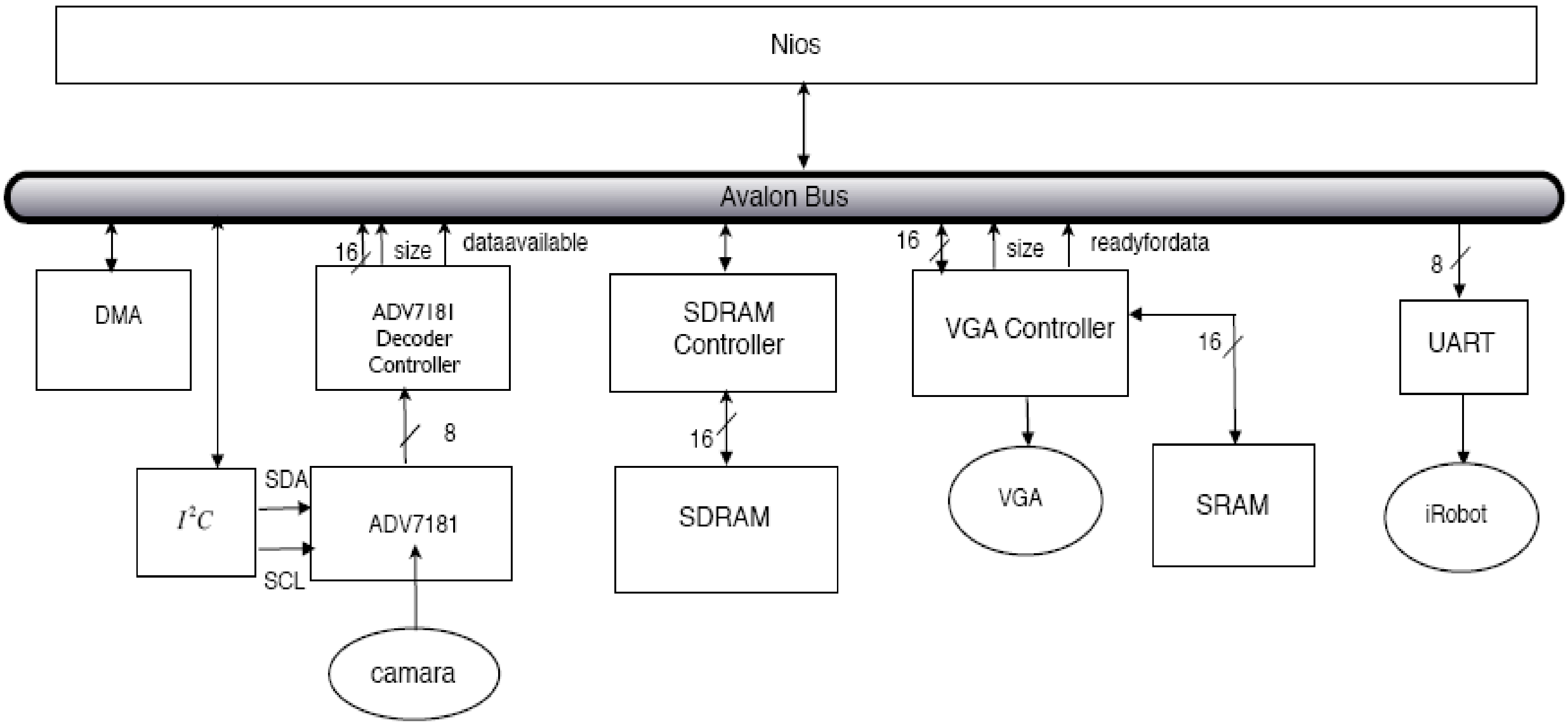
Hardware components:

- Video decoder controller: ADV 7181 interface, YUV->RGB, Buffer to Avalon bus
- VGA frame buffer
- I2C controller

