Augmented Reality on FPGA

Realtime Object Recognition and Image Processing

Logan P. Williams José E. Cruz Serrallés

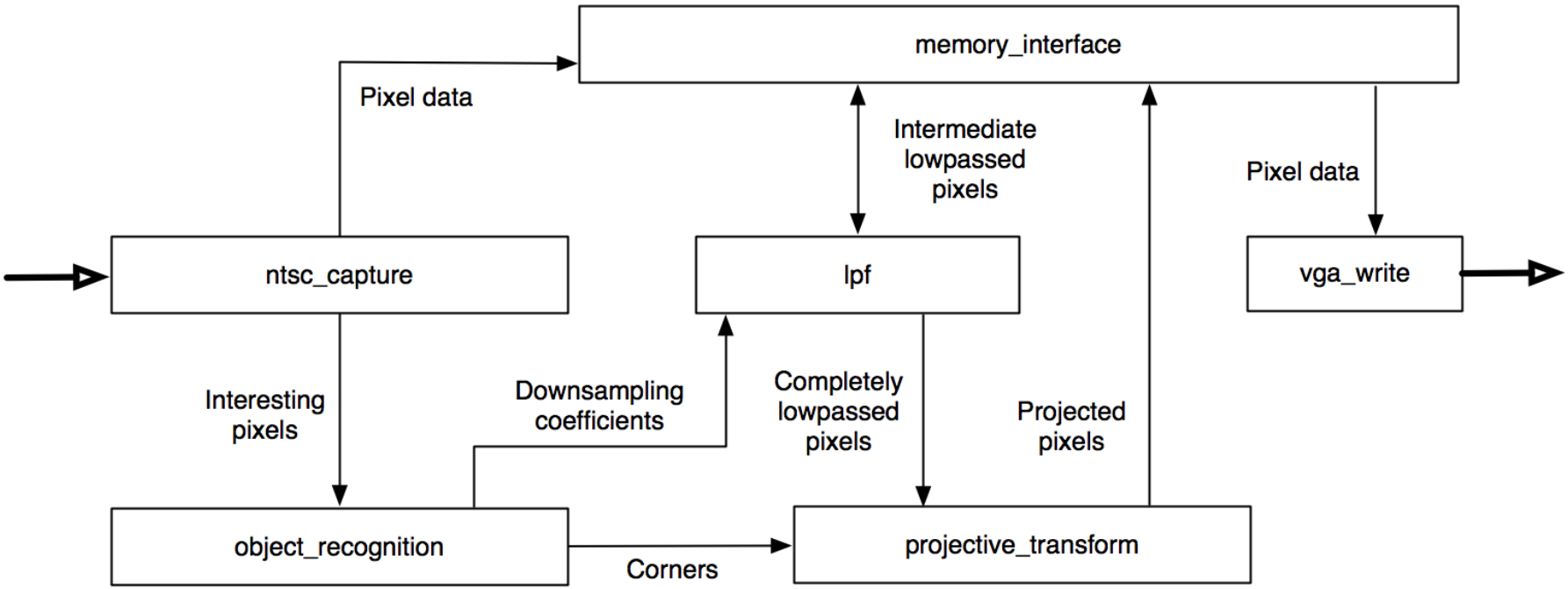
15 November 2011

Introduction

Overlay a digital image on a physical object in realtime.

example image

top-level overview



projective_transform

object_recognition

`LPF`: its purpose

- `projective_transform` → aliasing

graphic showing normal signal

`LPF`: its purpose

- `projective_transform` → aliasing

graphic aliases

`LPF`: its purpose

- `projective_transform` → aliasing
- Aliasing reduces the quality of an image

zoom in on aliased pixels

`LPF`: its purpose

- `projective_transform` → aliasing
- Aliasing reduces the quality of an image
- Lowpass filtering prevents aliasing

picture depicting lowpass filter in 2D

`LPF`: its purpose

- `projective_transform` → aliasing
- Aliasing reduces the quality of an image
- Lowpass filtering prevents aliasing
- Information of an image is mostly phase

picture of original picture

`LPF`: its purpose

- `projective_transform` → aliasing
- Aliasing reduces the quality of an image
- Lowpass filtering prevents aliasing
- Information of an image is mostly phase

picture of other image

`LPF`: its purpose

- `projective_transform` → aliasing
- Aliasing reduces the quality of an image
- Lowpass filtering prevents aliasing
- Information of an image is mostly phase

picture of original phase with other's magnitude

`LPF`: its purpose

- `projective_transform` → aliasing
- Aliasing reduces the quality of an image
- Lowpass filtering prevents aliasing
- Information of an image is mostly phase
- Symmetric Type I FIR filter → 0 phase distortion

picture of original phase with other's magnitude

`LPF`: its purpose

- `projective_transform` → aliasing
- Aliasing reduces the quality of an image
- Lowpass filtering prevents aliasing
- Information of an image is mostly phase
- Symmetric Type I FIR filter → 0 phase distortion
- Parks-McClellan: reasonable accuracy, symmetric, easily calculable

frequency response of Parks-McClellan filter

`LPF`: its purpose

- `projective_transform` → aliasing
- Aliasing reduces the quality of an image
- Lowpass filtering prevents aliasing
- Information of an image is mostly phase
- Symmetric Type I FIR filter → 0 phase distortion
- Parks-McClellan: reasonable accuracy, symmetric, easily calculable
- FIR PM filter reduces mem. acceses to 1.5/pixel

