HARDWARE ARCHITECTURE FOR REAL-TIME DISTANCE TRANSFORM

Jarmo H. Takala, Jouko O. Viitanen, Jukka P.P. Saarinen*

Tampere University of Technology
Signal Processing Laboratory
P.O.B. 553, 33101 Tampere, Finland

*VTT Automation
Machine Automation Laboratory
P.O.B. 13021, 33101 Tampere, Finland

ABSTRACT

A distance transform (DT) converts a binary image consisting of foreground (feature) and background (non-feature) pixels into a gray level image where each pixel contains the distance from the corresponding pixel to the nearest foreground pixel. The computation of the exact Euclidean DT is computationally complex task and, therefore, approximations are typically utilized. In this paper, an area-efficient architecture for computing a DT approximation is presented. The architecture utilizes order-based encoded distance representation allowing simple bitwise operations to be used for determining the distance to the nearest foreground pixel in the constrained neighborhood. Tabulated distance values are used thus cumulative errors are avoided. Due to the simple operations real-time operation can be expected.

1. INTRODUCTION

A distance transform (DT) converts a binary image consisting of foreground (feature) and background (non-feature) pixels into a gray level image where each pixel contains the distance from the corresponding pixel to the nearest foreground pixel. The DT's have been used in several applications including morphological processing, object detection [1], model-based image coding [2], printed circuit board inspection [3], collision avoidance and path finding in robotics [4], and data extraction from form documents [5].

Let us assume a binary $N \times N$ image $A = \{a(i,j)\}$ where $i,j = 0,\ldots,N-1$. The coordinates of foreground pixels are collected into a set $F = \{(m,n) \mid a(m,n) = 1\}$. The Euclidean Distance Transform (EDT) of an pixel $a(i,j)$ is calculated by

$$d^E(i,j) = \min_{(m,n)\in F} \sqrt{(i-m)^2 + (j-n)^2} \qquad (1)$$

The calculation of the exact EDT is essentially a global operation and, therefore, computationally complex. Due to this fact, the EDT is often approximated by propagating local distances. One example of this approach is the 3-4 chamfer DT [6] where the distance between 4-neighbours is approximated by 3 and diagonal neighbours by 4. Typical sequential chamfer DT algorithm requires two complete scans over the image data; the forward scan can be performed on-the-fly, but the backward scan can not be initiated until the forward scan has been completed. This requires expensive frame buffer for intermediate storage which implies also increased latency. The pipelining is also difficult due to the data dependencies. A hardware architecture of this kind is reported in [7] and [8]. The parallel chamfer DT algorithms are iterative, where the processing is repeated until no changes are performed. This requires construction of an parallel processor array [9] which is typically complex and expensive. In addition, parallel arrays still require storage for the complete image.

Another approarch is the Unified Distance Transform (UDT) algorithm [3] which uses independent row and column scans with tabulated distance values for avoiding the error accumulation due to the distance propagation. However, four scans are needed, and its total execution time is typically long compared to chamfer DT's. In addition, the UDT contains data dependent processing.

The previously suggested algorithms have mostly been realized on programmable architectures which typically fail to meet either the real-time constraints or cost requirements. In order to obtain cost effective realizations, e.g. for industrial machine vision applications, we have to find simpler solutions for approximated DT calculation. Such an implementation can be achieved by utilizing the z-clipped DT approach defined in [10] as

$$d(i,j \mid z) = \min\left(d(i,j), z\right), z > 0 \qquad (2)$$

In this paper, we present an application specific hardware architecture for computing z-clipped DT based on the parallel algorithm reported earlier in [11] and [12]. In Section 2, the clipped DT algorithm is shortly described. In Section 3, the algorithm is applied to sequential image data streams and the area-efficient DT architecture is presented. Finally, Section 4 summarizes the paper.

2. CLIPPED DISTANCE TRANSFORM

The z-clipped DT in (2) implies that, in order to define the distances to foreground pixels, we can perform a local search in constrained neighborhood of a pixel instead of the exhaustive global search found in (1). The size of the search area (see Fig. 1) is defined by the clipping distance z and the set of coordinates in the distance mask S_z is defined as

$$S_z = \left\{(i,j) \mid \sqrt{i^2+j^2} < z\right\} \qquad (3)$$

We can also utilize the fact that the distance to a foreground pixel at a certain location in the neighborhood is completely defined by the location of the pixel in the distance mask S_z. The corresponding distance metric can be retrieved from a table and, therefore, no arithmetic operations are needed. We use order-based encoded distance vectors for representing the distance values; assume an array R_z containing all the s_z different discrete distance values existing in the distance mask S_z

$$R_z = \Big\{(r_0, r_1, \ldots, r_{s_z-1}) \mid r_k = \sqrt{i^2+j^2} : \qquad (4)$$
$$k = 0,\ldots,s_z-1, (i,j) \in S_z, r_0 < r_1 < \ldots < r_{s_z-1}\Big\}$$

