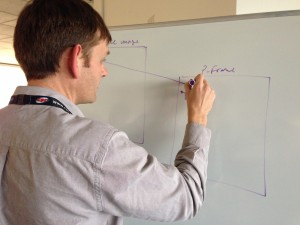
**Vectors from coarse motion estimation**

Liz: Gordon Hollingworth, our Director of Software, has been pointing the camera board at things, looking at dots on a screen, and cackling a lot over the last couple of weeks. We asked him what he was doing, so he wrote this for me. Thanks Gordon!

The Raspberry Pi is based on a BCM2835 System on a Chip (SoC), which was originally developed to do lots of media acceleration for mobile phones. Mobile phone media systems tend to follow behind desktop systems, but are far more energy efficient. You can see this efficiency at work in your Raspberry Pi: to decode H264 video on a standard Intel desktop processor requires GHz of processing capability, and many (30-40) Watts of power; whereas the BCM2835 on your Raspberry Pi can decode full 1080p30 video at a clock rate of 250MHz, and only burn 200mW.



Because we have this amazing hardware it enables us to do things like video encode and decode in real time without actually doing much work at all on the processor (all the work is done on the GPU, leaving the ARM free to shuffle bits around!) This also means we have access to very interesting bits of the encode pipeline that you’d otherwise not be able to look at.

One of the most interesting of these parts is the **motion estimation block** in the H264 encoder. To encode video, one of the things the hardware does is to compare the current frame with the previous (or a fixed) reference frame, and work out where the current macroblock (16×16 pixels) best matches the reference frame. It then outputs a set of vectors which tell you where the block came from – i.e. a measure of the motion in the image.

In general, this is the mechanism used within the application motion. It compares the image on the screen with the previous image (or a long-term reference), and uses the information to trigger events, like recording the video or writing a image to a disk, or triggering an alarm. Unfortunately, at this resolution it takes a huge amount of processing to achieve this in the pixel domain; which is silly if the hardware has already done all the hard work for you!

So over the last few weeks I’ve been trying to get the vectors out of the video encoder for you, and the attached animated gif shows you the results of that work. What you are seeing is the magnitude of the vector for each 16×16 macroblock equivalent to the speed at which it is moving! The information comes out of the encoder as side information (it can be enabled in raspivid with the -x flag). It is one integer per macroblock and is ((mb\_width+1) × mb\_height) × 4 bytes per frame, so for 1080p30 that is 120 × 68 × 4 == 32KByte per frame. And here are the results. (If you think you can guess what the movement you’re looking at here represents, let us know in the comments.)

Since this represents such a small amount of data, it can be processed very easily which should lead to 30fps motion identification and object tracking with very little actual work!

Go forth and track your motion!