LPF: the algorithm

1. Given an arbitrary image & skewing coefficients M_x & M_y.

graphic showing the interface between object_recognition and LPF
image
magnitude fourier plot of image

LPF: the algorithm

1. Given an arbitrary image & skewing coefficients M_x & M_y.
2. Fetch a filter with cutoff $\frac{\pi}{M_y}$.

magnitude plot of image
magnitude plot of filter with cutoff pi/2

LPF: the algorithm

1. Given an arbitrary image & skewing coefficients M_x & M_y.
2. Fetch a filter with cutoff $\frac{\pi}{M_y}$.
3. Filter each column and store in memory.

magnitude plot of image
magnitude plot of filter with cutoff pi/2
magnitude fourier plot of filtered

LPF: the algorithm

1. Given an arbitrary image & skewing coefficients M_x & M_y.
2. Fetch a filter with cutoff $\frac{\pi}{M_y}$.
3. Filter each column and store in memory.
4. Fetch a filter with cutoff $\frac{\pi}{M_x}$.

magnitude plot of filtered image
magnitude plot of filter with curoff pi/4

`LPF`: the algorithm

1. Given an arbitrary image & skewing coefficients M_x & M_y.
2. Fetch a filter with cutoff $\frac{\pi}{M_y}$.
3. Filter each column and store in memory.
4. Fetch a filter with cutoff $\frac{\pi}{M_x}$.
5. Filter each row and output to `projective_transform`.

magnitude plot of filtered image
mangitude plot of filter with cutoff pi/5
magnitude plot of output

`LPF`: the algorithm

1. Given an arbitrary image & skewing coefficients M_x & M_y.
2. Fetch a filter with cutoff $\frac{\pi}{M_y}$.
3. Filter each column and store in memory.
4. Fetch a filter with cutoff $\frac{\pi}{M_x}$.
5. Filter each row and output to `projective_transform`.
6. Repeat this process every refresh cycle.

magnitude plot of original
magnitude plot of filter
magnitude plot of output

memory_interface

- 1 image is a lot of data:
  $640 \cdot 480 \cdot 24$ bits ≈ 0.88MiB

memory_interface

- 1 image is a lot of data:
  $640 \cdot 480 \cdot 24$ bits $\approx$ 0.88MiB
- Total BRAM on board: 0.316MiB

memory_interface

- 1 image is a lot of data:
  $640 \cdot 480 \cdot 24$ bits $\approx$ 0.88MiB
- Total BRAM on board: 0.316MiB
- We need to store 4 images in memory!

memory_interface

- 1 image is a lot of data:
  $640 \cdot 480 \cdot 24$ bits $\approx$ 0.88MiB
- Total BRAM on board: 0.316MiB
- We need to store 4 images in memory!
- Let's use the ZBT RAM: $2 \cdot$ 2.25MiB

memory_interface

- 1 image is a lot of data:
  $640 \cdot 480 \cdot 24$ bits $\approx$ 0.88MiB
- Total BRAM on board: 0.316MiB
- We need to store 4 images in memory!
- Let's use the ZBT RAM: $2 \cdot$ 2.25MiB
- Not so fast: 1 clock cycle per memory access

memory_interface

- 1 image is a lot of data:
  $640 \cdot 480 \cdot 24$ bits ≈ 0.88MiB
- Total BRAM on board: 0.316MiB
- We need to store 4 images in memory!
- Let's use the ZBT RAM: $2 \cdot 2.25$MiB
- Not so fast: 1 clock cycle per memory access
- 1 pixel per address would require a clock speed $> 100MHz$

memory_interface

- 1 image is a lot of data:
  $640 \cdot 480 \cdot 24$ bits ≈ 0.88MiB
- Total BRAM on board: 0.316MiB
- We need to store 4 images in memory!
- Let's use the ZBT RAM: $2 \cdot 2.25$MiB
- Not so fast: 1 clock cycle per memory access
- 1 pixel per address would require a clock speed $> 100MHz$
- Let's store store 18 bits per pixel or 2 per address

`memory_interface`: operation

1.

system io: `ntsc_capture`

system io: vga_write

timeline