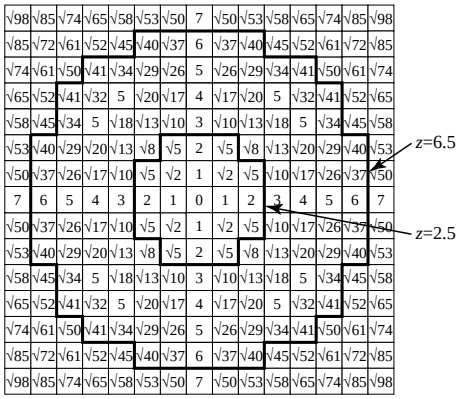

Figure 1. The Euclidean distances and distance masks for minimum distance search in z-clipped DT for clipping distances $z = 2.5$ and $z = 6.5$.

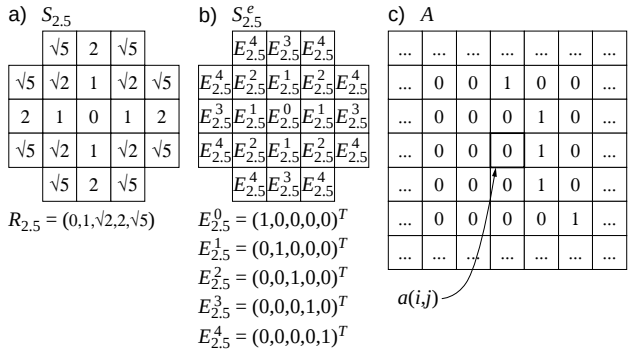

Figure 2. Example of the clipped distance transform with the clipping distance 2.5: a) the distance mask $S_{2.5}$, b) the corresponding order-based encoded distance mask $S^e_{2.5}$, and c) the neighborhood of a pixel $a(i, j)$ in an binary image A.

The distances can be represented with a binary valued encoded distance array E^k_z containing s_z elements as follows

$$E^k_z = (e^k_0, e^k_1, \ldots, e^k_{s_z-1})^T \tag{5}$$

where

$$e^k_l = \begin{cases} 1, & \text{if } l = k, 0 \le k < s_z \\ 0, & \text{otherwise} \end{cases}$$

With the previous arrays we may define a binary valued representation S^e_z of the Euclidean valued distance mask S_z

$$S^e_z(i,j) = \left\{ E^k_z \mid R_z E^k_z = \sqrt{i^2 + j^2}, (i,j) \in S_z \right\} \tag{6}$$

In order to find the set of distances to all the foreground pixels in the constrained neighborhood of the pixel $a(i, j)$ we form an encoded array $G^e(i, j \mid z)$ as follows

$$G^e(i,j \mid z) = \bigvee_{(m,n)\in S_z} a(i+m, j+n) S^e_z(m,n) \tag{7}$$

where $\bigvee$ is logical OR operation.

In the order-based encoded representation the elements of the distance vector are already ordered, thus finding the minimum corresponds to finding the first element with value one. The encoded z-clipped DT $d^e(i, j \mid z)$ is, therefore, defined as

$$d^e(i,j \mid z) = \min\left(G^e(i,j \mid z)\right) = (b_0, b_1, \ldots, b_{s_z-1})^T \tag{8}$$

where

$$b_k = \begin{cases} 1, & \text{if } b_l = 0, 0 \le l < k \text{ and } G^e_k(i,j \mid z) = 1 \\ 0, & \text{otherwise} \end{cases}$$

where $G^e_k(i, j \mid z)$ is the kth element in the $G^e(i, j \mid z)$ array.