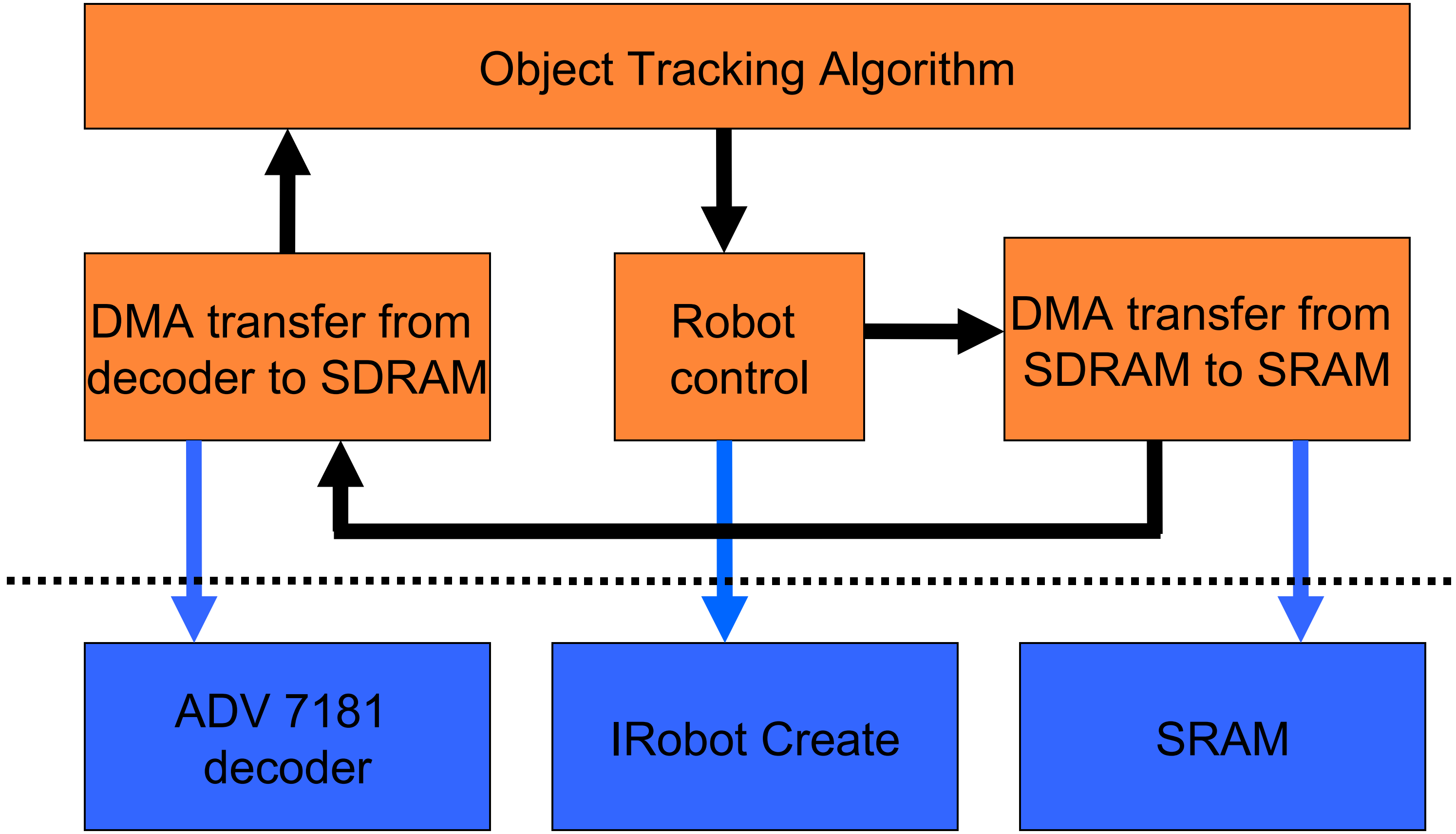
Software components:

- DMA transfer
- Object tracking
- Robot control

Project Overview: hardware structure



Project overview: software structure



Video Controller

- System needs video for recognition
- Solution: analog video camera
- DE2 has an onboard ADC (ADV7181)

