Fast Rotations: Low-cost arithmetic methods for orthonormal rotation

Gerben J. Hekstra
Department of Electrical Engineering
Delft University of Technology,
Delft, The Netherlands
gerben@cas.et.tudelft.nl

Ed F.A. Deprettere
Department of Electrical Engineering
Delft University of Technology,
Delft, The Netherlands
ed@cas.et.tudelft.nl

Abstract

In this paper, we introduce a new type of arithmetic operation that we have called "fast rotations" or "orthonormal μ-rotations". These are methods for orthonormal rotation over a set of fixed angles, with a very low cost in implementation, basically a few shift and add operations as opposed to lengthy Cordic operations. We also present the underlying theory for the construction of such fast rotation methods. Furthermore, we give examples where fast rotations have been applied successfully in a wide variety of applications. These include the low-cost and robust implementation of FIR filter banks for image coding, the generation of spherical sample rays in 3D graphics, and the computation of the Eigenvalue decomposition (EVD) and singular value decomposition (SVD).

1. Introduction

Fast rotations[3], also called orthonormal μ-rotations[1], are arithmetic methods for performing orthonormal rotation at very low cost in implementation. Related to Cordic[9, 10], they form a viable, low-cost alternative to the more expensive Cordic arithmetic for certain applications. Although fast rotations exist only for certain angles of rotation, they form a sufficient set to efficiently implement any orthogonal operation.

Fast rotations have already been applied in a number of applications in signal processing, image processing, computer graphics and array processing. All with very promising results, as their application has in some cases led to a reduction of the computational complexity by more than one order.

1.1. Outline of this paper

In section 2, we define the fast rotation, and present the first few fast rotations methods, along with their properties. A taxonomy of the different classes of fast rotations is given as well. In order to get a better understanding of the underlying theory, we introduce a polynomial representation for fast rotations in section 3 and also present the relationship to hyperbolic fast rotations. In section 4, we present methods of construction that produce new, higher order fast rotations of arbitrary precision. Finally, in section 5 we give a brief overview of some of the applications where fast rotations have been used, with great success, and where they have led to drastic reductions of the computational complexity.

2. Fast Rotation Methods

A fast rotation over an angle α is given by the matrix F as:

$$F = \begin{bmatrix} \hat{c} & -\hat{s} \\ \hat{s} & \hat{c} \end{bmatrix}, \tag{1}$$

where $(\hat{c}, \hat{s})$ is a pairwise approximation of a (cosine, sine) pair, with the magnification factor m given by:

$$m = \sqrt{\hat{c}^2 + \hat{s}^2}, \tag{2}$$

the rotation angle α given by:

$$\alpha = \arctan(\hat{c}, \hat{s}), \tag{3}$$

and satisfying the following conditions:

1. **close-to-orthonormal**
   The magnification factor m, which is the norm of F, is close to unity, *i.e.* $m = 1 + \varepsilon$. The error ε falls well below the precision of the number representation. Hence, this error is negligible compared to that of rounding-off the data, making the operation in practice *orthonormal.* We also call ε the *magnification error.*

2. **low-cost implementation**
   The fast rotation F is cheap to implement in terms of

hardware cost, in the order of a few shift and add operations. This cost is a function of the magnitude of the angle of rotation and of the required precision.

In the subsequent sections we present the first few fast rotation methods and a generalization for a subset of these fast rotations.

For the sake of simplicity in the formulas, we will currently assume positive rotations in the first quadrant only, with $\hat{c}, \hat{s} > 0$.

2.1. Method I fast rotations

The simplest of the fast rotation methods is identical to the standard Cordic[9] micro-rotation, with the approximation pair given by:

$$\begin{array}{rcl} \hat{c} & = & 1 \\ \hat{s} & = & 2^{\kappa} \end{array}, \qquad (4)$$

where the parameter κ is a non-positive integer and, due to its nature, is also called the *angle exponent*. Both the magnification factor m and angle of rotation α are functions of this angle exponent and, after substitution of (4) into Equations (2) and (3), are given by:

$$m = \sqrt{1+2^{2\kappa}} \qquad (5)$$

and

$$\alpha = \arctan(2^{\kappa}). \qquad (6)$$

Obviously, given a required precision, not all values of the angle exponent κ are suitable. For small enough values of κ, the error in magnification becomes negligible, and hence any additional scaling is unnecessary. We aim to use these rotations only in that domain for which this indeed occurs.

In most cases, we let the finite word length in the computations determine the required precision of the fast rotations. We assume the size of the mantissa to be N_{mant} bits, with the weight of the least significant bit given by lsb $= 2^{-N_{\text{mant}}}$. When employing rounding-to-nearest, the error due to rounding is given by $\varepsilon_{\text{round}} = \text{lsb}/2$. For rounding down, the error is given by $\varepsilon_{\text{chop}} = \text{lsb}$. For the sake of simplicity, we will assume the rounding-down model, and base our results on it. Equivalent results can be found for the round-to-nearest model.

For a fast rotation to be considered orthonormal, the magnification factor m must satisfy the condition:

$$1-\varepsilon_{\text{chop}} < m < 1+\varepsilon_{\text{chop}}. \qquad (7)$$

By inspection of Equation (5), it follows that $m > 1$, so we need only consider the right side inequality. Substitution of m, and squaring of both sides results in:

$$1+2^{2\kappa} < 1+2\varepsilon_{\text{chop}}+\varepsilon_{\text{chop}}^{2}. \qquad (8)$$

Subtracting 1 from both sides, and tightening the condition by removing the $\varepsilon_{\text{chop}}^{2}$ term on the right-hand side, we arrive at:

$$2^{2\kappa} \leq 2\varepsilon_{\text{chop}} < 2\varepsilon_{\text{chop}}+\varepsilon_{\text{chop}}^{2}. \qquad (9)$$

Substituting $\varepsilon_{\text{chop}}$ by $2^{-N_{\text{mant}}}$, and writing out the left hand inequality of (9) in terms of the exponents only, we arrive at the upper bound κ_u of the angle exponent as:

$$\kappa \leq \kappa_u = \frac{-N_{\text{mant}}+1}{2}. \qquad (10)$$

This result is well known from Cordic literature, where the "tail" of micro-rotations with $\kappa \leq \kappa_u$ do not influence the magnitude of the scaling factor any more.

