Maxeler Apps
Discrete Cosine Transform

Nov 2016

Discrete Cosine Transform (DCT)

- Similar to Fourier transform.
  Forms the basis of lossy compression in image/video codecs
- Converts blocks of pixels (colours) into same-sized blocks of frequency coefficients

- Key idea: discard “less important” coefficients and reduce precision of the rest
- Image can still be reconstructed (decoded) using remaining data

Image courtesy of https://people.xiph.org/~xiphmont/demo/daala/demo1.shtml

Maths underlying DCT

- In a 1D space, using DCT-II formulation:

$$X_k = \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{N}\left(n+\frac{1}{2}\right)k\right] \qquad k = 0, \ldots, N-1.$$

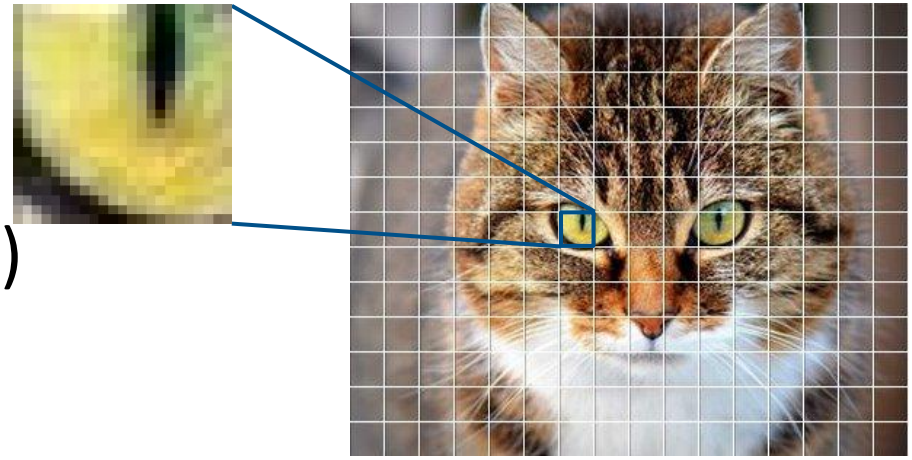
- In a 2D space, one can apply this transform to matrix rows and columns separately, which can be written as a matrix-matrix multiply:

$$V = CXC^T$$

where C is a matrix of cosine coefficients, X is an input matrix block, V is the transformed matrix block

Transformation results

- Applying DCT
  to a block of pixels
  (containing colour data)



- Results:

original pixel colours

184	133	119	252	64	183	81	86
181	227	84	52	253	19	178	173
68	194	86	151	206	109	84	255
210	13	75	17	67	220	62	252
97	181	248	161	108	74	248	33
45	76	85	42	96	7	216	164
202	46	59	152	155	143	152	109
156	227	126	224	191	188	220	33

frequency coefficients

134	-2	4	3	3	1	-7	2
1	2	3	-3	4	-7	3	-3
8	6	-6	5	2	2	-5	-5
-8	1	0	-3	-4	9	-1	2
5	5	-4	2	-6	7	6	-6
-6	-4	2	5	7	2	13	-8
-2	-3	-3	-6	-2	-2	-2	-4
7	5	-8	-3	-3	3	-4	-2

Low frequency coefficients more important; pruning part of high frequency data -- little loss in precision

Lossy compression

- Split an image into square 8×8 blocks of 8-bit pixels
- Encode each block independently as follows:
  - Apply discrete cosine transform
  - Reduce the entropy of the transformed block:
    - discard selected matrix elements
    - discard selected bits of the remaining elements
    - pack the remaining data into a 128 bit word
- The application encodes every 8×8 block of the input image (512 bits) in a 128 bit word: **4x compression**!
- Compression rate depends on how much is discarded

What is accelerated

DFE

- Implements the body of the loop:

```
for (i = 0; i < numBlocksOfPixels; i++) {
  for (j = 0; j < numBlocksOfPixels; j++) {
    transformed = discreteCosineTransform(imageBlock(i,j), blockSize);
    encodedBlock = discardAndPack(transformed);
  }
}
```