Video Controller

- **ADV7181 interface**
  - Output format: 1716 clocks, 8 bits wide, YUYV
  - Two interlaced fields of 262 lines
  - Horizontal/vertical sync
  - I2C configuration

Video Controller

- **YUV->RGB**
  - Y: luma (brightness); U/V: chroma (color)
  - Conversion done on a two-pixel YUYV block
  - Output: single 16-bit RGB pixel

Video Controller

- **Transfer through Avalon bus**
  - Frame needs to be sent to SDRAM using DMA
  - Problem: SDRAM has lax timings
  - Solutions attempted: FIFO, line buffer

- FIFO
  - ADV7181 interface puts in pixels, Avalon bus pulls pixels
  - Avalon flow control
  - Problem: different clock speeds, so not synchronous!
  - Data lost, corrupted image

- New idea: line buffer
  - Double buffering: two lines stored in block memory
  - Video ADC output writes to one while other is output through Avalon interface

VGA buffer

- **The need for VGA**
  - Use POTS as automated remote surveillance
  - Needs video output for human observers
  - Choice of stream or framebuffer
  - Choice of SRAM, SDRAM, Flash framebuffer

VGA buffer

- **Implementation**
  - Went with SRAM
  - Advantages : Fast, Simple
  - Disadvantages : Single Ported
  - Implications : Synchronize when to read/write

VGA buffer

- **Results**
  - Results : Slower than anticipated, some frame tearing
  - Ended up not being a big deal
  - Future directions: double buffering, change of backing memory type, modesetting?

DMA

- **The need for DMA**
  - Handle large transfers without using NIOS II
  - Also allows us to implement flow control with Avalon peripherals
  - Data transfer at a rate determined by limiting factor (the peripheral)
  - Just a drop in device

DMA

- **Results**
  - Slower than anticipated for SDRAM to SRAM
  - Weird race condition when processor writing to SDRAM and initiating a DMA transfer from/to SDRAM

Robot

- **iRobot Create**
  - Differential drive robot, moves in 2D plane
  - Only need a subset of capabilities (rotate)
  - Serves as mount for camera

Robot

- **Implementation**
  - Communicates using RS-232, which is just drop in peripheral for NIOS
  - Has an interface of opcodes which are simple bytes and allow basic scripting
  - "Turn left at a given speed till a given angle is sweeped out"
  - Only complication - had to expertly fabricate a null modem adapter

OBJECT TRACKING

Step 1: Recognize the target object in an image

We recognize an object by its color
(Assumption: an object has only one color)