Additionally, we can also define a lower bound for the angle exponent. The largest shift which has to be accounted for is the term 2^{κ} in $\hat{c}$, which is not allowed to exceed the width of the mantissa. This leads to the lower bound κ_l for the angle exponent in:

$$\kappa > \kappa_l = -N_{\text{mant}}. \qquad (11)$$

For values of the angle exponent κ beyond this bound, the fast rotation degrades to the identity operator.

The implementation of the fast rotation is illustrated by the rotation of the vector $[v_x\, v_y]^T$ over the angle α to the vector $[v'_x\, v'_y]^T$ in:

$$\begin{bmatrix} v'_x \\ v'_y \end{bmatrix} = F \cdot \begin{bmatrix} v_x \\ v_y \end{bmatrix}. \qquad (12)$$

Writing out the separate equations for the resulting vector $[v'_x v'_y]^T$, and substituting (4) leads to the classical Cordic equation.

$$\left\{ \begin{array}{rcl} v'_x & = & v_x - 2^{\kappa} v_y \\ v'_y & = & v_y + 2^{\kappa} v_x \end{array} \right. . \qquad (13)$$

It is plain to see, and well known from Cordic literature[9], that this operation can be done in direct form with two shift- and two add operations.

We prefer to express the cost of implementation in terms of **shift-add pairs**, as the computation in (13) can be performed pairwise for v'_x, v'_y.

2.2. Method II fast rotations

The next fast rotation method is based on the more accurate approximation pair:

$$\begin{array}{rcl} \hat{c} & = & 1-2^{2\kappa-1} \\ \hat{s} & = & 2^{\kappa} \end{array}. \qquad (14)$$

The magnification factor m and rotation angle α follow from substitution of the $\hat{c}, \hat{s}$ pair of Equation (14) into (2) and (3), and are given by:

$$m = \sqrt{1+2^{4\kappa-2}} \qquad (15)$$

and

$$\alpha = \arctan(\frac{2^{\kappa}}{1-2^{2\kappa-1}}). \qquad (16)$$

Note that, for the same angle exponent κ, this method produces rotations over angles of similar magnitude, but with a higher precision, in terms of how close m is to unity, as compared to Method I with the approximation pair of Equation (4).

Note also that, in writing out m^2 as the sum of the squares of $\hat{c}$ and $\hat{s}$, all terms cancel out except for the lowest power 1 and the highest power $2^{4\kappa-2}$. We call such fast rotations **maximal**, as the maximum number of terms cancel out against each other. The term with the highest power of of 2^{κ}, is the only term remaining that is a function of the angle exponent κ, and hence the smallest possible error ε is attained. The Method I fast rotations, although trivial, are also maximal.

We can use this rotation method, given the same required precision, for larger values of the angle exponent κ than the previous method. The upper bound κ_u for the angle exponent follows from a similar derivation as in Equations (7) to (11) as:

$$\kappa \leq \kappa_u = \frac{-N_{\mathrm{mant}}+3}{4}. \qquad (17)$$

The lower bound κ_l for this method follows from the limit value of the angle exponent κ for which the term $2^{2\kappa-1}$ in $\hat{c}$ vanishes, given a mantissa size of N_{mant}. This happens for

$$\kappa > \kappa_l = \frac{-N_{\mathrm{mant}}+1}{2}. \qquad (18)$$

Note that this lower bound is the same as the upper bound of Method I. At this boundary, the term $2^{2\kappa-1}$ in $\hat{c}$ vanishes, and the method automatically degrades to the Method I fast rotation.

Concerning its implementation; writing out Equation (12), substituting (14), leads to:

$$\left\{ \begin{array}{lcl} v'_x & = & v_x - 2^{\kappa}v_y - 2^{2\kappa-1}v_x \\ v'_y & = & v_y + 2^{\kappa}v_x - 2^{2\kappa-1}v_y \end{array} \right. . \qquad (19)$$

The cost of implementing this fast rotation is hence 4 shift- and 4 add operations, or two shift-add pairs.

2.3. Method III fast rotations

The next fast rotation method, Method III, is yet more accurate, and is given by the approximation pair:

$$\begin{array}{lcl} \hat{c} & = & 1-2^{2\kappa-1} \\ \hat{s} & = & 2^{\kappa}-2^{3\kappa-3} \end{array} . \qquad (20)$$

The magnification factor m and rotation angle α are given by:

$$m = \sqrt{1+2^{6\kappa-6}} \qquad (21)$$

and

$$\alpha = \arctan(\frac{2^{\kappa}-2^{3\kappa-3}}{1-2^{2\kappa-1}}). \qquad (22)$$

Note that this is, due to the form of m in Equation (21), also a **maximal** fast rotation.

The upper bound κ_u for the angle exponent for this method follows from an analogous derivation as of Equations (7) to (11) as:

$$\kappa_u = \frac{-N_{\mathrm{mant}}+7}{6}. \qquad (23)$$

The lower bound κ_l follows in a similar way as for the previous methods as:

$$\kappa_l = \frac{-N_{\mathrm{mant}}+3}{3}. \qquad (24)$$

which shows a slight overlap to the upper bound of the Method II fast rotations.

