Digital squares

Forchhammer, Søren; Kim, Chul E

Published in:
9th International Conference on Pattern Recognition,

Link to article, DOI:
10.1109/ICPR.1988.28324

Publication date:
1988

Document Version
Publisher's PDF, also known as Version of record

Link back to DTU Orbit

Citation (APA):
DIGITAL SQUARES

Soren Forchhammer and Chul E. Kim

Institute of Telecommunication, 343,
Technical University of Denmark,
DK 2800 Lyngby, Denmark

Computer Science Department
Washington State University
Pullman, Washington 99164-1210

Abstract

Digital squares are defined and their geometric properties characterized. A linear time algorithm is presented that given a convex digital region, determines whether or not it is a digital square. The algorithm also determines the range of the values of the parameter set of its preimages.

1. Introduction

This paper extends the results on digital rectangles in [4] to digital squares and investigates the range of parameter values of the preimages of a digital square. To analyze these problems we transform the boundary of a digital region into parameter space of slope and y-intercept.

2. Preimages of digital line segments

Suppose that \( Q = \{d_0, d_1, \ldots, d_k\} \), where \( d_i = (x_i, y_i) \), is a digital line segment and assume without loss of generality that \( s_{i+1} = x_{i+1} = x_i + 1 \). Refer to [4] for definitions. Let \( H(Q) = (u_0, u_1, \ldots, u_k) \) be the convex hull of \( Q \) with its vertices listed in counterclockwise order. Let \( u_{i+1} \) be the longest edge of \( H(Q) \) and assume that the points of \( Q \) lie on and above its extension. Let \( f \) be the line segment which spans \( Q \) so that \( Q = fr(Q) \), the digital image of \( f \). Such a line segment is called a preimage of \( Q \). Obtain line segment \( f' \) by a parallel translation of \( f \) downward until \( f' \) contains points of \( Q \) with the least vertical distance from \( f \). Since \( u_i \) and \( u_{i+1} \) are the leftmost and rightmost digital points on \( f \), we also denote them by \( u_0 \) and \( u_k \), respectively. Now let \( u_0 \) and \( u_k \) denote the leftmost and rightmost digital points on \( f' \), where \( u_0 \) and \( u_k \) coincide when \( f' \) has only one digital point. Points \( u_0, u_k, u_t \) and \( u_v \), are called the limiting points of digital line segment \( Q \).

Given a digital arc \( Q = \{d_0, d_1, \ldots, d_k\} \), it can be decided in \( O(n) \) time whether or not \( Q \) is a digital line segment, and if it is, then its limiting points \( u_0, u_k, u_t, u_v \) are also computable in linear time [1].

3. Digital edges and corner points

A convex digital region \( R \) is a digital square if there is a square whose digital image is \( R \), and the boundary of \( R \) may also be called a digital square if \( R \) is. (See [3] for definition of digital convexity.)

Let \( Q = \{d_0, d_1, \ldots, d_k\} \) be a digital line segment. Then its chain \( C_Q = \{c_1, c_2, \ldots, c_k\} \) consists of at most two distinct chain links. We say that \( Q \) lies in the first octant if its chain \( C_Q \) consists of two distinct chain links \((1,0)\) and \((1,1)\). If \( C_Q \) consists of chain link \((1,1)\), then \( Q \) may be considered to lie in either the first or second octant.

A line segment \( uv \) from \( u = (x_u, y_u) \) to \( v = (x_v, y_v) \), where \( x_u, x_v, y_u \) and \( y_v \) are real and \( x_u < x_v > 0 \) and \( y_u < y_v > 0 \), is said to lie in the first octant if \( y_v - y_u < x_v - x_u \). Other octants in which a line segment may lie are similarly defined.

Figure 1. Partition of \( B \).