The order-based encoded array $d^e(i, j \mid z)$ has to be decoded in order to obtain the final Euclidean valued z-clipped DT. This is easily performed with the aid of tabulated distance values in R_z defined in (4) as

$$d(i,j \mid z) = \begin{cases} z, & \text{if } d^e(i,j \mid z) = (0,\ldots,0)^T \\ R_z d^e(i,j \mid z), & \text{otherwise} \end{cases} \tag{9}$$

The previous procedure is illustrated with an example in Fig. 2 where clipping distance 2.5 is utilized. The corresponding Euclidean distance mask $S_{2.5}$ can be seen in Fig. 2.a) and the encoded distance mask $S^e_{2.5}$ in Fig. 2.b). When the encoded distance mask $S^e_{2.5}$ is applied, according to (7), to the pixel $a(i, j)$ with the neighborhood illustrated in Fig. 2.c) the result would be $G^e(i, j \mid 2.5) = (0, 1, 1, 1, 0)^T$. The minimum distance is found by selecting the first element with value one and clearing all the other elements which in this case results $d^e(i, j \mid 2.5) = (0, 1, 0, 0, 0)^T$. Decoding the encoded representation according to (9) results $d(i, j \mid 2.5) = R_{2.5} \cdot d^e(i, j \mid 2.5) = 1$.

3. HARDWARE ARCHITECTURE

The z-clipped DT algorithm described in the previous section can easily be applied to low-cost image processing systems where a separate camera is used and the image data is transferred in sequential fashion. In such systems, the sequential DT architecture can be used as a preprocessing unit connected to image data stream as illustrated in Fig. 3 where the analog video signal is digitized and transmitted as image data stream to feature extraction unit. This may be, e.g., a real-time edge detector producing binary image data stream. The next preprocessing unit would be the clipped DT which computes distance data stream on-the-fly to be postprocessed in a programmable processor.

The block diagram of the sequential clipped DT architecture is also shown in Fig. 3. The architecture is divided into four functional units: the neighborhood search unit, the minimum search unit, the decoding unit, and the control unit.

3.1. Neighborhood Search Unit

Due to the single-pass property and locality the sequential implementation may follow the approach used in [13] where the sequential image data slides under a hardwired mask. This means that the pixels under the mask need to be available at the same time instant, i.e., in order to access all the pixels under the distance mask S_z we need a FIFO array containing at most $2N\lceil z-1\rceil + 2\lceil z-1\rceil + 1$ 1-bit storage elements, i.e., a simple shift register.

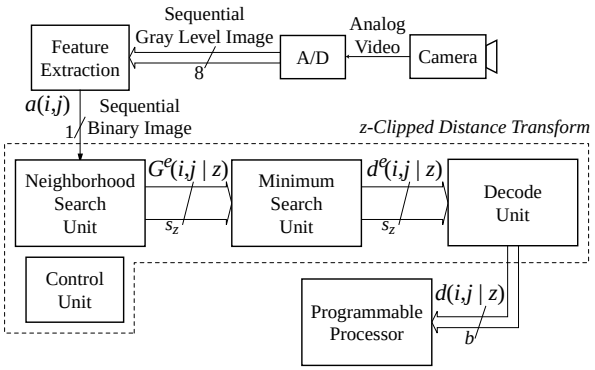

Figure 3. An example system utilizing the z-clipped DT and the block diagram of the clipped DT architecture.

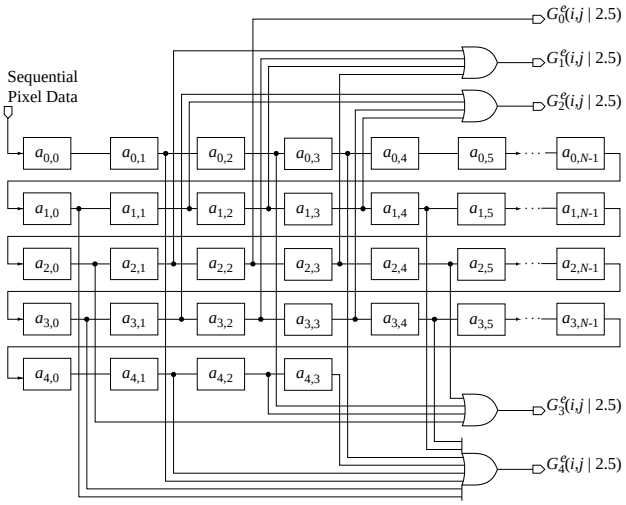

Figure 4. The neighborhood search unit for clipping distance 2.5.

Because the foreground pixels are labeled with binary one, according to (7), the presence of foreground pixels at equal distances within the distance mask can be identified by OR'ing all the pixel values at delay elements in the shift register corresponding to these locations. The structure of the search unit is illustrated in Fig. 4 where the foreground pixels are searched up to the clipping distance 2.5. Here notation $G_k^e(i,j \mid z)$ refers to the kth element of the array $G^e(i,j \mid z)$. It should be noted that the latency of the search is dependent on the size of the distance mask reflecting the length of the shift register.

