N-Dimensional Approximation of Euclidean Distance

Cardarilli, Gian Carlo; Di Nunzio, Luca; Fazzolari, Rocco; Nannarelli, Alberto; Re, Marco; Spano, Sergio

Published in:
IEEE Transactions on Circuits and Systems II: Express Briefs

Link to article, DOI:
10.1109/TCSII.2019.2919545

Publication date:
2020

Document Version
Peer reviewed version

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSII.2019.2919545, IEEE Transactions on Circuits and Systems II: Express Briefs

N-Dimensional Approximation of Euclidean Distance

Gian Carlo Cardarilli *Member, IEEE*, Luca Di Nunzio, Rocco Fazzolari, Alberto Nannarelli, *Senior Member, IEEE*, Marco Re, *Member, IEEE,* and Sergio Spanò

***Abstract*—Several applications in different engineering areas require the computation of the Euclidean distance, a quite complex operation based on squaring and square root. In some applications, the Euclidean distance can be replaced by the Manhattan distance. However, the approximation error introduced by the Manhattan distance may be rather large, especially in a multi-dimensional space, and may compromise the overall performance. In this paper, we propose an extension of the αMax+βMin method to approximate the Euclidean distance to a multi-dimensional space. Such a method results in a much smaller approximation error with respect to the Manhattan approximation at expenses of a reasonable increase in hardware cost. Moreover, with respect to the Euclidean distance, the αMax+βMin method provides a significant reduction in the hardware if the application can tolerate some errors.**

***Index Terms*—Euclidean distance approximation.**

I. Introduction

The evaluation of the Euclidean distance is needed in many different fields of engineering, such as machine learning, communications, bioinformatics, etc.. The computation of the Euclidean distance requires squaring and square root, which are expensive operators in hardware [1], [2]. For this reason, alternative distances have been introduced in the literature for different applications. Among these alternatives, the most popular is the Manhattan distance.

The performance degradation when replacing the Euclidean with the Manhattan distance depends on the application, and/or the data set.

For applications in machine learning, such as K-means clustering algorithms, the work done in [3], [4] and [5] show that the Euclidean distance provides more accurate results for these algorithms when compared with the Manhattan distance.

Moreover, in [6], it is shown that, by using the Euclidean distance in facial expressions classification, a better accuracy can be obtained with respect to Manhattan distance. In [7], the authors show that the use of Euclidean metric provides more accurate results in software faults prediction. In all the applications that are sensitive to the error introduced by the Manhattan distance approximation, it is necessary to use a method that reduces the approximation error by avoiding the complex circuitry of the Euclidean distance computation.

In this paper, we propose an extension of the αMax+βMin method to approximate the Euclidean distance from two-dimensions to a multi-dimensional space. This method provides better accuracy than the Manhattan distance especially when the dimensionality increases. We design two alternatives for the implementation of the two-dimensions (2D) αMax+βMin approximation, and we provide a trade-off analysis for error, latency, area and power dissipation. Moreover, based on the 2D αMax+βMin unit, we characterize the approximation error for a multi-dimensional space, and propose a tree-based multi-D hardware implementation.

G.C. Cardarilli, L. Di Nunzio, R. Fazzolari, M. Re and S. Spanò are with the Dept. of Electronic Engineering, Univ. of Roma “Tor Vergata”, Italy.

A. Nannarelli is with the Dept. of Applied Mathematics an d Computer Science (DTU Compute), Technical Univ. of Denmark, Lyngby, Denmark. E-mail: alna@dtu.dk

The results show that the distance approximation based on the multi-dimensional αMax+βMin method provides a significantly smaller error than the Manhattan distance and a much lower hardware cost than the Euclidean distance.

The paper is organized as follows. In Sec. II the αMax+βMin approximation in two-dimensions is explained and its hardware implementation is described. In Sec. III, the αMax+βMin approximation is extended to the multi-dimensional space, and error analysis and hardware architectures are presented. Conclusions are drawn in Sec. IV.

