# Stereo Matching on DECPeRLe$_1$

G. Berry, L. Moll, J. Vuillemin

April 28, 1993

## 1 Problem

From the two input images $a[x,y], b[x,y]$ (for $0 \leq x,y < N = 256$, 8b per pixel), compute the output matrix $d[x,y]$ where $d = d[x,y]$ is the ($5 = log_2(D)$ bits) index which achieves the maximum:

$$\begin{aligned} o[x,y] &= \max_{d<32} \{n[x,y,d] \times r[x+d,y]\} \\ n[x,y,d] &= \sum_{i,j<7} a[x+i,y+j] \times b[x+i+d,y+j] \\ r[x,y] &= 2^{24} \div \sqrt{m[x,y]} \\ m[x,y] &= \sum_{i,j<7} b[x+i,y+j] \times b[x+i,y+j] \end{aligned}$$

In these equations, $n, r$ and $m$ are 16b integers, while $o$ has 32 bits. To deal with out of bounds indices, pad both input matrices with zeroes on the boundaries (please check!). Let $K = 7$ be the window size, $D = 32$ the maximum distance between correlated windows and $N = 256$ the images size. We show below how to compute $d$ in $N(N+K)(D+2)$ operations. The corresponding pipe-lined hardware implementation requires less than $N^2(D+K+1)$ cycles. At 25MHz, we process each pair of images in less than $256^2 \times 40 \times 40\text{ns} = 100\text{ms}$, at a rate in excess of 10Hz.

# 2 Algorithm

We compute the numerator $n$ as follows:

$$\begin{aligned} p[x,y,d] &= a[x,y] \times b[x+d,y] \\ q[x,0,d] &= \sum_{j<7} p[x,j,d] \\ q[x,y+1,d] &= q[x,y,d] + p[x,y+7,d] - p[x,y,d] \\ n[0,y,d] &= \sum_{i<7} q[0,y,d] \\ n[x+1,y,d] &= n[x,y,d] + q[x+7,y,d] - q[x,y,d] \end{aligned}$$

Similarly for the denominator $m$:

$$\begin{aligned} p_2[x,y] &= b[x,y] \times b[x,y] \\ q_2[x,0] &= \sum_{j<7} p_2[x,j] \\ q_2[x,y+1] &= q_2[x,y] + p_2[x,y+7] - p_2[x,y] \\ m[0,y] &= \sum_{i<7} q_2[0,y] \\ m[x+1,y] &= m[x,y] + q_2[x+7,y] - q_2[x,y] \end{aligned}$$

The equations for $r$ and $o$ are as above.

# 3 Hardware implementation

The design processes the images in two stages: (D) computes $m$ and $r$ in $2N^2$ cycles; (N) computes $n$, $o$ and $d$ in $N^2D$ cycles.

## 3.1 Image Store

Both $a$ and $b$ are stored in RamW, four row-consecutive pixels per 32b word. As the next stage requires one pixel of $a$ and one of $b$ per cycle, we read RamW on even cycles. During the odd cycles, we write the next image pair $a', b'$ in RamW as it arrives from the input Fifo. At the rate of one input byte per two cycles, it takes $4N^2$ cycles to read in $a', b'$; DECPeRLe$_1$'s Rams

allow to only write one byte in each word, without modifying the remaining 3.

Note that 4 images stored in this format only occupy 1/8MB, so we have enough room in RamW to also store the $2^{16}$ entries of the inverse of square root table.

During (N), we traverse $a$ and $b$ along columns $x$ and $x+d$: $a_t = a[x,t], b_t = b[x+d,t]$ for $t < 256$. At the end of each column, increment $d \to d+1$; after 32 such increments, change $x \to x+1$ and $d \to 0$.

During (D), we traverse $b$ along column $x$: $a_t = b_t = b[x,t]$ for $t < 256$. At the end of each column, change $x \to x+1$. After $N^2$ cycles, wait another $N^2$ cycles for the Q store to be empty, and proceed with the (N) computation.

## 3.2 Q computation

This unit has two 8b inputs $a$ and $b$, which arrive on BusW[8..15] and BusW[16..23]. Compute the 16b $p$ through a single $8b \times 8b \to 16b$ multiplier (in M00, with M01 used for cornering the $b$ input). Feed $p$ through a $PM7$ unit (in M10) which computes the 16b of $q$.

In first approximation, the $PM7$ unit is:

```
PM7(p : [16]) = q : [16]
where
 p7 = Reg Reg Reg Reg Reg Reg Reg p;
 s = Add(16)(p, Reg q);
 q = Sub(16)(s, p7)
end where
```

In reality, we'll have to store $q$ in *carry-save* form in order to cope with 25MHz, and convert to binary through a retimed adder.

The output $q$ is sent on BusE[0..16] to SE, and on BusN[32..47] to SN, cornering in M20.

## 3.3 Q store

During (D), simply store all the $q_2[x,y]$ in RamE; remember that they arrive in *column order* (along the y axis) for $N^2$ cycles. After that, read them in *row order* (along the x axis) for the next $N^2$ cycles, and switch to (N).

During (D), use RamN and RamE to implement a double-buffer: one Ram receives $N \times D$ inputs $q[x+1, y, d]$ in *column order* (along the y axis), while the other one produces $N \times D$ outputs $q'[x, y, d]$ in *row order* (along the d axis).

## 3.4 n computation

The output $q'$ of the Q store appears on BusE[32..47] and feeds another $PM7$ unit which computes $n$ (in M33).

The result $n$ is shipped through BusW[32..47] and BusSW[16..31] to CSW for use during (D); it is also sent through direct connect (between M33 and M23 ) to the big multiplier, for use during (N).

## 3.5 r computation

During (D), the value of $n$ is used (by CSW) as an address for RamW. The inverse of square root output goes (through BusSW[0..15]) to RamS where it is stored, for use during the next (N).

## 3.6 d computation

During (D), the value of $n$ is multiplied by $r$ (from RamS) in a $16b \times 16b \rightarrow 32b$ multiplier (in M13,M23,M14,M24 so we have plenty of room). The multiplier's output feed a 32b Max unit, which produces one bit of decision on each cycle. This decision freezes the value of a 5b counter. Every 32 cycles, reset the Max unit, and send the counter's value (through BusN[0..4] and 5bits of BusNE) to the output Fifo.

The Max unit compares from high to low bits, with retiming every second bits (details on demand).