Concerning the implementation; writing out Equation (12), substituting (20), leads to:

$$\left\{ \begin{array}{lcl} v'_x & = & v_x - 2^{\kappa}v_y - 2^{2\kappa-1}v_x + 2^{3\kappa-3}v_y \\ v'_y & = & v_y + 2^{\kappa}v_x - 2^{2\kappa-1}v_y - 2^{3\kappa-3}v_x \end{array} \right. . \qquad (25)$$

The cost of implementing this fast rotation is three shift-add pairs. Note that the cost in shift-add pairs for the direct form implementation of all the fast rotations is equal to the sum of the number of terms in $\hat{c}$ and $\hat{s}$ minus one, to account for the term 1 in $\hat{c}$. From the trend of the fast rotation methods presented so far, we can see that a higher accuracy at a fixed order of magnitude of the angle rotation, or a larger angle of rotation at a fixed accuracy both naturally imply a higher cost of implementation.

2.4. Higher order fast rotations

These first three simple forms of fast rotation have been found by solving for $\hat{c}$ and $\hat{s}$ having a fixed number of power-of-two terms, and constraining m so that maximal fast rotations are found. While this is possible for a small number of terms, it becomes unmanageable for higher order fast rotations. Moreover, they do not always exist for certain combinations of the number of terms.

In the case when $\hat{c}$ and $\hat{s}$ have three and two terms respectively, no maximal solution is found. Instead, we propose the Method IV rotation as given by the approximation pair:

$$\begin{array}{lcl} \hat{c} & = & 1-2^{2\kappa-1}-2^{4\kappa-3} \\ \hat{s} & = & 2^{\kappa}-2^{5\kappa-4} \end{array} . \qquad (26)$$

The magnification factor m and rotation angle α are given by:

$$m = \sqrt{1+2^{8\kappa-6}+2^{10\kappa-8}} \qquad (27)$$

and

$$\alpha = \arctan\left(\frac{2^{\kappa} - 2^{5\kappa-4}}{1 - 2^{2\kappa-1} - 2^{4\kappa-3}}\right). \qquad (28)$$

Note that this is *not* a maximal fast rotation as we can see by the form of m in Equation (21).

Likewise, we propose the Method V rotation, which again is maximal, as given by the approximation pair:

$$\begin{aligned} \hat{c} &= 1 - 2^{2\kappa-1} + 2^{4\kappa-3} \\ \hat{s} &= 2^{\kappa} - 2^{3\kappa-3} + 2^{5\kappa-5} \ . \end{aligned} \qquad (29)$$

The magnification factor m and rotation angle α are given by:

$$m = \sqrt{1 + 2^{10\kappa-10}} \qquad (30)$$

and

$$\alpha = \arctan\left(\frac{2^{\kappa} - 2^{3\kappa-3} + 2^{5\kappa-5}}{1 - 2^{2\kappa-1} + 2^{4\kappa-3}}\right). \qquad (31)$$

A general expression exists for a subset of maximal fast rotations, parameterized in N, with the approximation pair given by:

$$\begin{aligned} \hat{c} &= 2\sum_{i=0}^{N}(-x^2)^i - 1 \\ \hat{s} &= x\left(2\sum_{i=0}^{N}(-x^2)^i - (-x^2)^N\right) \ , \end{aligned} \qquad (32)$$

where the variable x is introduced both for sake of simplicity, and to illustrate the possibility of a polynomial representation. Substituting $x = 2^{\kappa}$ for $N = 0$, and $x = 2^{\kappa-1}$ for $N \geq 1$ leads to fast rotation methods, expressed as a function of the angle exponent κ.

The magnification factor m follows from substitution of $\hat{c}, \hat{s}$ of Equation (32) into (2) and by successive elimination, and is given by

$$\begin{aligned} m &= \sqrt{1 - (-x^2)^{2N+1}} \\ &= \sqrt{1 + x^{4N+2}} \ , \end{aligned} \qquad (33)$$

which shows that the method is indeed maximal.

When implemented in a direct form, the cost of this general method is equal to $(2N + 1)$ shift-add pairs. In practice, it does not pay to go beyond Method III ($N = 1$) with direct form implementation, as more efficient *factored* implementation schemes exist for certain higher order fast rotations, as we shall see in the following sections.

The previously presented Methods I, III and V are the first three members of this series for $N = 0, 1, 2$ respectively. Note well that the terms in $\hat{c}$ and $\hat{s}$ do *not* follow the Taylor series expansion of $\cos(x)$ and $\sin(x)$. The fast rotation methods generated by Equation (32) only form a subset of the possible maximal fast rotations.

2.5. Taxonomy

So far, we have seen fast rotation methods which, when applied only in a specific domain of the angle exponent, are considered *orthonormal* within a given precision. No additional scaling was used to bring them to this precision.

We can classify this group, and also others, according to the taxonomy shown in Figure 1. The aforementioned group

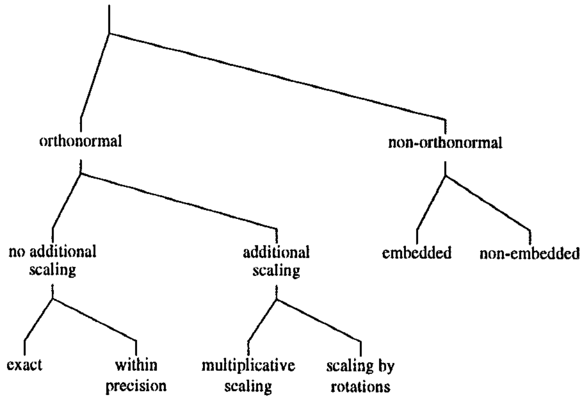

Figure 1. Taxonomy of fast rotations

would be classified as: *Orthonormal, no additional scaling, within precision.*

We will not treat all classes in detail, but we will try to give a flavour of some of them. Our main interest goes towards the *orthonormal* class. The *non-orthonormal* class contains methods which are used outside of their range of being orthonormal, in the special case that the application allows it. We will currently leave this class out of consideration.