II. The αMax+βMin Approximation in Two-Dimensions

The Euclidean distance (two-dimensional) is defined as

$$z = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

where (x_1, y_1) and (x_2, y_2) are the coordinates of the two points.

The αMax+βMin approximation is defined as

$$z_A = \alpha \cdot \text{Max} + \beta \cdot \text{Min}$$

where Max and Min are the maximum and the minimum, respectively, between $|x_2 - x_1|$ and $|y_2 - y_1|$, and α and β are two constants which minimize the approximation error [8]. The smallest error is obtained for

$$\alpha_0 = \frac{2\cos\frac{\pi}{8}}{1+\cos\frac{\pi}{8}} = 0.9604\ldots, \quad \beta_0 = \frac{2\sin\frac{\pi}{8}}{1+\cos\frac{\pi}{8}} = 0.3978\ldots$$

α_0 and β_0 can be approximated by fractions of simple multiples of powers of two. Table I shows some values from [8].

TABLE I
Values of α and β and approximation errors.

		error (%)	
α	β	largest	average
1/1	1/2	11.80	8.68
1/1	1/4	11.61	3.20
1/1	3/8	6.80	4.25
15/16	15/32	6.25	3.08

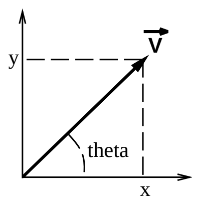

Fig. 1. Unit vector.

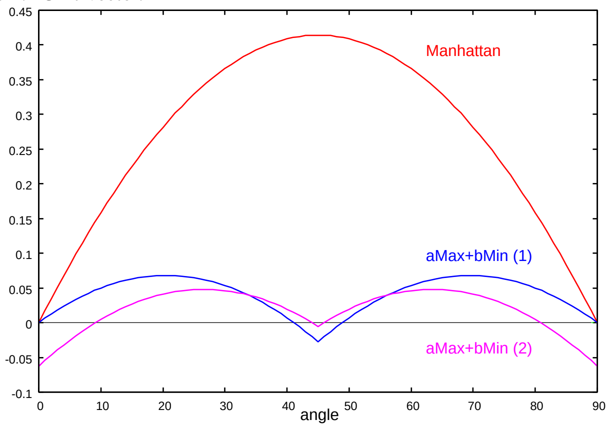

Fig. 2. Relative error for αMax+βMin and Manhattan distance approximation (double precision).

To characterize the approximation error, we resort to the unit vector (Fig. 1). The magnitude of the vector is $|z| = 1$ and the coordinates of the two points are $(0, 0)$ and (x, y). By rotating the vector, we evaluate the αMax+βMin distance for different angles from $\theta = 0$ to $\theta = \pi/2$.

We also compare the αMax+βMin method with the Manhattan distance. Since in the first quadrant both x and y are positive, the Manhattan distance is

$$z_M = x + y \; .$$

For the αMax+βMin method, we choose two sets of (α, β): $\alpha_1 = 1$, $\beta_1 = 3/8$ and $\alpha_2 = \frac{15}{16}$, $\beta_2 = \frac{15}{32}$; because, according to Table I, these pairs of values give the best trade-off error vs. simple multiples. Therefore, for the αMax+βMin method we have

$$z_{A1} = Max + \frac{3}{8}Min \qquad (1)$$

$$z_{A2} = \frac{15}{16}Max + \frac{15}{32}Min \qquad (2)$$

where Max=x for $\theta \leq \pi/4$ and Max=y for $\theta > \pi/4$.

