MEPI's Complexity

Without Noise

In the following we will denote addition, multiplication, division and logical comparison respectively with **a**,**m**, **d** and **c**. Considering just the component x and by omitting arguments, eq. (8) of the paper can be rewritten as:
$d_x = -P\ atan\left(\frac{Im(\hat{B}\hat{b}_0^*)}{Re(\hat{B}\hat{b}_0^*)}\right)$.

In fact, by multiplying both members of eq. (7) of the paper by the quantity $\frac{\hat{b}_0^*}{|\hat{b}_0|^2}$ and by extracting their phase we have,

$$d_x = -P\ atan\left(\frac{Im(\frac{\hat{B}\hat{b}_0^*}{|\hat{b}_0|^2})}{Re(\frac{\hat{B}\hat{b}_0^*}{|\hat{b}_0|^2})}\right).$$

Since $|\hat{b}_0|^2$ (as well as K) is a real number and it is common to the numerator and denominator of *atan* argument, it can be omitted in the following computation.

By exploiting the property of complex numbers it holds

$$\begin{aligned} Im(\hat{B}\hat{b}_0^*) &= Re(\hat{B})Im(\hat{b}_0^*) + Im(\hat{B})Re(\hat{b}_0^*) \\ Re(\hat{B}\hat{b}_0^*) &= Re(\hat{B})Re(\hat{b}_0^*) - Im(\hat{B})Im(\hat{b}_0^*). \end{aligned} \tag{1}$$

Since, $Im(\hat{b}_0^*) = -Im(\hat{b}_0)$, $Re(\hat{b}_0^*) = Re(\hat{b}_0)$, $Im(\hat{B}) = Im(\hat{b}_0) + Im(\sum_{k=1}^{K-1}\hat{b}_k)$ and $Re(\hat{B}) = Re(\hat{b}_0) + Re(\sum_{k=1}^{K-1}\hat{b}_k)$, eqs. (1) become

$$Im(\hat{B}\hat{b}_0^*) = -Re(\hat{b}_0)Im(\hat{b}_0) - Re(\sum_{k=1}^{K-1}\hat{b}_k)Im(\hat{b}_0) + Im(\hat{b}_0)Re(\hat{b}_0) +$$

$$+Im(\sum_{k=1}^{K-1}\hat{b}_k)Re(\hat{b}_0) = Im(\sum_{k=1}^{K-1}\hat{b}_k)Re(\hat{b}_0) - Re(\sum_{k=1}^{K-1}\hat{b}_k)Im(\hat{b}_0)$$

$$Re(\hat{B}\hat{b}_0^*) = (Re(\hat{b}_0))^2 + Re(\sum_{k=1}^{K-1}\hat{b}_k)Re(\hat{b}_0) + (Im(\hat{b}_0))^2 + Im(\sum_{k=1}^{K-1}\hat{b}_k)Im(\hat{b}_0).$$

Hence, for $K = 2$, i.e. $\hat{B} = \hat{b}_0 + \hat{b}_1$, the computation of *atan* argument, in terms of $\hat{b}_0$ and $\hat{b}_1$, requires 6 **m**s, 4 **a**s and 1 **d**.

For an $M \times M$ block, the computation of *atan* requires $log_2(M/2)+1$ **c**s: one to determine the sign of the motion component, the remaining $log_2(M/2)$ to determine its absolute value through the search in a binary tree (built once for all). In fact, the precomputed values, corresponding to all the allowed integer shifts for an $M \times M$ block (divided by P), are