Step 2: Calculate how the target is moving

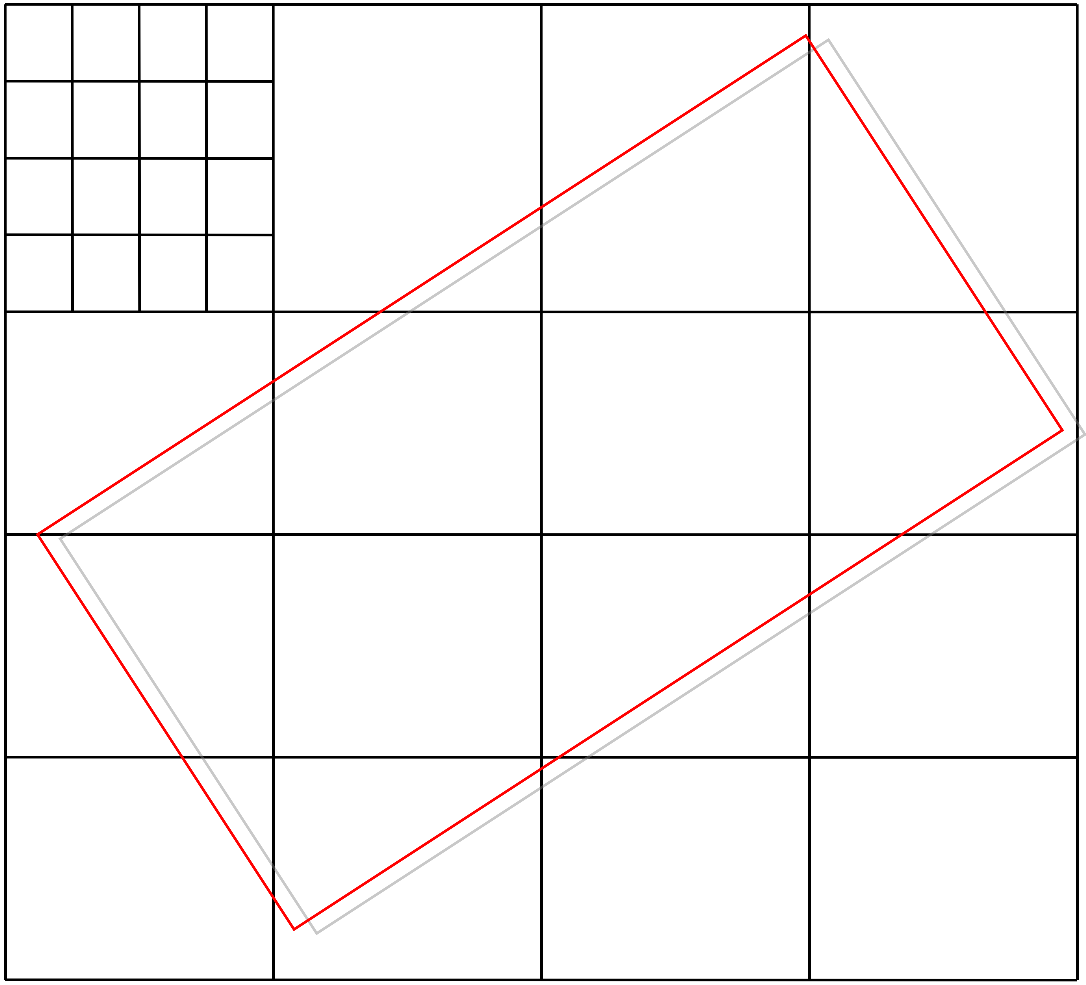
We calculate an object's motion by its center's position.
(Assumption: an object has regular shape)