The relative error for $0 \leq \theta \leq \pi/2 = 90°$ is shown in Fig. 2 for double-precision simulations. For the Manhattan method the largest error is $\epsilon_M = 0.41$ and the average error is $\bar{\epsilon}_M = 0.30$. For the αMax+βMin method, the errors are significantly smaller:

1) maximum $\epsilon_{A1} = 0.0680$ and average $\bar{\epsilon}_{A1} = 0.047$;
2) maximum $\epsilon_{A2} = 0.0625$ and average $\bar{\epsilon}_{A2} = 0.035$.

Clearly, the approximation provided by the αMax+βMin method is superior.

A. αMax+βMin Hardware Implementation for 2D

In this section, we evaluate the performance of the approximation unit for the two sets of values (α_1, β_1)=(1, $\frac{3}{8}$) and (α_2, β_2)=($\frac{15}{16}$, $\frac{15}{32}$).

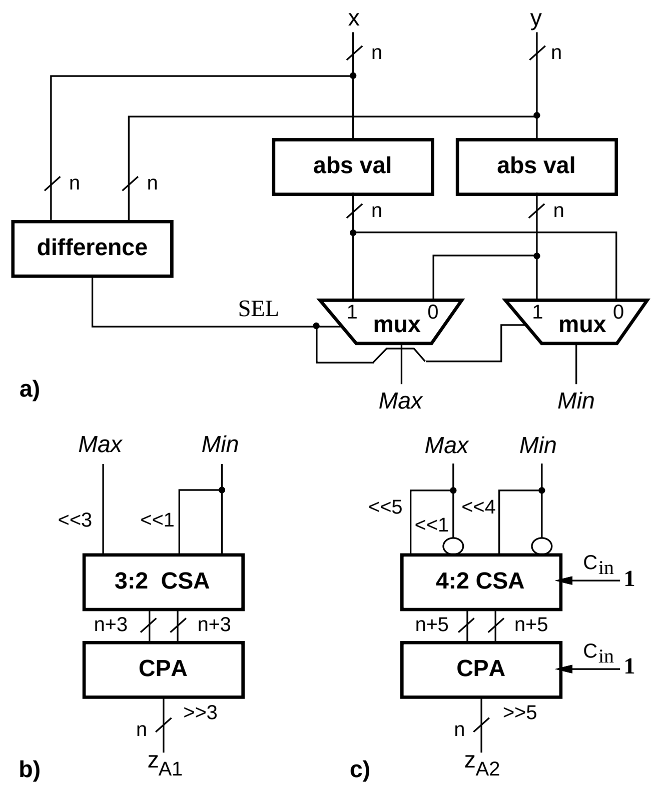

Fig. 3. Units for αMax+βMin approximation. a) Common part; b) α_1, β_1; c) α_2, β_2.

For the implementation of the units we opted for a 45 nm CMOS library of standard cells, the synthesis is based on Synopsys tools. The FO4 delay[1] for this low power library is 64 ps and the area of the NAND-2 gate is 1.06 μm^2.

The units are implemented in fixed-point with the following bit-widths: 32, 24, 16 and 8. We assume that the inputs to the units are $x = x_2 - x_1$ and $y = y_2 - y_1$.

The first stage of the αMax+βMin unit, depicted in Fig. 3 a), consists of the following blocks:

Block **difference** determines the largest input. It performs the difference $x - y$ if the numbers have same sign, or the sum if the signs are different. The Max is selected according to the outcome of the addition/subtraction and the sign of the operands (bit SEL).

In parallel with the Max/Min computation, the two blocks **abs val** process x and y to determine their absolute value: sign checking followed by two's complementation when negative.

The selection of Max and Min is performed by the two multiplexers **mux**.

The second stage of the unit, Fig. 3 b) and c), differs depending on the (α, β) values.

For the (α_1, β_1) values, the approximation is done by a Carry-Save Adder (**3:2 CSA**) followed by a Carry-Propagate Adder (**CPA**) to compute

$$Max + \frac{3}{8}Min = \frac{1}{8}(8Max + 2Min + Min)$$

In this way, we decompose the product in wired-shifts and sums. Since we need to add three addends, we use the CSA

[1]A 1 FO4 delay is the delay of an inverter of minimum size with a load of four minimum sized inverters.