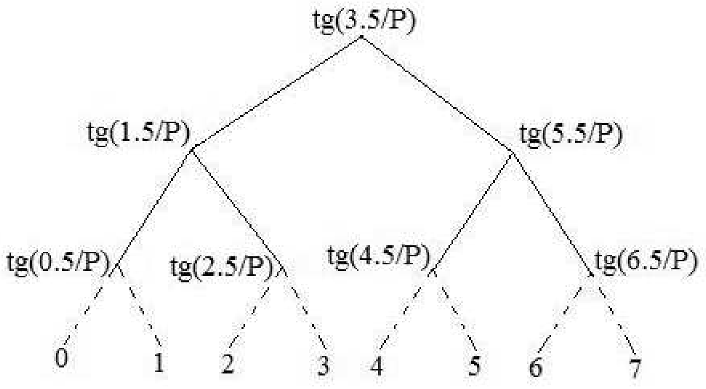

Figure 1: The tree built for computing *atan* for a 16×16 block. Nodes labels are the left extrema (divided by P) of the intervals centered on the admissible values. The dashed lines before leaves show that the tree recalled for the computation of operations is composed just of the solid part: the comparisons at the last level (of the solid part) of the tree lead to the admissible motion values (dashed leaves).

$M/2$ (i.e. half of the block dimension). It is the largest shift value that still guarantees a comparison among aligned blocks for motion estimation. The nodes of the tree contain the left extrema of the intervals (as happens in quantization) whose center is the shift value (i.e. d_x/P). Fig. 1 shows the tree for a 16×16 block: the allowed integer motion values are $d_x = 0, 1, 2, .., 7$ while $P = (2M-1)/((K-1)\pi\omega_x) = 4.9338$ for $\omega_x = 1$ and $K = 2$.

The computation of $-P$ requires 1 **m**, 1 **a** and 1 **d**.

With regard to $\hat{b}_0(\omega_x, 0)$, by exploiting the separability of the Fourier transform, $M(M-1)$ **a**s are required for the sum along columns (or rows) plus the operations required by the second coefficient of the 1D FFT of the resulting 'sum' signal. The FFT of the sum signal with length M can be computed by exploiting the symmetry of *cos* function in the unitary circle.

More precisely, let $\{a_k\}_{0 \le k \le M-1}$ be the sum signal. Its FFT at frequency value $\omega = 1$ is

$$\hat{a}(1) = \sum_{k=0}^{M-1} a_k e^{-i\frac{2\pi k}{M}}$$

and then

$$Re(\hat{a}(1)) = \sum_{k=0}^{M-1} a_k cos(\frac{2\pi k}{M})$$

$$Im(\hat{a}(1)) = \sum_{k=0}^{M-1} a_k sin(\frac{2\pi k}{M}).$$

Let us first analyze the *Re* component (note that $\hat{a}(1) = \hat{b}_0(1, 0)$). The distinct absolute values of *cosine* function are the ones that are defined in the first quadrant of the Cartesian plane, i.e. for $k = 0, 1, 2...., M/4$. In the other quadrants, *cosine* function just changes its sign. More precisely, $\forall\ k = 0, 1, 2...., M/4$ we have:

$$cos(2k\pi/M) = -cos(2(M/2-k)\pi/M)$$

$$cos(2k\pi/M) = -cos(2(M/2+k)\pi/M)$$

$$cos(2k\pi/M) = cos(2(M-k)\pi/M).$$

Hence, $Re(\hat{a}(1))$ can be rewritten as follows

$$Re(\hat{a}(1)) = (a_0 - a_{M/2})cos(0) + \sum_{k=1}^{M/4-1} \left(a_k - a_{M/2-k} - a_{M/2+k} + a_{M-k}\right) cos(2\pi k/M) +$$

$$+(a_{M/4} - a_{3M/4})cos(\pi/2).$$

Since $cos(0) = 1$ and $cos(\pi/2) = 0$, the computation of $Re(\hat{a}(1))$ requires $M - 3$ **as** and $M/4 - 1$ **ms**. With regard to the Im part, we can use the same strategy. In fact, $\forall\ k = 0, 1, 2...., M/4$ we have:

$$sin(2k\pi/M) = sin(2(M/2-k)\pi/M)$$

$$sin(2k\pi/M) = -sin(2(M/2+k)\pi/M)$$

$$sin(2k\pi/M) = -sin(2(M-k)\pi/M).$$

Then $Im(\hat{a}(1))$ can be rewritten as follows

$$Im(\hat{a}(1)) = (a_0 + a_{M/2})sin(0) + \sum_{k=1}^{M/4-1} \left(a_k + a_{M/2-k} - a_{M/2+k} - a_{M-k}\right) sin(2\pi k/M) +$$

$$+(a_{M/4} - a_{3M/4})sin(\pi/2).$$

Let us now consider the quantity $\left(a_k + a_{M/2-k} - a_{M/2+k} - a_{M-k}\right)$.