If the first point \( d_{i+1} \) of \( B_{i+1} \) is a 4-neighbor of the last point \( d_i \) of \( B_i \), then the digital edge \( Q_{i+1} \) is \( B_{i+1} \). Suppose that \( d_{i+1} \) is not a 4-neighbor of \( d_i \). If \( d_i \in B_{i+1} \), then \( d_i \) must be a digital line segment, then \( d_i \) is not contained in a quadrangle formed by any preimages of \( B_i \)'s and so \( R \) is not an odd digital quadrangle. Therefore, for \( R \) to be an odd digital quadrangle, \( d_i \in B_{i+1} \) must be a digital line segment and the digital edge \( Q_{i+1} \) is \( d_j \in R_{i+1} \). For each \( 1 \leq i \leq 4 \), the first point of \( Q \) is \( d_i \) and called the tail of \( Q \) and the last point of \( Q \) is \( d_k \) and called its head. Note that \( t_{i+1} \) must be \( d_i \) and if \( t_i, t_{i+1} \) is a 4-neighbor of \( h_i \). (See Figure 2).

4. Digital quadrangles

In the following we investigate necessary and sufficient conditions for a convex digital region \( R \) with four odd digital edges to be an odd digital quadrangle. As stated above, whether or not \( R \) is an odd quadrangle depends on the existence of preimages of digital edges that form a quadrangle not containing points in \( \mathbb{R}^2 \) which are neighbors of the digital corner points of \( R \).

CH2614-6/88/0000/0672$01.00 © 1988 IEEE
digital edge $Q_i$ of $R$ as shown in Figure 2 after rotation by a multiple of $\pi/2$. Point $q_i$ is the 4-neighbor of $R$ to its right and $p_i$ is the 8-neighbor of $h_i$ toward $Q_{i+1}$. The following lemma offers the desired necessary and sufficient conditions.

Lemma 4.1. Let $R$ be a convex digital region with four odd digital edges $Q_1$, $Q_2$, $Q_3$, and $Q_4$. Then $R$ is an odd digital quadrangle if and only if there are some preimages of $Q_i$'s forming a quadrangle that does not contain $q_i$ for any $1 \leq i \leq 4$.

Proof. If $R$ is an odd digital quadrangle, then there is an odd quadrangle $r$ whose digital image is $R$. Thus, trivially, $Q_i$ is the digital image of an extension of an edge of $r$ to span $Q_i$ for each $i$. Now suppose that there is an odd quadrangle $r$ with its edges formed by some preimages of $Q_i$'s such that $r$ does not contain $q_i$ for any $1 \leq i \leq 4$. Since $Q_i$ is the digital image of an extension of an edge of $r$ for all $i$, $r$ cannot contain any point in $\neg R$, except possibly $q_i$'s, that are neighbors of the boundary points of $R$. By assumption, however, $q_i$'s are contained in $r$ and thus, $R$ is the digital image of $r$ and hence, an odd digital quadrangle. □

Suppose that for each $1 \leq i \leq 4$, either $Q_i \cup \{q_i\}$ or $\{q_i\} \cup Q_{i+1}$ is not a digital line segment. Then no quadrangle formed by any preimages of $Q_i$'s contains $q_i$. For, any preimage of $Q_i$ or $Q_{i+1}$ passes between $h_i$ and $q_i$, excluding $q_i$ from the corner formed by any preimages of $Q_i$ and $Q_{i+1}$.

So now assume that for some $i$, $Q_i \cup \{q_i\}$ and $\{q_i\} \cup Q_{i+1}$, are both digital line segments. If neither $Q_i \cup \{p_i\}$ nor $\{h_i\} \cup Q_{i+1}$ is a digital line segment, then $q_i$ is contained inside the corner formed by any preimages of $Q_i$ and $Q_{i+1}$, and so $R$ is not an odd digital quadrangle. $q_i$ is excluded from inside the corner formed by every preimage of $Q_i \cup \{p_i\}$ combined with $Q_{i+1}$ and $Q_{i+1}$ combined with $\{h_i\} \cup Q_{i+1}$. If one and only one of $Q_i \cup \{p_i\}$ and $\{h_i\} \cup Q_{i+1}$ is a digital line segment then the legal change of $Q_i$ or $Q_{i+1}$ is performed. If both are digital line segments then $Q_i \cup \{p_i\}$ and $Q_{i+1}$ are considered along with $Q_i$ and $\{h_i\} \cup Q_{i+1}$.

These observations lead to the algorithm below that determines whether or not a given digital region $R$ is an odd digital quadrangle. Moreover, it returns a set of sets of four digital line segments whose preimages produce all preimages of $R$.