TABLE II
RESULTS OF IMPLEMENTATIONS FOR αMAX+βMIN AND MANHATTAN APPROXIMATION.

	αMax+βMin (1)			αMax+βMin (2)			Manhattan unit		
n.bit	delay	Area	P_{ave}	delay	Area	P_{ave}	delay	Area	P_{ave}
	$[ps]$	$[\mu m^2]$	$[\mu W]$	$[ps]$	$[\mu m^2]$	$[\mu W]$	$[ps]$	$[\mu m^2]$	$[\mu W]$
32	1,084	2,924	89	1,112	4,100	136	572	2,180	45
24	972	2,285	68	1,075	3.646	126	502	1,610	34
16	873	1,419	41	997	2,131	70	456	1,140	25
8	676	683	18	805	974	28	388	510	11

P_{ave} measured at 100 MHz

to reduce the operands from 3 to 2, and then we perform a carry-propagate addition in the CPA. At the end of the approximation, the result is shifted of 3 positions to the right (division by 8).

For the (α_2, β_2) values, the approximation is:

$$\frac{15}{16}Max + \frac{15}{32}Min = \frac{1}{32}(32Max - 2Max + 16Min - Min)$$

The multiplications by constants 15 and 15×2 are implemented by the subtractions $16X - X$ and $32X - 2X$, respectively. In this case, we need a 4:2 CSA to sum up the four terms. The complementation is done by inverting the bits (one's complement) and by adding "1" in the least-significant position of the sum in CSA and CPA, as illustrated in Fig. 3 c).

For both variants of the αMax+βMin unit the critical path is through the blocks **difference** – **mux** – **CSA** – **CPA**, and it is roughly the delay of two n-bit adders.

The implementation results are reported in Table II.

By comparing the αMax+βMin alternatives (1) and (2), Type 2 is slightly slower because of the extra XOR delay in the CSA (4:2 vs. 3:2) and the longer carry chain due to the 5 positions shifting. Moreover, in Type 2 both area and power are about 50% larger than Type 1, because of the extra gates in the CSA 4:2 and in the CPA.

Table II also reports the results of the implementation of a Manhattan approximation unit.

The Manhattan distance approximation unit consists of two **abs val** blocks followed by a **CPA**. However, the **abs val** block is reduced to a conditional bit-complementer (when the number is negative) and the two's complement carry is added in the CPA. If both numbers are negative, one of the two carries is pre-added in a n-bit array of half-adders. In this way, we avoid to propagate the carry when producing the absolute value.

This method of transforming the carry necessary for two's complementation, can also be applied to the αMax+βMin unit. However, since the latency of block **abs val** is hidden by the latency of block **difference** in Fig. 3 a), we prefer to compute the absolute values and avoid transferring the two's complement carries to simplify the operations in the CSA.

The delay of αMax+βMin (1) is about double that of the Manhattan unit. By interpolating the delay values obtained from the hardware implementation (Table II) for a generic number of bits, we obtain:

$$\begin{aligned} t_{A1}(n) &= 20 + 2 \cdot 100 \log_2 n \qquad (\alpha\text{Max+}\beta\text{Min (1)}) \\ t_M(n) &= 100 + 90 \log_2 n \qquad (\text{Manhattan}) \end{aligned}$$

TABLE III
RESULTS OF IMPLEMENTATIONS FOR n-BIT FIXED-POINT MULTIPLIER.

	Squarer/multiplier		
n.bit	delay	Area	P_{ave}
	[ps]	$[\mu m^2]$	$[\mu W]$
32	3,210	16,020	406
24	2,440	8,900	222
16	1,710	3,620	80
8	891	850	20

P_{ave} measured at 100 MHz

This result shows clearly that the two adders (**difference** and **CPA** in Fig. 3) determine the critical path of the αMax+βMin unit.

