Image Processing on Microcontrollers

"Libraries and Interfaces, Oh Joy!"

Paul Pounds

16 April 2013

University of Queensland
But first…

Some house keeping
## Calendar at a glance

<table>
<thead>
<tr>
<th>Week</th>
<th>Dates</th>
<th>Lecture</th>
<th>Reviews</th>
<th>Demos</th>
<th>Assessment submissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>25/2 – 1/3</td>
<td>Introduction</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>4/3 – 8/3</td>
<td>Principles of Mechatronic Systems design</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>11/3 – 15/3</td>
<td>Principles of Sailing</td>
<td></td>
<td></td>
<td>Design brief</td>
</tr>
<tr>
<td>4</td>
<td>18/3 – 22/3</td>
<td>Sensor Fusion and Filtering</td>
<td>Progress review 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>25/3 -29/3</td>
<td>Your Soldering is Terrible</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Break</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>8/4 – 12/4</td>
<td>3D printing</td>
<td>Progress seminar</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>15/4 – 19/4</td>
<td>By request</td>
<td></td>
<td>25% demo</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>22/4 – 26/4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>29/4 – 3/5</td>
<td>Progress review</td>
<td></td>
<td>50% demo</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>6/5 – 10/5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>13/5 – 17/5</td>
<td></td>
<td></td>
<td>75% demo</td>
<td>Preliminary report</td>
</tr>
<tr>
<td>12</td>
<td>20/5 – 24/5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>27/5 – 31/5</td>
<td>Closing lecture</td>
<td></td>
<td>Final testing</td>
<td>Final report and addendum</td>
</tr>
</tbody>
</table>
FAQ Roundup

• None as yet
Lab matters

• Food wrapper and drinks bottles in c404
  – Lab is No Food Zone. Eat outside plz kthxbye.

• Wednesday 1 pm is Doug’s regular lab time
  – Great opportunity to be inducted
  – You will be inducted.
Incremental demo 1

• First opportunity to strut your stuff!
  – Earned marks will be capped at 25% of total

• Easier scenario:
  – No dragon
  – No volcano
  – No tears (hopefully)

By appointment only – so far no takers
Onwards…

Image processing what?
What are we talking about?

• Image processing – computer vision stuff
  – Frames captured from digital cameras, etc
  – Computations made using pixel values

• Microcontrollers – single chip processors
  – No stand-alone RAM, co-processors, etc
Key questions

• How do we get the image in?
• How are images stored?
• How do we perform operations on them?
• How do microcontrollers do this?
• Can we do this fast enough?
• How do we do this practically?
Getting the image in

- Two general approaches:
  - Parallel input
  - Serial input
- Image is cached in registers before being written to memory
  - Slow, and ties up processor while shuffling data
- Direct Memory Addressing takes data directly from inputs and writes to storage
  - Fast, but can take some effort to avoid overruns
How images are stored

• The image is stored as a sequence of values representing each pixel in the frame:
How images are stored

• The image is stored as a sequence of values representing each pixel in the frame:

<table>
<thead>
<tr>
<th>Address</th>
<th>Value</th>
<th>Pixel</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>0x000001</td>
<td>0x00</td>
<td></td>
</tr>
<tr>
<td>0x000002</td>
<td>0x00</td>
<td></td>
</tr>
<tr>
<td>0x000003</td>
<td>0x255</td>
<td></td>
</tr>
<tr>
<td>0x000004</td>
<td>0x00</td>
<td></td>
</tr>
<tr>
<td>0x000005</td>
<td>0x00</td>
<td></td>
</tr>
<tr>
<td>0x000006</td>
<td>0x00</td>
<td></td>
</tr>
<tr>
<td>0x000007</td>
<td>0x00</td>
<td></td>
</tr>
<tr>
<td>0x000008</td>
<td>0x00</td>
<td></td>
</tr>
<tr>
<td>0x000009</td>
<td>0x255</td>
<td></td>
</tr>
<tr>
<td>0x00000A</td>
<td>0x00</td>
<td></td>
</tr>
<tr>
<td>0x00000B</td>
<td>0x00</td>
<td></td>
</tr>
<tr>
<td>0x00000C</td>
<td>0x00</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>
How images are stored

- The image in memory is just one long, continuous data structure:

```plaintext
  0x0000
  0x0004
  0x0008
  0x000C
  0x0010
```
How image processing works

- Image operations are sequentially processed for each pixel, and stored in a parallel array
  - Eg. Differential kernel: \( q_i = p_i - p_{i-1} + 128 \)
How image processing works

- Pixel values in the image are retrieved with indirect memory reads
  - Adjacent columns use simple offsets
  - Adjacent rows require more complex mapping
  - Multiple channels make life more complex still

\[ n - r \]
\[ n - 1 \]
\[ n \]
\[ n + 1 \]
\[ n + r \]
\[ n + 2r + 2 \]

\[ n = \text{current pixel address} \]
\[ c = \text{length of row} \]
How microcontrollers work