3.2. Minimum Search Unit

The presence of foreground pixel at certain distance at the neighborhood is indicated by binary one at corresponding distance signal $G_l^e(i,j \mid z), l = 0,\ldots,s_z-1$. These signals are already ordered according to the distance value they represent, thus the minimum distance is found simply by finding the first signal containing binary value one and clearing all the subsequent signals. Such a structure for clipping distance 2.5 is illustrated in Fig. 5.

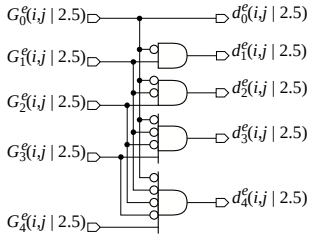

Figure 5. The minimum search unit for clipping distance 2.5.

3.3. Decoding Unit

The resulting distance values generated by the minimum search unit need to be decoded into standard number representation, but there are several choises for this. The Euclidean distance values are real numbers thus a floating-point representation would be suitable. However, in low-cost systems the floating-point arithmetic is typically too expensive for real-time processing and fixed-point representation is often used.

When converting the real valued distances into a fixed-point format, the real values are quantized according to the required accuracy and available word width. In practice this means scaling the distance metric by weight w and rounding the result to the nearest integer as follows

$$Q\left[d(i,j \mid z)\right]_w = \lfloor d(i,j \mid z)w + 0.5 \rfloor \tag{10}$$

Typically in DSP applications fractional fixed-point representation is utilized where the weight is power of two. However, when representing distances in square grids, better accuracy can be obtained by utilizing the weights in used in the chamfer DT algorithms, e.g., integer weights 3 and 5 [6]. Better approximations can be obtained by allocating more bits for the representation, i.e., utilizing larger weights.

The weight affects also the size of the distance table. This can be seen from Table 1 where the bit patterns of some quantized representations are listed. It can be seen that $Q\left[d(i,j \mid 4.0)\right]_3$ and $Q\left[d(i,j \mid 4.0)\right]_4$ require both 5-bit representation, but the weight 3 produces only eight different distance metrics with smaller relative maximum error ($e_3 = 0.0572\%$) than the fractional representation with weight 4 ($e_4 = 0.0607\%$). The same kind of behaviour can be expected when utilizing greater clipping distances. Once the representation is selected, the implementation of the decoding unit is simple decoding logic.

The previously described decoding approach results implementations that support only one number representation. This restricts the applicability of the fabricated DT chip. The interfacing can be generalized by replacing the quantized metrics in a table by index to a table, i.e., the metrics table is stored into the programmable processor instead of including them into the decoding logic. The decoding logic provides only indeces to the table which reduces the number of signals in the DT architecture and allows the floating-point format to be used for representing the metrics in table. The disadvantage is obviously the need for an extra table and an extra table access in the processor.

Although in the previous discussion the minimum search and the decoding are separate units, in practical implementations these may be combined because the intermediate encoded distance representation $d^e(i,j \mid z)$ is not needed and the final distances can

Table 1. The fixed-point representations of quantized distance metrics $Q\left[d(i,j \mid 4.0)\right]_w$.

Accurate Distance	$w=3$ 5-bit	$w=2^2$ 5-bit	$w=5$ 6-bit	$w=2^3$ 6-bit
0	00000	00000	000000	000000
1	00011	00100	000101	001000
$\sqrt{2}$	00100	00110	000111	001011
2	00110	01000	001010	010000
$\sqrt{5}$	00111	01001	001011	010010
$\sqrt{8}$	01000	01011	001110	010111
3	01001	01100	001111	011000
$\sqrt{10}$	01001	01101	010000	011001
$\sqrt{13}$	01011	01110	010010	011101

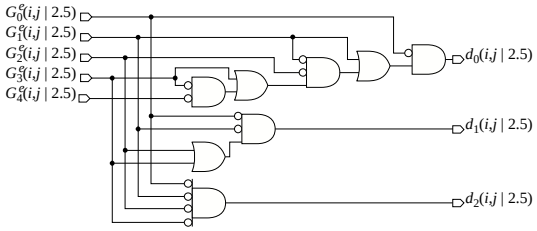

Figure 6. The combined minimum search and decoding unit for clipping distance 2.5.

be computed directly from the $G^e(i,j \mid z)$ signals in Fig. 3. This approach is reflected in Fig. 6 where the index to the table containing metrics for $d(i,j \mid 2.5)$ is formed directly from the encoded distance signals.