Of this orthonormal class, we have a further dichotomy into methods without additional scaling and those *with.*

- **no additional scaling** Methods belonging to this class require no further scaling to bring them to a required precision. The choice of the $\hat{c}, \hat{s}$ approximation pair already gives accurate enough methods. One further dichotomy is made into methods which are exact and those that work within the precision.
    - **within precision** These are methods like the ones that we have already shown. They are only used in a certain domain of the angle exponent, such that the magnification error falls well within the round-off error, making the method in practice orthonormal.
    - **exact** These are methods for which $\varepsilon = 0$ under all conditions, i.e. they are already orthonormal

by themselves. For circular rotation, these methods are trivial rotations over $\frac{k\pi}{2}$. However, for hyperbolic rotation, there are non-trivial solutions, such as Walther's[10] method to extend the range of hyperbolic Cordic.

- **additional scaling** Methods belonging to this class have in common that they are the product of a base rotation, which is not accurate enough by itself to be a fast rotation, and a necessary scaling operation to bring the whole to the required precision. The scaling operation is performed as either:
  - **multiplicative scaling** The scaling occurs with multiplicative scaling steps, decomposed as a sequence of shift-add operations.
  - **scaling rotations** The scaling occurs with (a sequence of) extra rotations, of which the magnification error compensates that of the base rotation, such that the *overall* magnification factor is within the required precision.

We will present methods of construction in section 4 that are related to some of the above classes.

3. Polynomial representation

We have already seen an example of a polynomial representation for the maximal fast rotations of Equation (32).

We will use a polynomial representation as it facilitates the representation of fast rotations, simplifies the proofs, and allows us to gain more insight into the underlying theory.

The polynomial representation of an approximation pair $\hat{c}, \hat{s}$ is given by two polynomials in x: $c(x)$ and $s(x)$, where the powers of the angle exponent in 2^{κ} are replaced by powers of the parameter x.

For example, the polynomial approximation pair for the fast rotation Method I is given by:

$$\begin{array}{rcl} c(x) & = & 1 \\ s(x) & = & x \end{array} . \tag{34}$$

Likewise, the polynomial representation of the magnification $m(x)$ and angle of rotation $\alpha(x)$ are given by

$$m(x) = \sqrt{1+x^2} \tag{35}$$

and

$$\alpha(x) = \arctan(x) . \tag{36}$$

The relation between the approximation pair $\hat{c}, \hat{s}$ and the polynomials $c(x), s(x)$ is given by:

$$\begin{array}{rcl} \hat{c} & = & c(2^{\kappa}) \\ \hat{s} & = & s(2^{\kappa}) \end{array} . \tag{37}$$

We can state that the polynomials $c(x)$ and $s(x)$ *generate* approximation pairs for the rational points $x = 2^{\kappa}$.

The polynomial approximation pair $c(x), s(x)$ of Equation (34) is **canonic**. This means that the term with the lowest power of x in $s(x)$ is x itself. The polynomial approximation pairs that we will work with are not necessary canonic, but can be made to be with a simple transformation. However, we *do* adhere to the convention that the lowest power of x in $s(x)$ must be 1.

As a second example, the polynomial representation of the fast rotation Method II is given by:

$$\begin{array}{rcl} c(x) & = & 1-2x^2 \\ s(x) & = & 2x \\ m(x) & = & \sqrt{1+4x^4} \\ \alpha(x) & = & \arctan\left(\frac{2x}{1-2x^2}\right) \end{array} . \tag{38}$$

Note that this representation is on purpose not canonic, so as to avoid fractional coefficients in the polynomials. The Equations (14) to (16) are obtained by substitution of x by $2^{\kappa-1}$.

The polynomial representation of the fast rotation methods I through to V are presented in Table 1.

To aid further discussion, we introduce the auxiliary polynomial $U_N(x)$ as being

$$U_N(x) = \sum_{i=0}^{N} x^i . \tag{39}$$

We will use this polynomial extensively in the next sections, both to aid a compact representation, and to simplify proofs, using the set of rules below.

$$(1-x)U_N(x) = 1-x^{N+1} \tag{U.1}$$

$$U_N(x) = \frac{1-x^{N+1}}{1-x} \tag{U.2}$$

$$(1+x^N)U_{N-1}(x) = U_{2N-1}(x) \tag{U.3}$$

$$(1+x^{2^L})U_{2^L-1}(x) = U_{2^{L+1}-1}(x) \tag{U.4}$$

$$U_{2^L-1}(x) = \prod_{l=0}^{L-1} 1+x^{(2^l)} \tag{U.5}$$

$$U_{N-1}(x)U_{M-1}(x^N) = U_{MN-1}(x) \tag{U.6}$$

For the sake of brevity, we omit the proofs for the above rules.