Step 3: Reposition the device such that the target is always in sight

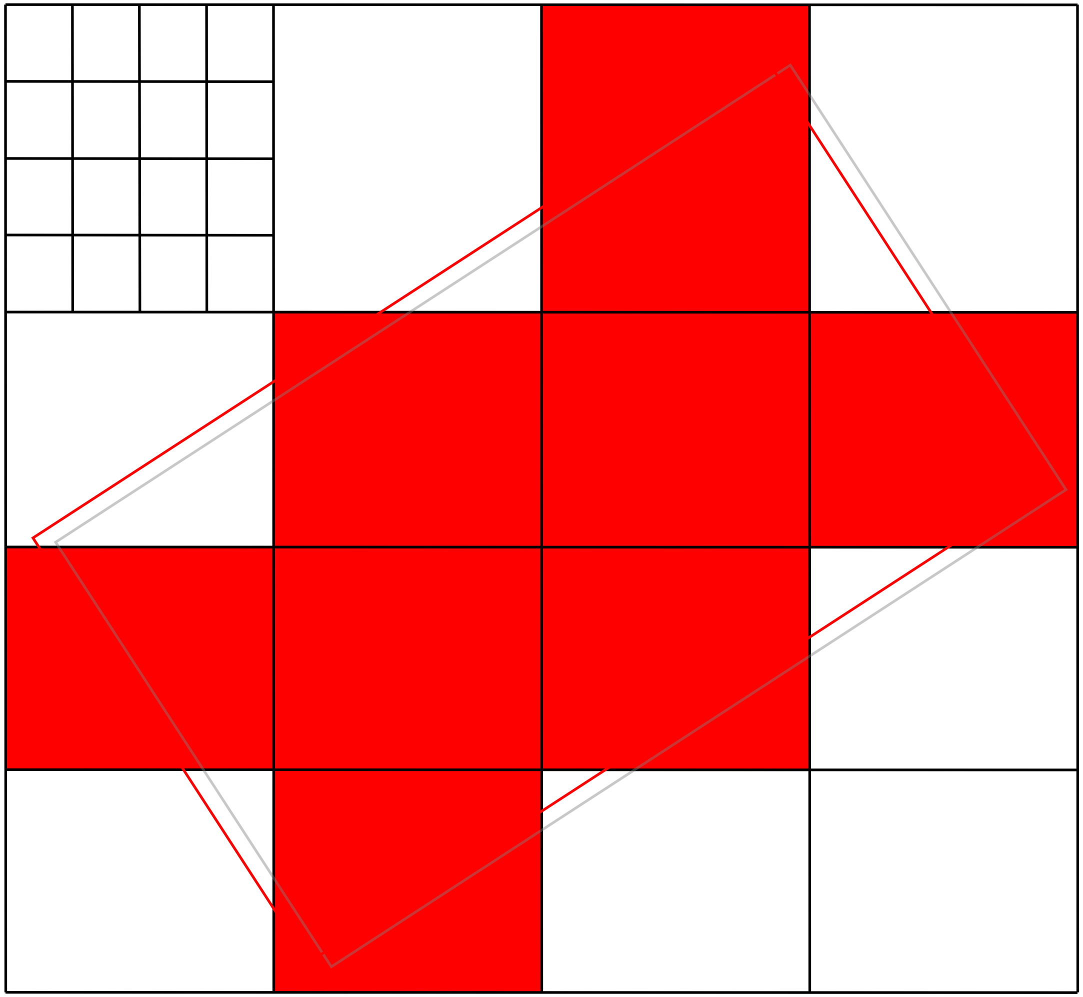
OUR ALGORITHM
- A UTOPIAN SCENARIO

pixel

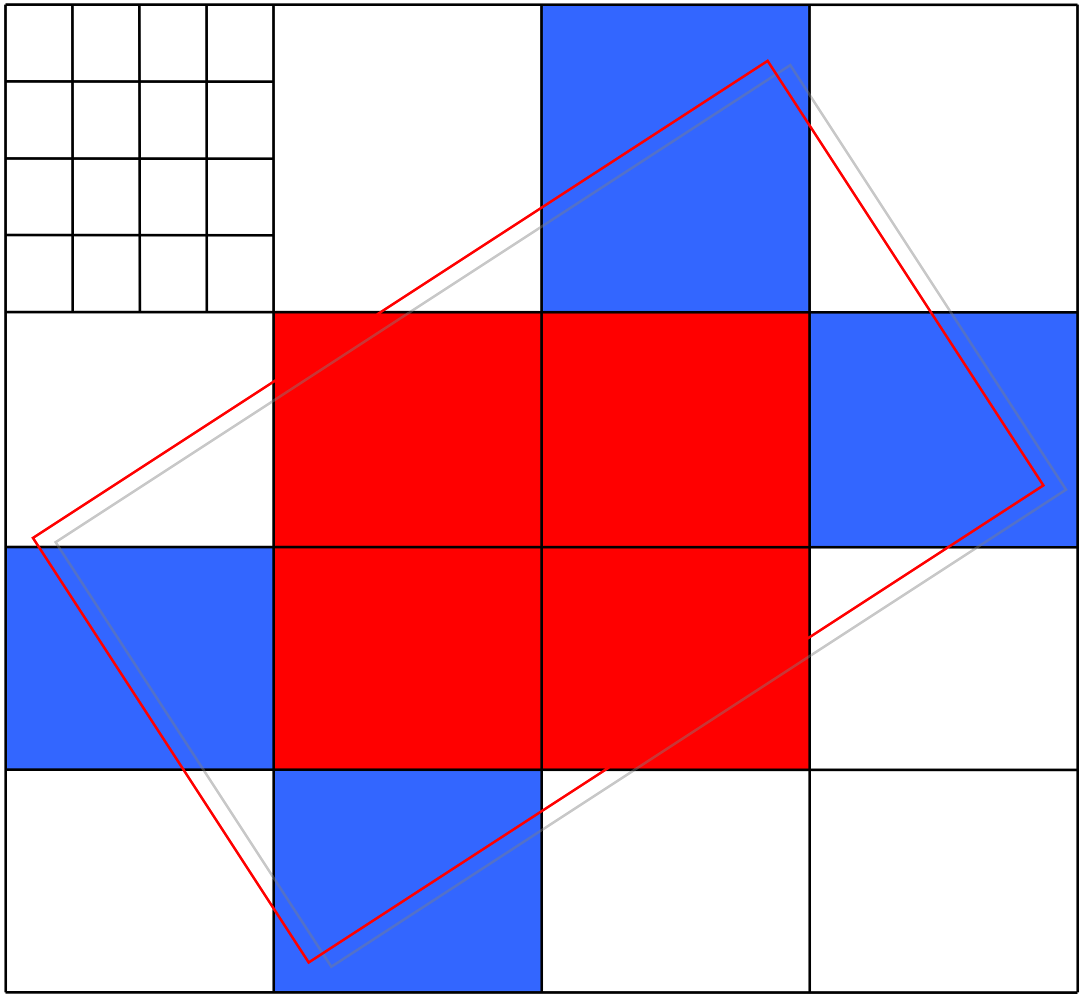
N x N block



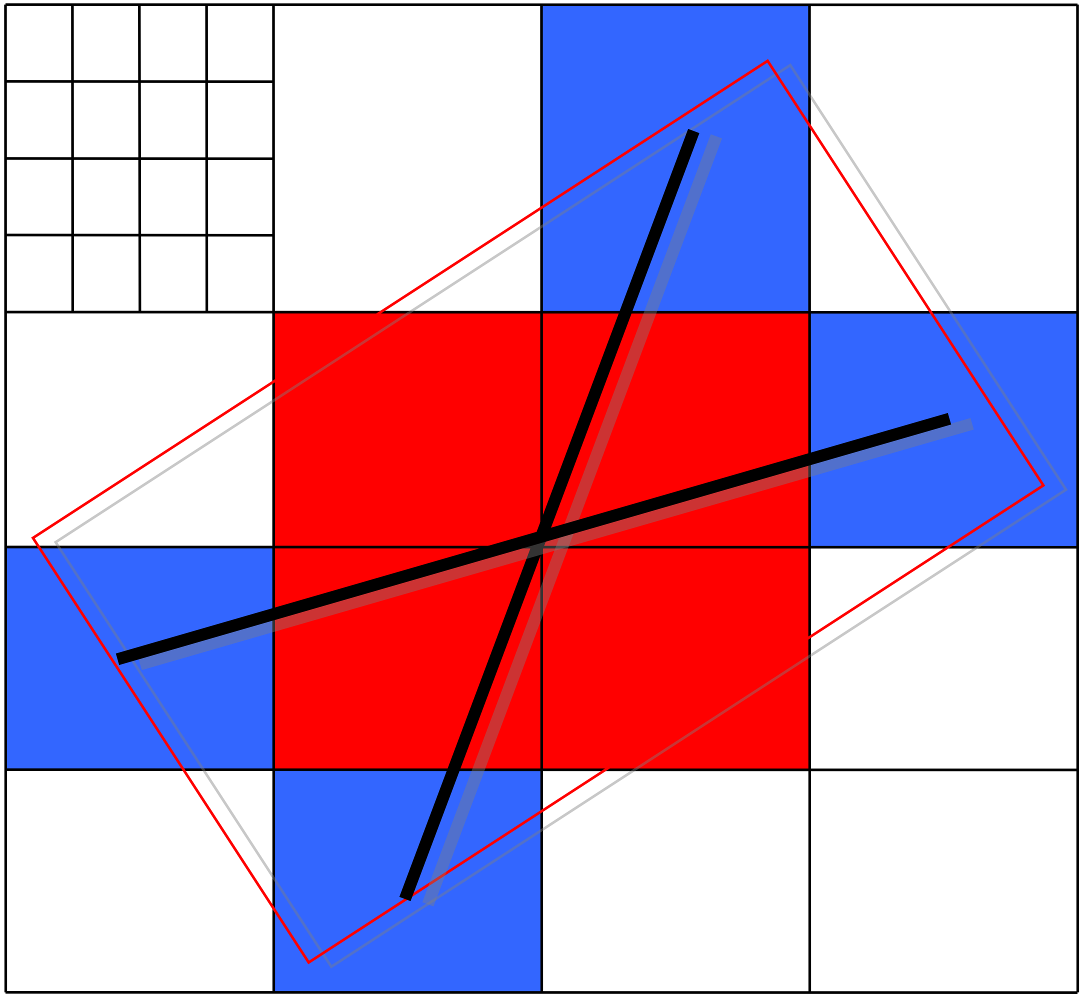
OUR ALGORITHM
- A UTOPIAN SCENARIO



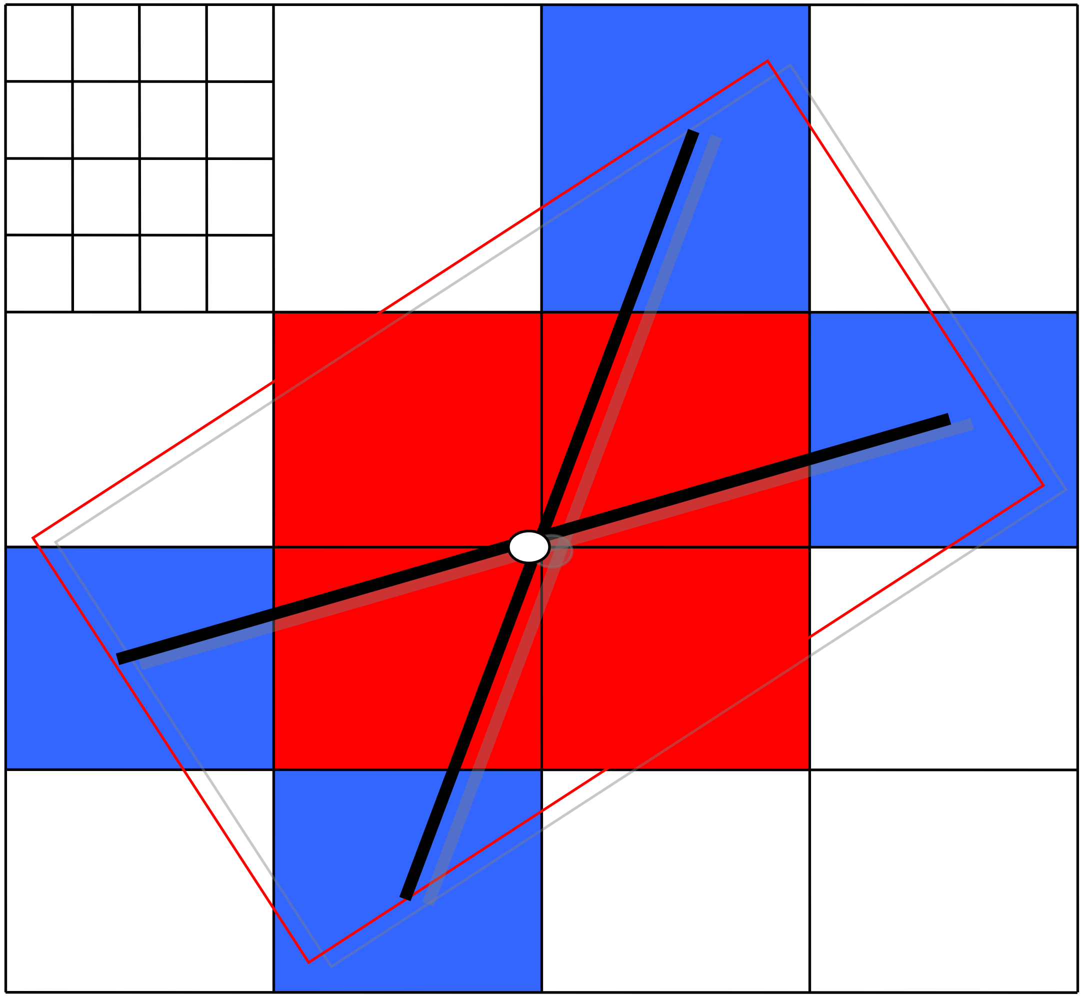
OUR ALGORITHM
- A UTOPIAN SCENARIO



OUR ALGORITHM
- A UTOPIAN SCENARIO