The whole architecture will contain multi-level logic networks which are the critical paths in the design especially when larger clipping distances are used. Fortunately these are combinatorial logic networks without any feedback loops, thus the paths can be avoided by pipelining the design. In addition, pipelining a structure of this kind is fairly easy with the aid of modern synthesis tools, which are capable of finding the optimal locations for the pipeline registers in the network for balancing the delays.

3.4. Control Unit

The control unit is needed to generate control signals according to line and frame synchronization pulses obtained from the video signal. Apart from the obvious timing signal generation tasks, the control unit has an important task for guiding the neighborhood search operation; in the shift register, pixel $a(i,N-1)$ at the end of an image line is followed by a pixel $a(i+1,0)$ at the beginning of the next image line thus those pixels are fed to the neighborhood search operation at the same time although they should never occur under the distance mask simultaneously. Therefore, when the distance mask is placed near the image borders, the control unit generates signals which force certain shift register elements to output zero value to the OR networks seen in Fig. 6.

4. SUMMARY

In this paper, we have applied previously presented parallel DT algorithm to sequential computation and described an area-efficient DT architecture which is well suited to high-speed operation. This architecture has been verified in VHDL environment and synthesized for a generic ASIC technology. Next we are developing FPGA prototypes to be used in image processing systems based on DSP processors for evaluating the total system performance. The performance and area measures obtained from the prototypes are to be compared with other DT algorithms and implementations.

5. REFERENCES

[1] Viitanen J., Hänninen P., Saarela R., and Saarinen J. "An Efficient Method for Image Pattern Matching". *Parallel Processing for Computer Vision and Display* (Dew P., Earnshaw R., Heywood T., Eds.), pp. 210-224, Addison-Wesley, 1989.

[2] Pei S., Ko C., and Su M. "Global Motion Estimation in Model-Based Image Coding by Tracking Three-Dimensional Contour Feature Points". *IEEE Trans. Circuits Syst. Video Technol.*, 8:181-190, 1998.

[3] Paglieroni D.W. "Distance Transforms: Properties and Machine Vision Applications". *CVGIP: Graph. Models Image Process.*, 54:56-74, 1992.

[4] Verbeek P.W., Dorst L., Verwer B.J.H., and Groen C.A. "Collision Avoidance and Path Finding in Through Constrained Distance Transformation in Robot State Space". *Proceedings of Int. Conf. Intelligent Autonomous Systems*, Amsterdam, the Netherlands, 1986, pp. 634-641.

[5] Mao J., and Mohiuddin K. "Form Dropout Using Distance Transform". *Proc. IEEE Int. Conf. Image Process.*, San Jose, CA, U.S.A., 1997, pp. 3:328-331.

[6] Borgefors G. "Distance Transformations in Digital Images". *Comput. Vision Graphics Image Process.*, 34:344-371, 1986.

[7] Viitanen J., and Kean, T. "Image Pattern Recognition Using Configurable Logic Cell Arrays". *New Advances in Computer Graphics* (Earnshaw R., Wyvill B., Eds.), pp. 355-368, Springer-Verlag, Tokyo, 1989.

[8] Takala J., and Viitanen J. "A Model-Based System for Vision Guidance of a Work Machine Manipulator". *Proc. Int. Conf. Machine Automation*, Tampere, Finland, 1994, pp. 319-325.

[9] Pan Y., Trahan J.L., and Vaidyanathan R. "A Scalable and Efficient Algorithm for Computing the City Block Distance Transform on Reconfigurable Meshes". *The Computer Journal*, 40:435-440, 1997.

[10] Verwer B.J.H., Verbeek P.W., and Dekker S.T. "An Efficient Uniform Cost Algorithm Applied to Distance Transforms". *IEEE Trans. Patt. Anal. Machine Intell.*, 11:425-429, 1989.

[11] Viitanen J., and Takala J. "SIMD Parallel Calculation of Distance Transformations". *Proc. IEEE Int. Conf. Image Process.*, Austin, TX, U.S.A., 1994, pp. 3:645-649.

[12] Takala J.H., and Viitanen J.O. "Distance Transform Algorithm for Bit-Serial SIMD Architectures". Submitted to *Comput. Vision Image Unterstanding*, 1998.

[13] Alzahrani F.M., and Chen T. "A Real-Time Edge Detector: Algorithm and VLSI Architecture". *Real-Time Imaging*, 3:363-378, 1997.