We are now able to compactly represent the maximal fast rotations of Equation (32) as:

$$\begin{array}{rcl} c(x) & = & 2U_N(-x^2)-1 \\ s(x) & = & x\left(2U_N(-x^2)-(-x^2)^N\right) \\ m(x) & = & \sqrt{1-(-x^2)^{2N+1}} \\ \alpha(x) & = & \arctan\left(x\frac{2-(-x^2)^N-(-x^2)^{N+1}}{1-(-x^2)^N-2(-x^2)^{N+1}}\right) \end{array} . \tag{40}$$

method	$c(x)$	$s(x)$	$m(x)$	canonic	maximal	substitution
I	1	x	$\sqrt{1+x^2}$	yes	yes	$x=2^{\kappa}$
II	$1-2x^2$	$2x$	$\sqrt{1+4x^4}$	no	yes	$x=2^{\kappa-1}$
III	$1-2x^2$	$2x-x^3$	$\sqrt{1+x^6}$	no	yes	$x=2^{\kappa-1}$
IV	$1-2x^2-2x^4$	$2x-2x^5$	$\sqrt{1+4x^8+4x^{10}}$	no	no	$x=2^{\kappa-1}$
V	$1-2x^2+2x^4$	$2x-2x^3+x^5$	$\sqrt{1+x^{10}}$	no	yes	$x=2^{\kappa-1}$

Table 1. Polynomial representation of fast rotation methods I through to V.

The formula for $\alpha(x)$ follows from multiplying both numerator and denominator with $1-(-x^2)$, and applying rule (U.1).

3.1. Hyperbolic fast rotations

We define $c_h(x), s_h(x)$ as the hyperbolic approximation pair, and the hyperbolic fast rotation F_h as given by:

$$F_h(x) = \begin{bmatrix} c_h(x) & s_h(x) \\ s_h(x) & c_h(x) \end{bmatrix}, \qquad (41)$$

with the hyperbolic magnification factor $m_h(x)$, and the hyperbolic angle of rotation $\alpha_h(x)$, given by:

$$m_h(x) = \sqrt{c_h{}^2(x) - s_h{}^2(x)} \qquad (42)$$

and

$$\alpha_h(x) = \operatorname{arctanh}(\frac{s_h(x)}{c_h(x)}). \qquad (43)$$

The relationship between the circular and hyperbolic functions are well known. From[11] they are given for the hyperbolic sine and cosine as:

$$\begin{array}{rcl} \cosh(x) & = & \cos(ix) \\ \sinh(x) & = & -i\sin(ix) \end{array} . \qquad (44)$$

where $i^2=-1$.

A similar relationship holds between circular and hyperbolic fast rotations in:

$$\begin{array}{rcl} c_h(x) & = & c(ix) \\ s_h(x) & = & -is(ix) \end{array} . \qquad (45)$$

Due to the nature of the circular polynomial approximation pairs that we have seen so far, with $c(x)$ and $s(x)$ being even and odd functions respectively[1], the relationship (45) results in $c_h(x)$ and $s_h(x)$ having *real* coefficients. Hence the hyperbolic fast rotation F_h is realizable.

[1]Note well that $c(x)$ and $s(x)$ being even and odd functions is not a natural consequence for fast rotations. There exist polynomial approximation pairs that do not fit this pattern, and hence have no realizable hyperbolic counterpart.

Substitution of the relationships of (45) in Equations (42) and (43) leads to the further relationships between the hyperbolic and circular counterparts in:

$$\begin{array}{rcl} m_h(x) & = & \sqrt{c^2(ix)+s^2(ix)} \;=\; m(ix) \\ \alpha_h(x) & = & \operatorname{arctanh}(-i\frac{s(ix)}{c(ix)}) \;=\; -i\alpha(ix) \end{array} . \qquad (46)$$

The relationships (45) and (46) can be applied to obtain the dual, hyperbolic forms of the fast rotation methods presented so far, as well as for most of the fast rotations which are obtained with the techniques described in the next section.

As an example, the hyperbolic fast rotation Method IIIh, given below, is given by application of the above relationships on the circular fast rotation Method III.

$$\begin{array}{rcl} c_h(x) & = & 1+2x^2 \\ s_h(x) & = & 2x+x^3 \\ m_h(x) & = & \sqrt{1-x^6} \\ \alpha_h(x) & = & \operatorname{arctanh}(\frac{2x+x^3}{1+2x^2}) \end{array} . \qquad (47)$$

4. Constructing higher-order fast rotations

Note that, for the maximal fast rotations of Equation (32), the cost of a direct implementation grows with the increase of the precision and the magnitude of the rotation angle, although very slowly. Inspection of Equation (33) shows that the magnification error is an exponential function of the cost of implementation.

In this section we will present construction methods and implementation schemes that perform even better, and where the magnification error is a hyper-exponential function of the cost.

These methods aim at generating new fast rotations by simple construction rules. One of such methods of construction is to take a known fast rotation method, and scale it to attain a higher precision. This scaling can be done either as multiplicative scaling, or extension with special scaling rotations as we shall see in subsections 4.1 and 4.2.

4.1. Multiplicative scaling

Using the multiplicative scaling method, a fast rotation F is expressed as the product:

$$F = F_{\text{base}} \cdot S\,, \tag{48}$$

where F_{base} is the so called *base rotation*, having the same form as (1), but with the base approximation pair $\hat{c}_{\text{base}}, \hat{s}_{\text{base}}$.

This base rotation is scaled to the required precision with the scaling operator given by the matrix S as:

$$S = \begin{bmatrix} \Lambda & 0 \\ 0 & \Lambda \end{bmatrix}, \tag{49}$$

where the *scaling factor* Λ is cheap to implement, basically a short sequence of shift-add operations.

Although it is theoretically possible to take one of the fast rotations of Methods I to V as the base rotation, practical experiments show that convergence to a higher precision is slow, due to the square root in the formula of m.