Algorithm odd-quadrangle (R,S).
// Given a convex digital region $R$, returns a set S of all digital line segments $(Q_1, Q_2, Q_3, Q_4)$ whose preimages produce quadrangles with $R$ as their digital image if $R$ is an odd digital quadrangle and returns $\emptyset$ otherwise. //

1. Obtain the boundary of $R$, $B = (d_0, d_1, \ldots, d_a)$, listed in counterclockwise order, and partition $B$ into $B_1, B_2, B_3$, and $B_4$.

2. If for some $i$, $B_i$ is not a digital line segment then set $S \leftarrow \emptyset$ and return.

3. For each $1 \leq i \leq 4$, obtain $Q_i$ from $B_i$ and determine $h_i$ and $t_i$.

4. If for each $1 \leq i \leq 4$, either $Q_i \cup \{q_i\}$ or $\{q_i\} \cup Q_{i+1}$ is not a digital line segment then $S := \{(Q_1, Q_2, Q_3, Q_4)\}$ and return.

5. $S := \{(Q_1, Q_2, Q_3, Q_4)\}$

6. For $i = 1$ to 4 do the following:

6.1. if $B_i \cup \{q_i\}$ and $\{q_i\} \cup B_{i+1}$ are both digital line segment

6.2. then for all current $(Q_1, Q_2, Q_3, Q_4)$ of $S$ do

6.2.1. $Q_i \leftarrow Q_i \cup \{p_i\}$

6.2.2. $Q_{i+1} \leftarrow \{h_i\} \cup Q_{i+1}$

6.2.3. $S' := S \cup \{(Q_1, Q_2, Q_3, Q_4)\}$

6.2.4. if $Q_i$ is a digital line segment then $S' := S' \cup \{(Q_1', Q_2, Q_3, Q_4)\}$

6.2.5. if $Q_{i+1}'$ is a digital line segment then $S' := S' \cup \{(Q_1, Q_2, Q_3', Q_4)\}$

6.2.6. return $S' := S$.

7. Return.

Theorem 4.2. Algorithm odd-quadrangle determines whether or not a given convex digital region $R$ is an odd digital quadrangle in $O(n)$ time, where $n$ is the number of boundary points of $R$. Moreover, if $R$ is an odd digital quadrangle, it returns sets of four digital edges whose preimages form all odd quadrangle $r$ which are a preimage of $R$.

Proof. The correctness of the algorithm is due to Lemma 4.1 and discussions preceding the algorithm as $S$ contains all the possible quadruples of digital line segments that excludes every $q_i$. It takes $O(n)$ time to obtain the boundary $B$ of $R$ assuming that $R$ is given by a run-length code and to partition $B$ because one traversal of $B$ is sufficient. So step 1 takes $O(n)$ time. For each $1 \leq i \leq 4$, whether or not $B_i$ is a digital line segment may be determined in time linear to the number of points in $B_i$ and thus step 2 in $O(n)$ time.

Trivially, step 3 takes constant time. Obviously step 4 also takes $O(n)$ time. Since steps 6.1 and 6.2 take $O(n)$ time and are iterated four times, and in each iteration there are at most 16 sets of digital edges, step 6 is $O(n)$ time. Step 7 is of $O(1)$ time, and so all steps together take $O(n)$ time.

Similarly an algorithm, called even-quadrangle, may be designed to determine whether or not a convex digital region is an even digital quadrangle and return its four digital edges if it is.

5. Digital squares

A square can be described by four parameters $(a, \delta, \epsilon, \zeta)$, where $\alpha$ is the slope of its edge in the first (second) octant, $\delta$ and $\epsilon$ are the $y$-intercept and $x$-intercept, respectively of the extension of the edge, and $\zeta$ its side length.

Consider an odd digital quadrangle $R$ and its four digital edges $(Q_1, Q_2, Q_3, Q_4)$ determined by the algorithm 'odd-quadrangle'. $R$ is a digital square if and only if there is a preimage $\ell_i$ of $Q_i$ for each $1 \leq i \leq 4$ such that $\ell_i$'s form a square $r$. Let $\alpha_i, \delta_i, \epsilon_i$ be the slope, $x$-intercept and $y$-intercept of the extension of $\ell_i$, respectively. Then $\alpha_i = \alpha_i', \epsilon_i = \epsilon_i', \delta_i = \delta_i$ and $\zeta_i = (1/\alpha_i) \epsilon_i$. Also if $p_1$ is the distance between $\ell_1$ and $\ell_2$ and $p_2$ is the distance between $\ell_2$ and $\ell_4$, then $\sqrt{1 + \alpha_1^2} p_1 = \delta_1 - \delta_1'$ and $\sqrt{1 + (1/\alpha_1)^2} p_2 = \epsilon_1 - \epsilon_1'$. Denoting $\delta_1, \delta_1'$
by \( p_1 \) and \( y_2+y_4 \) by \( p_2 \), we must have \( p_1' = p_2' \) because \( p_1 = p_2 \) and \( a_1 = 1/2a_2 \).