It can be written as $\left(a_k - a_{M/2+k}\right) - \left(a_{M-k} - a_{M/2-k}\right)$. As it can be observed, the two quantities $\left(a_k - a_{M/2+k}\right)$ and $\left(a_{M-k} - a_{M/2-k}\right)$ have already been computed during the calculation of the real part, hence it is not necessary to recompute them. It turns out that, the computation of $Im(\hat{a}(1))$ requires $M/2 - 1$ **as** and $M/4 - 1$ **ms**.

Moreover, 2 **ds** are required for the normalization of FFT. The same amount of operations is required for $\hat{b}_1(\omega_x, 0)$ (or for $\sum_{k=1}^{K-1} \hat{b}_k$ in case of $K > 2$). Finally the $\sum_{k=1}^{K-1} \hat{b}_k$ requires $(K-2)M^2$ **as**.

Considering both x and y direction, the total amount of operations required by the algorithm is

	Additions	**Multiplications**	**Divisions**	**Comparisons**
MEPI without noise	1054	34	12	8
MEPI with noise	1826	303	22	10

Table 1: Number of additions (and subtractions), multiplications, divisions and comparisons required by MEPI for a 16×16 block and $K = 2$. **Top** without noise, **Bottom** with noise in the steps 3.b and 3.c of the algorithm, i.e. when space filling curves are adopted.

$$\underbrace{(K-2)M^2}_{sumofsubsequentframes} + \underbrace{4M(M-1)+4(M-3+M/4-1+M/2-1+M/4-1+2)}_{FFTcomputation} +$$

$$+\underbrace{2(6+4+1)}_{atanargument} + \underbrace{2(1+1+1)}_{constantP} + \underbrace{2(log_2(M/2)+1)}_{atanvalue}$$

that is

$$\underbrace{(K-2)M^2}_{sumofsubsequentframes} + \underbrace{4M^2+4M-16}_{FFTcomputation} +$$

$$+ \underbrace{22}_{atanargument} + \underbrace{6}_{constantP} + \underbrace{2log_2(M/2)+2}_{atanvalue}$$

that is

$$(K+2)M^2 + 4M + 2log_2(M/2) + 14.$$

Then, for a block of size $M \times M$,

$$(K+2) + 4/M + (2/M^2)log_2(M/2) + 14/M^2$$

operations are required for each pixel.

Hence, for $M = 16$ and $K = 2$, 4.3281 operations per pixel are required, while the complexity is $O((K+2)M^2)$.

A short taxonomy in terms of how many occurrences are required for each kind of operation is shown in Table I.

With Noise

In case of noise, the error bound in eq. (11) requires 6 **a**s, 3 **d**s and 8 **m**s. In fact, $|\hat{c}_0|^2 = Re^2(\hat{c}_0) + Im^2(\hat{c}_0)$ while $|\hat{C}|^2 = Re^2(\hat{C}) + Im^2(\hat{C}) = (Re(\hat{c}_0) + Re(\sum_{k=1}^{K-1} \hat{c}_k))^2 + (Im(\hat{c}_0) +$

$Im(\sum_{k=1}^{K-1}\hat{c}_k))^2$. Moreover, $\sum_{k=1}^{K-1}c_k$ requires $(K-2)M^2$ **a**s, while $Re(\hat{c}_0), Im(\hat{c}_0), Re(\sum_{k=1}^{K-1}\hat{c}_k)$ and $Im(\sum_{k=1}^{K-1}\hat{c}_k)$ require $2M(M-1)+2(M-3+M/4-1+M/2-1+M/4-1+2)$ that is $2M(M-1)+2(2M-4)$, as shown in the previous section. The same amount of operations is required for the evaluation of $\Delta^2 y$.