Instead, Götze[2] uses scaled rotations with a base approximation pair given by:

$$\begin{array}{rcl} \hat{c}_{\text{base}} & = & 1-2^{2\kappa-2} \\ \hat{s}_{\text{base}} & = & 2^{\kappa} \end{array} . \tag{50}$$

In effect, this is the fast rotation Method I of (4) applied twice, and hence the magnification factor m_{base} is **square-root free**. Due to this property, a quadratic converging scaling sequence exists to bring the fast rotation method quickly to the required precision.

Switching to a polynomial representation, substituting x for $2^{\kappa-1}$, we arrive the polynomial base approximation pair:

$$\begin{array}{rcl} c_{\text{base}}(x) & = & 1-x^2 \\ s_{\text{base}}(x) & = & 2x \end{array} , \tag{51}$$

with the magnification factor $m_{\text{base}}(x)$ and the angle of rotation $\alpha_{\text{base}}(x)$ given by:

$$m_{\text{base}}(x) = 1 + x^2\,, \tag{52}$$

and

$$\alpha_{\text{base}}(x) = \arctan(\frac{2x}{1-x^2}) = 2\arctan(x)\,. \tag{53}$$

We define the scaling steps $\lambda_i(x)$ for $i \geq 0$ as:

$$\lambda_i(x) = (1 + x^{(2^i)})\,, \tag{54}$$

and also define the *overall* scaling $\Lambda = \Lambda_L(x)$ as the product of the L scaling steps for $i = 0, 1, \ldots, L-1$:

$$\begin{array}{rcl} \Lambda_L(x) & = & \prod_{i=0}^{L-1} \lambda_i(x) \\ & = & U_{2^L-1}(x) \end{array} . \tag{55}$$

The latter formula follows directly by substitution of (54) in the above and application of rule (U.5).

Multiplicative scaling of the base rotation with the first scaling step $\lambda_0(-x^2) = 1 - x^2$, results in an overall magnification of $(1+x^2)(1-x^2) = (1-x^4)$, which is clearly much closer to unity. Continued scaling results in fast rotations of arbitrary precision.

The complete fast rotation, being the rotation with the base approximation pair of (50), followed by L consecutive scaling steps, is given by:

$$\begin{array}{rcl} c(x) & = c_{\text{base}}(x)\Lambda & = (1-x^2)\Lambda_L(-x^2) \\ s(x) & = s_{\text{base}}(x)\Lambda & = (2x)\Lambda_L(-x^2) \end{array} . \tag{56}$$

The overall magnification factor follows from the product of the scaling factor $\Lambda_L(-x^2)$ and the base magnification factor of Equation (52) via application of rule (U.1) as:

$$m(x) = 1 - (-x^2)^{(2^L)}\,. \tag{57}$$

The overall angle of rotation is the same as that of the base rotation, $\alpha(x) = \alpha_{\text{base}}(x)$, as the scaling operator S does not alter it.

It is without doubt that this is a very powerful method. It is particularly useful in constructing fast rotations of arbitrary precision for large values of the angle exponent. The cost of implementation of the base rotation is two shift-add pairs, that of the scaling steps is L shift-add pairs. The overall cost is hence $L+2$ shift-add pairs, while the magnification error $\varepsilon(x)$, found by inspection of (57), is a *hyper-exponential* function of L.

4.2. Extended rotation

For the extended rotation method, the fast rotation F is expressed as the product:

$$F = F_{\text{base}} \cdot F_{\text{ext}}\,, \tag{58}$$

where F_{base} again a base rotation, as in the previous subsection. This base rotation is followed by an extension rotation F_{ext}, also having the same form as (1), but with the extension approximation pair $\hat{c}_{\text{ext}}, \hat{s}_{\text{ext}}$. The magnification factor m_{ext} is tailored such that it compensates that of the base rotation, m_{base}, resulting in a higher precision fast rotation for F.

We treat the scaling by extended rotation only for the case of maximal fast rotations, although the method is applicable for other types as well.

For maximal fast rotations, the specially tailored extension rotations, parameterized in M, are given by the approximation pair:

$$\begin{array}{rcl} c_{\text{ext}}(x) & = & U_M(-x^2) \\ c_{\text{ext}}(x) & = & xU_{M-1}(-x^2) \end{array} , \tag{59}$$

with the magnification factor and angle of rotation resulting in:

$$m_{\text{ext}}(x) = \sqrt{U_{2M}(-x^2)}, \tag{60}$$

and

$$\alpha_{\text{ext}} = \arctan\left(x\frac{1+(-x^2)^M}{1+(-x^2)^{M+1}}\right). \tag{61}$$

The derivation of the present compact form of the above formulas is left out for the sake of brevity.

Notice the special form of the magnification factor in (60). This rotation is itself not suitable as a fast rotation, but its purpose will be revealed next.

We state that the magnification factor of any maximal fast rotation, so also for m_{base}, allows itself to be written out in the following form:

$$m_{\text{base}}(x) = \sqrt{1-(-u^2)}, \tag{62}$$

and that such a u, expressed in x, can indeed be found.