Also the area and the power dissipation of the αMax+βMin units are larger than in the Manhattan unit. However, the smaller error in αMax+βMin may allow, for some applications, to converge more rapidly and save both execution time and energy.

As for the computation of the Euclidean distance $\sqrt{x^2 + y^2}$, it requires squaring (multiplication), addition and square root. Especially square root is a complicated operation in hardware, There are several alternatives for its implementation, but all have long latency or large area [9].

To give an idea of the costs for the Euclidean distance computation, we report in Table III only the implementation results for a squarer/multiplier for 32, 24, 16 and 8 bits operands. Square root is normally a multi-cycle operation for precisions above 8 bits, and fast implementation methods normally require one or more multipliers [9].

Table III show that latency, area and power dissipation make the computation of the Euclidean distance very expensive if we need more accuracy than the one provided by the αMax+βMin approximation.

III. EXTENSION TO N DIMENSIONAL EUCLIDEAN DISTANCE

In this section, we extend the αMax+βMin approximation to a N-dimensional hyperspace. The 2D αMax+βMin approximation is based on determining the maximum (and the minimum) between the length of the two segments which is a binary operation. We refer to the approximation (1) as *amax* in the following.

To extend the method to N dimensions, we can apply the approximation iteratively N-1 times. For example, for 3-D we need to apply (1) twice:

$$approx.dist._{3D} = amax(amax(x, y), z)$$

Clearly, since *Max* and *Min* are not linear operators, the approximation *amax* is not associative for $D > 2$.

Since *amax* is a binary operator, the dimensions can be arranged in a binary tree. For example for 4-D:

$$approx.dist._{4D} = amax(amax(x, y), amax(z, w))$$

which requires three *amax* units, but has depth of two levels and reduced latency with respect to the iterative implementation of the approximation.

Next, we determine the maximum and average error for the multidimensional αMax+βMin approximation. We compute

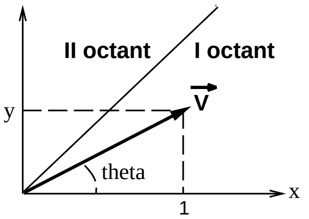

Fig. 4. 2D vector for maximum αMax+βMin error computation.

the maximum error analytically for 2D and 3D, and, by simulations for dimensions up to 10-D.

A. Maximum relative error in 2D

To compute the maximum relative error in closed form, we resort to the geometric representation in Fig. 4. We assume a vector of components $x = 1$ and $y = \tan\theta$ between the points $(0,0)$ and $(1,y)$ for $\theta \in [0, \frac{\pi}{4}]$. In the I octant $0 \leq y \leq x = 1$. In the II octant $\theta \in [\frac{\pi}{4}, \frac{\pi}{2}]$, x and y are swapped: $y = 1$ and $x = \tan(\frac{\pi}{2} - \theta)$ with $0 \leq x \leq y = 1$. We refer to the I octant $\theta \in [0, \frac{\pi}{4}]$ in the following.
The Euclidean distance for the module of the vector in Fig. 4 is

$$d_{2D} = \sqrt{1+y^2} \tag{3}$$

Since $x \geq y$, $Max = x = 1$ and $min = y$, for the αMax+βMin approximation we have

$$d_{a2} = 1 + \beta y \tag{4}$$

By combining (3) and (4), we obtain the relative error

$$\epsilon_{2D} = \frac{(1+\beta y) - \sqrt{1+y^2}}{\sqrt{1+y^2}} = \frac{1+\beta y}{\sqrt{1+y^2}} - 1 \tag{5}$$

The maximum relative error can be determined by taking the first derivative of (5) to find the maximum:

$$\frac{d\epsilon_{2D}}{dy} = \frac{\beta - y}{(1+y^2)^{\frac{3}{2}}} = 0 \quad \Rightarrow \quad y = \beta$$

Therefore:

$$\epsilon_{MAX\ 2D} = \frac{1+\beta^2}{\sqrt{1+\beta^2}} - 1 = \sqrt{1+\beta^2} - 1 = \frac{\sqrt{73}}{8} - 1 = 0.068000468 \tag{6}$$

The case can be extended to any x and $y = x\tan\theta$ with $\theta \in [0, \frac{\pi}{4}]$.