- Processes one block per cycle
- Evaluates 2 matrix-matrix multiplies amounting to 1024 multiplies and 1024 adds

CPU

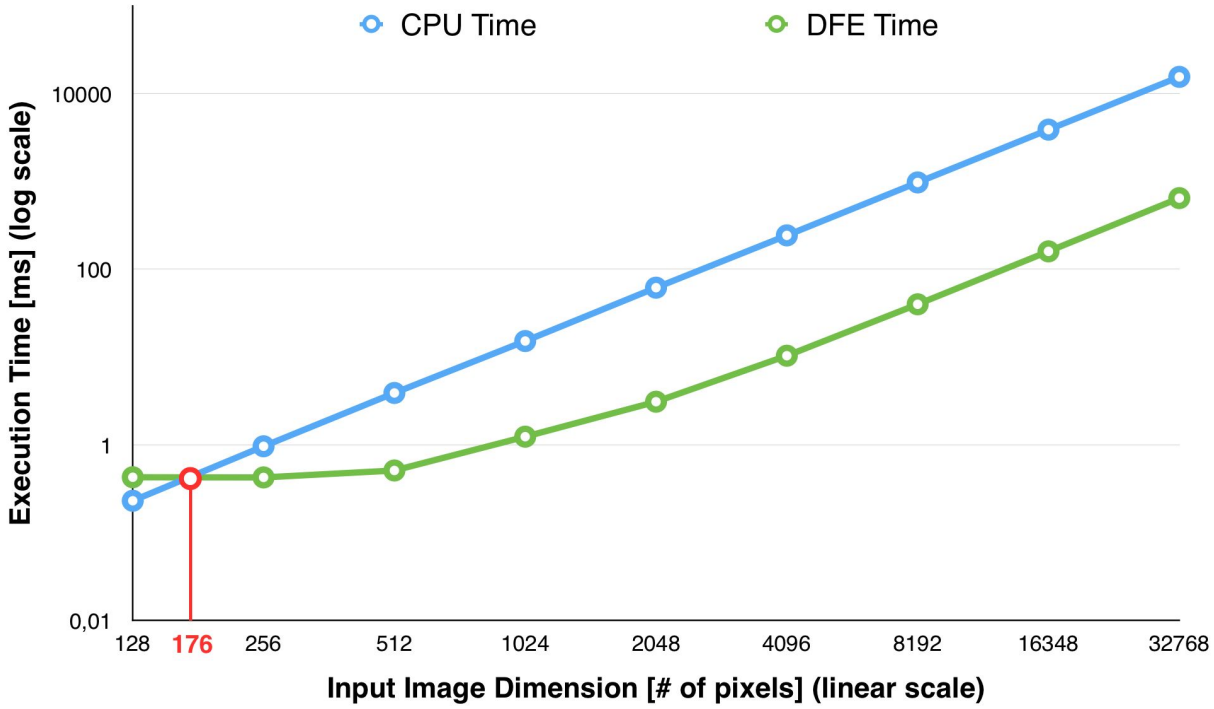
- Reorders image matrix to facilitate block traversal

Precision

- This application assumes input image is 8-bit grayscale
- Matrix of cosine values is precomputed on CPU using float32 representation
- DFE converts cosine values and pixel colours to a fixed point type with 10 integer bits and 14 fractional bits (10.14)
- Matrix-matrix multiplies are evaluated on DFE in 10.14 fixed point type
- Resulting frequencies are rounded to nearest integers

Results:

- Same execution time for 176x176 pixel images
- Speedup saturation around 24.5x for larger input images



- CPU Model: Intel(R) Xeon(R) CPU E5-2697 v2 @ 2.70GHz - 64GB RAM
- DFE Model: MAX4 MAIA Card @ 200Mhz