We take advantage of this fact, constructing the fast rotation as $F(x) = F_{\text{base}}(x) \cdot F_{\text{ext}}(u)$. The overall magnification then follows as the product $m(x) = m_{\text{base}}(x) \cdot m_{\text{ext}}(u)$ which, when substituting (62) and (60) into it, and applying rule (U.1), leads to:

$$\begin{array}{rcl} m(x) & = & \sqrt{1-(-u^2)}\sqrt{U_{2M}(-u^2)} \\ & = & \sqrt{1-(-u^2)^{2M+1}} \end{array}, \tag{63}$$

which shows a $(2M+1)$-fold increase in precision. Note that the resulting fast rotation F is also maximal, hence the method of extension can be applied recursively. The overall angle of rotation is given by:

$$\alpha(x) = \alpha_{\text{base}}(x) + \alpha_{\text{ext}}(u). \tag{64}$$

To illustrate the extension method, we take for the base rotation F_{base} the fast rotation Method III, with:

$$\begin{array}{rcl} c_{\text{base}}(x) & = & 1-2x^2 \\ s_{\text{base}}(x) & = & 2x-x^3 \\ m_{\text{base}}(x) & = & \sqrt{1+x^6} \\ \alpha_{\text{base}}(x) & = & \arctan(\frac{2x-x^3}{1-2x^2}) \end{array}, \tag{65}$$

and for the extension rotation the most simple form of (59), for $M = 1$, with:

$$\begin{array}{rcl} c_{\text{ext}}(x) & = & 1-x^2 \\ c_{\text{ext}}(x) & = & x \\ m_{\text{ext}}(x) & = & U_2(-x^2) = \sqrt{1-x^2+x^4} \\ \alpha_{\text{ext}}(x) & = & \arctan(\frac{x}{1-x^2}) \end{array}. \tag{66}$$

Note that the cost of implementation for this extension is two shift-add operations.

The magnification factor $m_{\text{base}}(x)$ is expressed in the required form of (62), by taking $u = \sigma x^3$, with $\sigma \in \{-1,+1\}$. Note that there are *two* solutions for u, depending on the choice of σ. This parameter σ is equivalent to the direction of rotation for the extension rotation. This method yields *two* new fast rotations F over the (different) angles $\alpha(x) = \alpha_{\text{base}}(x) + \alpha_{\text{ext}}(\sigma x^3)$. The scaling factor is independent of σ, and follows from the derivation:

$$\begin{array}{rcl} m(x) & = & m_{\text{base}}(x) \cdot m_{\text{ext}}(\sigma x^3) \\ & = & \sqrt{1+x^6}\sqrt{U_2(\sigma x^3)} \\ & = & \sqrt{1+x^6}\sqrt{1-x^6+x^{12}} \\ & = & \sqrt{1+x^{18}} \end{array}. \tag{67}$$

The total cost of implementation is for this case only five shift-add pairs, of which three are for the base rotation, and two for the extension. Note that the cost of this new fast rotation is the same as that of Method V, while attaining a far greater accuracy for the same values of the angle exponent.

Recursive application of L levels of the method of extension yields 2^L new, very high accuracy, fast rotations.

In practical cases, only the extension of type (59) are used recursively with $M = 1,2$, and for the initial base rotations Methods II and III. Extension of Method I is not applied in practice, as it leads only to Method III (at the same cost), and a non-canonic version of Method I itself.

5. Applications

Fast rotations have already been applied in a wide range of applications. We will briefly present a few of these below.

5.1. Hemisphere ray tracing

The impetus to the development of the maximal fast rotations has been the problem of high resolution hemisphere sampling in the radiosity algorithm[3, 8]. Here, an artificial 3D environment is sampled by a large number of spherical rays $r_{i,j}$, defined in the local coordinate system of the sampling hemisphere as:

$$r_{i,j} = R(\varphi_i, \theta_j) \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \tag{68}$$

where the 3×3 rotation matrix R is given by two consecutive embedded 2×2 rotations as:

$$R(\varphi,\theta) = \begin{bmatrix} \cos(\varphi) & -\sin(\varphi) & \\ \sin(\varphi) & \cos(\varphi) & \\ & & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos(\theta) & & \sin(\theta) \\ & 1 & \\ -\sin(\theta) & & \cos(\theta) \end{bmatrix}, \tag{69}$$

and where the angles φ_i, θ_j that determine the location of the ray in spherical space, are both *uniformly* distributed with the angular increments $\Delta\varphi, \Delta\theta$ as:

$$\begin{aligned} \varphi_i &= (i+\tfrac{1}{2})\Delta\varphi \\ \theta_j &= (j+\tfrac{1}{2})\Delta\theta \end{aligned} . \tag{70}$$

Since we are dealing with a sampling problem, we are free to choose our spherical sampling "frequencies", the angular increments $\Delta\varphi, \Delta\theta$, in which case we chose them to be one of the fast rotation angles, possibly oversampling the environment slightly. It turned out that the fast rotation Method II was sufficient for the range of the angular sampling frequencies, indexed by the angle exponent κ, that was required by the application.

The most time consuming operation of the hemisphere sampling is the generation of a spherical bundle of rays, and computing the inner product of these with the vertices of a patch, which is necessary in the computation of the intersection points of the rays with that patch.

In[3] we take advantage of the facts that $\Delta\varphi, \Delta\theta$ can be chosen to be fast rotation angles, and that the rays $r_{i,j}$ allow an efficient incremental computation scheme using fast rotations, to reduce the cost of of the most time consuming operation. Both the generation *and* the inner product can be computed with only *one* fast rotation of Method II, requiring only 2 shift-add pairs, and which is performed in only one clock cycle.

The alternative way for computing this would require no less that 11 multiplications and 6 additions for the two embedded 2×2 rotations and the inner product, not even taking into account the computation of the sine and cosine for the angles, which is a problem to itself. As such, through the application of fast rotations, we have achieved a reduction of the arithmetic complexity by more than one order.

A dedicated ASIC[5], capable of computing intersection points using the new technique, has been fabricated. It incorporates a hardware unit that performs Method II fast rotations every clock cycle (at 25 MHz), to 24 bit precision, and over a large domain for the angle exponent.

5.2. Eigenvalue Decomposition (EVD) and related algorithms

In[2, 1] fast rotations are applied to lower the computational complexity of the Eigenvalue decomposition (EVD) of symmetric matrices.