B. Maximum Relative Error in 3D

For the three-dimensional space, we extend the case of Fig. 4 to the third coordinate z with $x = 1$, $y = \tan\theta$ and $z = \tan\phi$ for $\theta \in [0, \frac{\pi}{4}]$ and $\phi \in [0, \frac{\pi}{4}]$. As a result, $0 \leq y \leq x = 1$ and $0 \leq z \leq x = 1$.

The Euclidean distance is

$$d_{3D} = \sqrt{1+y^2+z^2} \tag{7}$$

For the 3D case, we have to apply the binary αMax+βMin approximation twice. Since $y \leq x$ and $z \leq x$, $Max = x = 1$. Moreover, the result of applying the first αMax+βMin approximation to x will result in the maximum for the second approximation:

$$d_{a3} = (1+\beta y) + \beta z \tag{8}$$

By combining (7) and (8), we obtain the relative error

$$\epsilon_{3D} = \frac{1+\beta y+\beta z}{\sqrt{1+y^2+z^2}} - 1 \tag{9}$$

By calculating the two partial derivatives from (9), we obtain

$$\begin{cases} \frac{\partial\epsilon_{3D}}{\partial y} = \frac{\beta(-yz+z^2+1)-y}{(1+y^2+z^2)^{\frac{3}{2}}} = 0 \\ \frac{\partial\epsilon_{3D}}{\partial z} = \frac{\beta(-yz+y^2+1)-z}{(1+y^2+z^2)^{\frac{3}{2}}} = 0 \end{cases} \tag{10}$$

which is a second degree symmetric system. By simulation of random values for $y \leq x$ and $z \leq x$, we determine the maximum relative error occurring when $y = z$. Consequently, for (10), we obtain:

$$y = \beta \quad \text{and} \quad z = \beta$$

By substituting y and z in (9), we obtain

$$\epsilon_{MAX\ 3D} = \frac{1+2\beta^2}{\sqrt{1+2\beta^2}} - 1 = \sqrt{1+2\beta^2} - 1 = \frac{\sqrt{82}}{8} - 1 = 0.131923142 \tag{11}$$

Therefore, the maximum relative error for the αMax+βMin method in 3D is about 13%, obtained for the points between $(0,0,0)$ and $(x, \beta x, \beta x)$.

C. Maximum Relative Error in m-D

For multi-dimension m-D, when $m > 3$, we opted for simulation of random vectors to empirically find the maximum error for the αMax+βMin approximation. Table IV reports the maximum and average errors from $m = 2$ to $m = 10$. The error values are also plotted in Fig. 5. The maximum relative error for $m \in [2, 10]$ are obtained in the vector starting in the origin and with ending coordinates in $\vec{V}$.

Table IV also reports the average error for multi-dimensional Manhattan distance (rightmost column). By comparing ϵ_{ave} for the two approximation methods, the Manhattan distance error is about one order of magnitude higher than the αMax+βMin (1) approximation. For example, for 6D (m=6) the average error is about 10% for αMax+βMin and about 120% for Manhattan with respect to the Euclidean distance.

TABLE IV
ERRORS FOR MULTI-D αMAX+βMIN AND MANHATTAN APPROXIMATION FOR $m \in [2, 10]$.