Two **c**s (with the tolerance value τ) occur in step 3) of the algorithm. In case of step 3.a), we proceed as the case without noise, and then we can compute the motion vector (d_x, d_y) through equations (8) and (9) of the paper. Thus, the total amount of operations is

$$\underbrace{(K-2)M^2}_{sumofsubsequentframes} + \underbrace{4M(M-1)+4(2M-4)}_{FFTcomputation} +$$

$$+\underbrace{2(6+3+8)}_{errorbound} + \underbrace{2}_{logicalcomparisons} + \underbrace{2(6+4+1)}_{atanargument} + \underbrace{2(1+1+1)}_{constantP} + \underbrace{2(log_2(M/2)+1)}_{atanvalue},$$

that is

$$(K+2)M^2 + 4M - 16 + 34 + 2 + 22 + 6 + 2log_2(M/2) + 2,$$

that is

$$(K+2)M^2 + 4M + 2log_2(M/2) + 50.$$

Hence, for $K = 2$ and $M = 16$, in case of step 3.a), 4.4687 operations per pixels are required.

In the case 3.d), we have the same operations except for the *atan* computation, since motion cannot be estimated and it is set to $(0,0)$.

On the contrary, in case of step 3.b) or 3.c) a space filling curve is introduced for one direction. It means that the two (for $\hat{c}_0$ and $\hat{c}_1$) 1D FFTs on M^2 samples (organized along one of the 2 space filling curves) need $2(2M^2-4)$ operations. Moreover, the computation of *atan* is still obtained by searching among all its possible (precomputed) values, corresponding to all possible shifts, $[-M/2+Md_x, M/2+Md_x]$ in step 3.b) (or $[-M/2+Md_y, M/2+Md_y]$ in step 3.c)); the number of comparisons is still $log_2(M/2)+1$. More precisely, in case of step 3.b) the component d_x is computed using the 2D algorithm while d_y is unknown. Nonetheless, for each value of d_x, i.e. $-M/2, -M/2+1, ..., -1, 0, 1, M/2-1, M/2$, it is possible to precompute the values of *atan* for each admissible integer value of d_y. In other words, the admissible motion along the adopted space filling curve falls in the range $[-M/2+Md_x, M/2+Md_x]$. Since Md_x is known and d_y must assume integer (positive and negative) values, $log_2(M)$ **c**s are required for the search in the binary tree. Notice that $log_2(M) = log_2(M/2)+1$ since the tree now contains both negative and positive values.

Summing up, in case of step 3.b) or 3.c), the total amount of operations is then

$$\underbrace{(K-2)M^2}_{sumofsubsequentframes} + \underbrace{4M(M-1)+4(2M-4)}_{FFTcomputation} + \underbrace{2(6+3+8)}_{errorbound} + \underbrace{2}_{logicalcomparisons} +$$

$$+ \underbrace{2(2M^2-4)}_{FFTcomputationSFC} + \underbrace{2(6+4+1)}_{atanargument} + \underbrace{3+4}_{constantP} + \underbrace{2(log_2(M/2)+1)}_{atanvalue},$$

that is

$$(K+6)M^2 + 4M + 2log_2(M/2) - 16 + 34 + 2 - 8 + 22 + 7 + 2$$

and then

$$(K+6)M^2 + 4M + 2log_2(M/2) + 43.$$

It turns out that in case of step 3.a) or 3.d) of the algorithm the total amount of operations per pixel is

$$(K+6) + 4/M + (2/M^2)log_2(M/2) + 43/M^2,$$

while its complexity is $O((K+6)M^2)$.

For K=2 and M=16, it corresponds to 8.4414 operations per pixel.