Consider the feasible parameter regions; \((a,\delta)\) domains of \( Q_1 \) and \( Q_3 \) and \(-(-a,\gamma)\) domains of \( Q_2 \) and \( Q_4 \). These domains are each either a triangle or a convex quadrangle as shown in (2). From \((a,\delta)\) domains of \( Q_1 \) and \( Q_3 \), we may obtain \((a_1, p_1') \) and \((\delta_1)\) domain by plotting \( a \) against \( p_1' = z_3 \delta_1 \) from \(-(-a,\gamma)\) domains of \( Q_2 \) and \( Q_4 \), obtain \((\pm a_2, p_2')\) domain by plotting \(-1/\alpha\) against \( p_2' = \gamma_2 \gamma_4 \) (Figure 3). Then for all \( 1 \leq i \leq 4 \), there is \( \xi_i \), so \( Q_i \) forming a square \( r \) which is a preimage of \( R \) if and only if \((a_1, p_1') \) and \((\pm a_2, p_2')\) domains have nonempty intersection.

![Figure 3](image)

**Algorithm odd-square \((R, a)\).**

1. If \( R \) is an odd digital square, returns \((a, p')\) in \( a \) for squares \( r \) whose digital image is \( R \) if and not, returns \( \emptyset \).
2. Set \( S = \emptyset \).
3. If \( S = \emptyset \), then stop.
4. For all \((Q_1, Q_2, Q_3, Q_4)\) of \( S \) do
   4.1. Obtain \((a,\delta)\) domains of \( Q_1 \) and \( Q_3 \) and \(-(-a,\gamma)\) domains of \( Q_2 \) and \( Q_4 \), each of which is either a triangle or a convex quadrangle.
   4.2. From \((a,\delta)\) domains of \( Q_1 \) and \( Q_3 \) obtain \((a_1, p_1')\) domain and from \(-(-a,\gamma)\) domains of \( Q_2 \) and \( Q_4 \) obtain \((-a_2, p_2')\) domain.
   4.3. Compute the intersection \( a' \) of \((a_1, p_1') \) and \((-a_2, p_2')\) domains and set \( a = a' \).
5. Return.

**Theorem 5.1.** Algorithm odd-square runs in \( O(n) \) time determining whether or not a given convex digital region \( R \) is an odd digital square. Moreover, if \( R \) is an odd digital square, it returns the range of \((a, p')\) values for which square \( r \) is a preimage of \( R \).

**Proof.** The discussions preceding the algorithm shows that it is correct. As shown in Theorem 4.2, step 1 takes \( O(n) \) time and trivially step 2 takes \( O(1) \) time. Since the limiting points of \( Q_i \) are computed in \( O(n) \) time for each \( 1 \leq i \leq 4 \), the domain of \( Q_i \) is computable in \( O(n) \) time, making step 4.1 \( O(n) \) time. Since \((a,\delta)\) domains of \( Q_1 \) and \( Q_3 \) are each either a triangle or a convex quadrangle, it takes constant time to compute \((a_1, p_1')\) domain, which is a convex polygon of at most 8 vertices. Similarly, \((-a_2, p_2')\) domain can be computed in constant time and so step 4.2 is of \( O(1) \) time. Step 4.3 also takes constant time, since finding the intersection of two convex polygons with at most 8 vertices can be computed in constant time (5). Therefore, the algorithm has \( O(n) \) time complexity.

An \( O(n) \) time algorithm, called even-square, may also be designed to test whether or not a given even digital quadrangle is a digital square. Therefore, given a convex digital region \( R \), we determine whether it is a digital quadrangle in \( O(n) \) time.

**References**