	αMax+βMin (1)			Manhattan
m	ϵ_{MAX}	$\vec{V}$	ϵ_{ave}	ϵ_{ave}
2	0.068	$(1,\beta)$	0.046	0.320
3	0.132	$(1,\beta,\beta)$	0.072	0.571
4	0.141	$(1,\beta,\beta,\beta^2)$	0.076	0.784
5	0.201	$(1,\beta,\beta,\beta,\beta^2)$	0.107	0.982
6	0.209	$(1,\beta,\beta,\beta,\beta^2,\beta^2)$	0.100	1.159
7	0.217	$(1,\beta,\beta,\beta,\beta^2,\beta^2,\beta^2)$	0.094	1.324
8	0.218	$(1,\beta,\beta,\beta,\beta^2,\beta^2,\beta^2,\beta^4)$	0.090	1.480
9	0.274	$(1,\beta,\beta,\beta,\beta,\beta^2,\beta^2,\beta^2,\beta^4)$	0.132	1.627
10	0.281	$(1,\beta,\beta,\beta,\beta,\beta^2,\beta^2,\beta^2,\beta^2,\beta^4)$	0.133	1.766

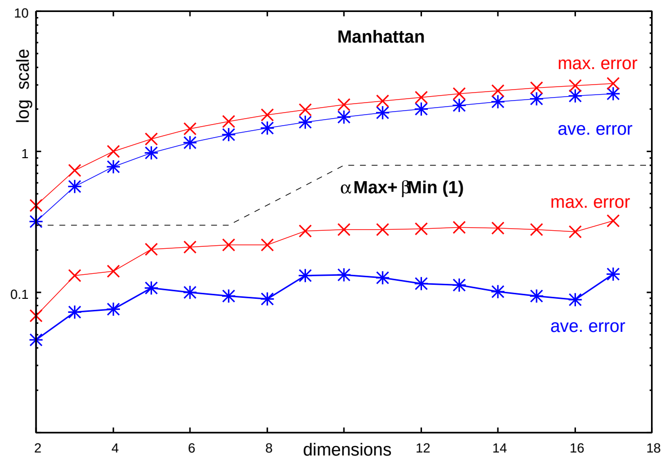

Fig. 5. Plots of maximum and average relative errors for Manhattan and αMax+βMin distance for $m \in [2, 17]$ (logarithmic scale on y-axis).

We illustrate in Fig. 5 the trends of the errors for $m \in [2, 17]$. For the Manhattan distance (top plots) the errors grow linearly (the scale is logarithmic in the figure). As for αMax+βMin distance, the maximum error increases by about 5% (Table IV) when the tree depth increases $m = \{3, 5, 9, 17\}$, and the trends are rather flat until an extra level is added.

D. Hardware Implementation for Multi-Dimensional αMax+βMin Approximation

The hardware implementation for a multi-dimensional αMax+βMin (1) unit, is obtained by building a binary tree of 2D units. Fig. 6 shows the implementation of 3D, 4D, and 5D αMax+βMin (1) approximation units. Only the inputs at the first level of the tree require the block to compute the absolute value (marked with a square in the figure).

Table V reports the implementation results of the multi-dimensional αMax+βMin (1) and Manhattan approximation units for the 24-bit case.

The latency of the multi-dimensional αMax+βMin units grows about 1.0 ns per tree-level. The units can be pipelined by placing registers after each level to reach a throughput of 1 GOPS (10^9 operations per second).

If the error by the multi-dimensional Manhattan approximation is tolerable, its hardware cost is clearly much smaller: absolute values for each dimension, followed by an adder tree and a final CPA.

In contrast, the cost of the implementation of the Euclidean distance is much higher than the multi-dimensional αMax+βMin approximation. A squarer, or multiplier, is required for each dimension resulting in either long latency (multiplications executed sequentially), or large area (one multiplier for each dimension). Moreover, the square root is even more expensive than squaring. If we opt for an iterative implementation (Newton-Raphson or SRT) its latency is several clock cycles, while for a faster solution (e.g., polynomial approximation) a few parallel multipliers are needed.