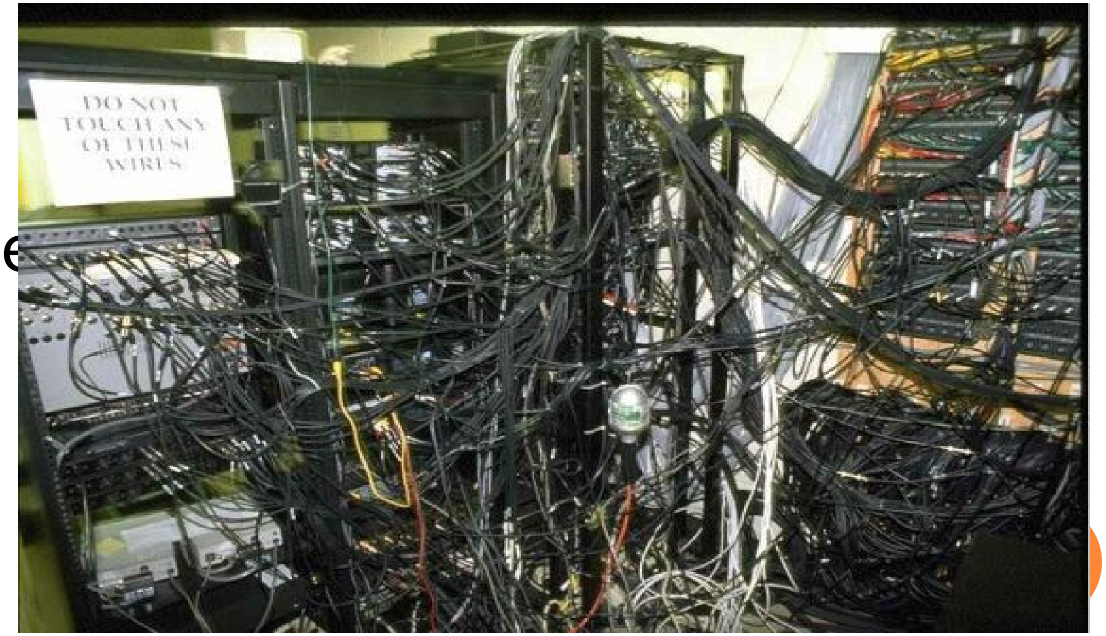


OUR ALGORITHM
- A UTOPIAN SCENARIO



WHAT MAKES LIFE EVEN HARDER

- DE2 has limited computational power
- Computer is not good at interpreting colors
- Image has noise
- Video is not stable



TRICKS TO ALLEVIATE COMPUTATION

Divide an image into blocks

- Reduce number of comparisons

Replace multiplications by shifts and additions

- x*320 = (x<<8)+(x<<6)

Replace divisions by shifts, additions and subtractions

- $$\frac{x}{5} = \frac{x}{(4+1)} = \frac{(\frac{x}{4})}{1+\frac{1}{4}} = \frac{x}{4} \times (1 - \frac{1}{4} + \frac{1}{16} - \frac{1}{64} + \frac{1}{256} \ldots)$$

Experiences

- Design of a embedded system with focusing on the most critical issue: all about timing!
- Usage of the ADV 7181 decoder, DMA
- Data transfer between different frequency domains
- Simple image recognition algorithm
- Interfacing between different hardware components

Lessons learned

- Importance of a good design
- Always keep things synchronous
- Try to keep things simple unless you have a good reason to make it complicated
- Don’t trust the hardware
- Be careful with estimating how much time something will take (both in real life and in hardware!)