The EVD of a symmetric $n\times n$ matrix A is defined as:

$$\Lambda = QAQ^T , \tag{71}$$

where Q is an orthogonal matrix, and Λ is a diagonal matrix of the eigenvalues. In the iterative cyclic-by-rows Jacobi algorithm for the EVD, this matrix Q is built up out of a regular sequence of embedded 2×2 rotations $q_{s,i,j}$, of which the angles of rotation are chosen such that the matrix A is diagonalised to Λ in each step.

$$Q = \prod_{s=1}^{S} \prod_{i=1,j=i}^{n,n} q_{s,i,j} , \tag{72}$$

where S is the number of sweeps for the algorithm to reach convergence.

In[1], a set of approximating angles is built up, for $N_{\text{mant}} = 32$-bit precision, with the maximal fast rotation Methods I, II, and III, and with the multiplicative scaling method for those angles for which the angle exponent κ is larger than the upper bound for Method III in (23). Taking advantage of the fact that the EVD is an *iterative* algorithm, that converges to the solution Λ, we approximate the exact rotations $q_{s,i,j}$ with the approximate $\tilde{q}_{s,i,j}$ which is the embedding of the fast rotation from the set which is the closest, in terms of the angle of rotation, to the exact rotation. Determining which fast rotation is the closest is done on-line.

Even though the convergence, expressed in the number of sweeps S, is slower, the new method *does* converge. However, when comparing the actual cost, measured in shift-add pairs, as compared to the cost of exact rotations using Cordic, the new method outperforms the other by a factor of more than 7, which is an improvement of almost one order.

The same idea can be applied to singular value decomposition (SVD), the QR decomposition, and the adaptive versions thereof as well.

5.3. Orthogonal FIR filter banks

In the above application for the EVD, we have seen that an orthogonal matrix Q can be represented as a regular sequence of embedded fast rotations. In[4, 7] we have developed a *greedy* algorithm that approximates an orthogonal matrix Q by Q_T, implemented as a sequence of T embedded fast rotations q_t, keeping the total cost as low as possible.

$$Q_T = \prod_{t=1}^{T} q_t , \tag{73}$$

where the embedded fast rotation q_t works on the indices $(i,j)_t$, with the angle of rotation α_t.

The greedy algorithm chooses the q_t at each step such that Q_T converges as quick as possible to the required Q, and can be seen as a successive approximation of it. The network of fast rotations that implements Q_T is hence often irregular. Note that Q_T by itself is orthogonal at all times, and can be used as an orthogonal approximation of Q.

In[7, 6] we have applied such approximations for the efficient orthogonal implementation of large image transforms, such as the lapped orthogonal transform (LOT). The LOT is decomposed recursively into a network of orthogonal matrices of decreasing size and butterfly operations. Applying the above approximation to the orthogonal matrices leads to a low-cost, but very robust, implementation of the LOT. In[6], we report an implementation for a 16×32 LOT (512 elements, full matrix) requiring only 776 shift-add pairs. A direct implementation, with less desirable numerical properties, would require 512 multiplications, so again we achieve an improvement of almost one order. The same technique has been applied to other image transforms, such as the DCT, with promising results. A prototype chip[7] harbouring a number processors with a 4 stage fast rotation pipeline, is fabricated to prove the advantages over multiplier-based DCT and LOT implementations.

6. Conclusions

We have presented a formal definition and classification of fast rotations, as well as methods of construction to obtain them.

We have shown a number of applications in which fast rotations have been successfully applied, in some cases leading to a reduction of the computational complexity by more than one order, as compared to traditional methods. In fact, already two ASICs[5, 7], both relying heavily on the application of these fast rotations, have been fabricated.

References

[1] J. Götze and G. Hekstra. An algorithm and architecture based on orthonormal μ-rotations for computing the symmetric EVD. *Integration, the VLSI Journal*, 20:21–39, 1995.

[2] J. Götze, S. Paul, and M. Sauer. An efficient Jacobi-like algorithm for parallel eigenvalue computation. *IEEE Trans. Comput.*, 42:1058–1065, 1993.

[3] G. Hekstra. Definition and structure of hemisphere and viewing rays. Technical Report ET/NT/Radio-9, TU Delft, November 1993.

[4] G. Hekstra. Fast matrix multiplication using orthogonal rotations. Technical Report ET/NT/Fact-10, TU Delft, February 1995.

[5] G. J. Hekstra and E. F. Deprettere. A chip set for a ray-casting engine. In *Proceedings VLSI Signal Processing*, volume IX, pages 211–220, San Francisco, US, October 1996.

[6] G. J. Hekstra, E. F. Deprettere, R. Heusdens, and M. Monari. Recursive approximate realization of image transforms with orthonormal rotations. In *Proceedings International Workshop on Image and Signal Processing*, Manchester, UK, November 1996.

[7] G. J. Hekstra, E. F. Deprettere, R. Heusdens, and Z. Zeng. Efficient orthogonal realization of image transforms. In *Proceedings SPIE*, Denver, Colorado, US, August 1996.

[8] L.-S. Shen. *A Parallel Image Rendering Algorithm and Architecture based on Ray Tracing and Radiosity Shading*. PhD thesis, Delft University of Technology, September 1993. ISBN 90-5326-012-9/CIP.

[9] J. E. Volder. The CORDIC trigonometric computing technique. *IRE Transactions on electronic computers*, pages 330–334, Sept. 1959.

[10] J. Walther. A unified algorithm for elementary functions. *proceedings of the AFIPS Spring Joint Computer Conference*, pages 379–385, 1971.

[11] D. Zwillinger, editor. *Standard Mathematical Tables and Formulae*. CRC press, 30th edition, 1989.