IV. Conclusions

In this paper, we present the extension of the αMax+βMin method for the approximation of the Euclidean distance from two to multi-dimensions. The approximation error in αMax+βMin is much smaller than the error obtained by Manhattan distance, and the hardware cost (delay, area, power dissipation) of the αMax+βMin units is much lower than the implementation of the Euclidean distance by squarers and square root. Consequently, the αMax+βMin method is a viable solution when the approximation error by the Manhattan distance is not acceptable for the application, and the hardware cost of the exact Euclidean distance computation is too high.

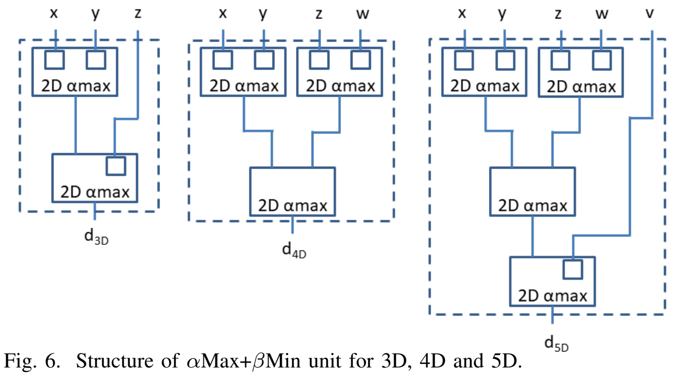

Fig. 6. Structure of αMax+βMin unit for 3D, 4D and 5D.

TABLE V
Results of implementations for 24-bit multi-dimensional αMax+βMin (1) and Manhattan approximation units.

dimensions	αMax+βMin (1)			Manhattan		
	delay [ps]	Area [μm^2]	P_{ave} [μW]	delay [ps]	Area [μm^2]	P_{ave} [μW]
3	1,940	3,780	125	590	1,830	55
4	2,070	5,320	187	700	2,290	80
5	3,040	7,050	260	750	2,730	100

P_{ave} at 100 MHz

References

[1] S. Kuang, J. Wang, and C. Guo, “Modified Booth Multipliers With a Regular Partial Product Array,” *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 56, no. 5, pp. 404–408, May 2009.
[2] E. Antelo, P. Montuschi, and A. Nannarelli, “Improved 64-bit Radix-16 Booth Multiplier Based on Partial Product Array Height Reduction,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 64, no. 2, pp. 409–418, Feb. 2017.
[3] A. Singh, A. Yadav, and A. Rana, “K-means with Three different Distance Metrics,” *International Journal of Computer Applications*, vol. 67, no. 10, pp. 13–17, 2013.
[4] S. Kapil and M. Chawla, “Performance Evaluation of K-means Clustering Algorithm with Various Distance Metrics,” in *1st IEEE International Conference on Power Electronics, Intelligent Control and Energy Systems (ICPEICES-2016)*, 2016, pp. 1–4.
[5] M. Kutyowska, “K-Nearest Neighbours Method as a Tool for Failure Rate Prediction ,” *Periodica Polytechnica Civil Engineering*, vol. 62, no. 2, pp. 318–322, 2018.
[6] L. Greche1, M. Jazouli, N. Es-Sbai, A. Majda, and A. Zarghili, “Comparison Between Euclidean and Manhattan Distance Measure for Facial Expressions Classification,” in *2017 IEEE International Conference on Wireless Technologies Embedded and Intelligent Systems (WITS)*, 2017, pp. 1–4.
[7] D. Kaur, “A Comparative Study of Various Distance Measures for Software Fault Prediction,” *International Journal of Computer Trends and Technology*, vol. 17, no. 3, pp. 117–120, 2014.
[8] R. G. Lyons, *Understanding Digital Signal Processing*, 2nd ed. Prentice Hall, 2004.
[9] M. Ercegovac and T. Lang, *Digital Arithmetic*. Morgan Kaufmann Publishers, 2004.