- Microcontrollers execute instructions in memory to perform operations on data
  - One or more ‘working’ registers take and return values, depending on processor

<table>
<thead>
<tr>
<th>Address</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>…</td>
<td>…</td>
</tr>
<tr>
<td>0xn00001</td>
<td>0x00</td>
</tr>
<tr>
<td>0xn00002</td>
<td>0x00</td>
</tr>
<tr>
<td>0xn00003</td>
<td>0x255</td>
</tr>
<tr>
<td>0xn00004</td>
<td>0x00</td>
</tr>
<tr>
<td>0xn00005</td>
<td>0x00</td>
</tr>
<tr>
<td>0xn00006</td>
<td>0x00</td>
</tr>
<tr>
<td>0xn00007</td>
<td>0x00</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>…</td>
<td>…</td>
</tr>
<tr>
<td>0xm00001</td>
<td>0x128</td>
</tr>
<tr>
<td>0xm00002</td>
<td>0x128</td>
</tr>
<tr>
<td>0xm00003</td>
<td>0x255</td>
</tr>
<tr>
<td>0xm00004</td>
<td>0x128</td>
</tr>
<tr>
<td>0xm00005</td>
<td>0x128</td>
</tr>
<tr>
<td>0xm00006</td>
<td>0x128</td>
</tr>
<tr>
<td>0xm00007</td>
<td>0x128</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
</tr>
</tbody>
</table>
The fundamental problems

• Read from one structure, perform the computation and then write to another
  – Slow context switching ties up processor

• Microprocessors are ‘narrow pipes’, but images are ‘wide’ data
  – How can we process images fast enough?

• Performing operations on images is tedious
  – Lots of effort to write functional code
Other considerations

• Computer vision problems are often strict real-time, or time-sensitive
  – Must guarantee some achievable frame rate

• Interrupts for other functions of the microcontroller may disrupt image flow
  – Typically no or limited provision for pipelining or pre-caching
Max frame rate calculation

Is your microcontroller up to the job?

Good question!

• Simple calculation:
  1. How many pixels?
  2. How many computations per pixels?
  3. How much input read/mem write delay?
Max frame rate calculation

Eg. Simple level threshold

• 320x200 single-channel (greyscale) image
  – 64000 pixels total
  – 100 μs image read delay

• 60 MHz PIC processor
  – 1 operation per 4 clock cycles ~ 15 Mips

• 8 instructions per pixel
  – Byte read (1), compare (1), branch instruction (1), byte write (1), increment pointer (avg. 4)
Max frame rate calculation

- Each instruction = 4*1/60 = 0.0667 μs
- Total frame read and processing:
  - 64000*8* 0.0667 + 100 = 34 ms
  - Max frame rate is 29 Hz, without doing image output, other IO, servicing interrupts or pretty much anything else.

NB: Typical computer vision processes require hundreds or thousands of atomic operations.
Also, image encoding

- Most images aren’t just raw bits in memory
  - Still images are often compressed (e.g., JPEG)
  - Video images arrive in sequential frames which often have additional data and encoding

- The microcontroller may have to decode the image stream to operate on raw pixels
  - Probably have to re-encode them after, too
The moral of the story

• Embedded microcontrollers are really terrible for image processing
  – Limited clock speed, slow IO, narrow pipes

• Don’t do more than the most trivial CV tasks with microcontrollers
  – If you are going to, take the time to strictly optimise your code
Image processing for reals

• Use a real computer - Linux is your friend

• Use a mature CV package
  – OpenCV, Roborealm (Windows only), SimpleCV, or Matlab

• Spend the time getting your toolchain right
  – CV libraries typically come as precompiled machine code that must be linked in
  – Ideally, find someone else who has gotten it working on your platform and use their code
Image processing for reals

1. Pick your platform, “X”
2. Pick your programming language, “Y”
3. Find image processing library for X on Y
4. Spend the next week figuring out why it doesn’t work

Code + headers + precompiled libraries + compiler + linking = functional code
Some handy practical guides

OpenCV

C++:
- http://stackoverflow.com/questions/15634301/opencv-for-beginners
- http://opencv-srf.blogspot.ch/

Python:
- http://www.opencvpython.blogspot.ch/
Still having trouble?

• Please, *please* consider offboard processing
  – And then strongly consider using Matlab

• Use a fully-fledged Single Board Computer

• Also check out Roborealm – it’s easy!
Gratuitous project tips II

• Store energy for use in doldrums
• Sail slower
  – 5 mins is plenty of time
• Centreboards and fins
  – Not just for surfer hippies anymore!
• You will not be the first (nor the last) group to submerge your servos
  – Consider plastic baggy?
Questions?

bleep
Tune-in next time for...

Fluid Mechanics

or

“Back by popular demand!”

Fun fact: 98 per cent of all humans have parasites