アルゴリズム 44－7
（1995．3．17）

誤差耐性のある強凸包構成問題に対する並列解法

陳慰 和田幸一 川口喜三男

名古屋工業大学電気情報工学科

平面上にある n 個の点集合 Sが与えられたとき, Sのϵ-強凸δ-包 P は次のように定義される凸多角形である: (i)Pの頂点は Sに属する, (ii) Pの外にある Sの点と Pとの距離はδ以内である, (iii) Pの各頂点を任意にϵだけ移動しても Pは凸である. 本稿ではこの一般化した凸包問題を解く並列アルゴリズムを提案する. その結果として, Sのϵ-強凸 $O(\epsilon)$-包は, (1) 誤差のない演算を用いるとき $O(\log n)$ 時間, n プロセッサで求められる, (2) 丸め誤差を含む演算を用いるとき $O(\log^3 n)$ 時間, n プロセッサで求められる. ここで提案する並列アルゴリズムは, 逐次アルゴリズムとみなすと既知の逐次アルゴリズムと同じ計算時間でより正確な強凸包を求めることができるので, 改良した新しい逐次アルゴリズムにもなる. 本研究で用いる並列計算モデルは CREW PRAM である.

Parallel Methods for Constructing Strongly Convex Hull Using Exact and Rounded Arithmetic

Wei Chen, Koichi Wada and Kimio Kawaguchi

Department of Electrical and Computer Engineering,
Nagoya Institute of Technology
Showa, Nagoya 466, Japan

The ϵ-strongly convex δ-hull of a set S of n points in the plane is defined as a convex polygon P with the vertices taken from S such that no point of S lies farther than δ outside P and such that even if the vertices of P are perturbed by as much as ϵ, P remains convex. In this paper we propose parallel methods for computing such a generalized convex hull. Using exact arithmetic we can construct an ϵ-strongly convex $O(\epsilon)$-hull in $O(\log n)$ time using n processors, and using rounded arithmetic we can construct an ϵ-strongly convex $O(\epsilon+\mu)$-hull in $O(\log^3 n)$ time using n processors, where μ is the rounded unit. Our algorithms can be also taken as the improved sequential algorithms which run faster and compute the solution more precisely than the known sequential algorithms. The parallel computation model which we use in this paper is CREW PRAM.

1 Introduction

Traditional geometric algorithms in which numerical issues are not usually considered arise surprisingly problems in practice. The major reason is that the basic geometric tests needed to solve the problems are unreliable or inconclusive when implementing imprecise computations such as ordinary floating point arithmetic. This uncertainty makes the constructed geometric objects inaccuracy or even not satisfying the proposed geometric properties. Recently, robust geometric algorithms whose correctness are not spoiled by numerical errors have been well developed [7,9,11-19]. Among them, robust convex hull algorithms attract increasing attention since convex hull problems are the simplest and oldest problems in computational geometry and have many applications. Assuming that the coordinates of the input points are N-bit integers and the operations on N-bit integers have unit cost, in order to avoid rounding in all cases (the integers are converted into floating point before performing additions and multiplications). On the other hand, the exact algorithm must use $(2N+1)$-bit integer arithmetic, but the robust algorithms requires only N-bit (mantissa) in floating point arithmetic. Fortune [9] describes an $O(n \log n)$ time robust algorithm for computing an ϵ-weakly convex hull H of a set S of n points, meaning that each vertex may have to be perturbed by as much as ϵ in order to make H convex. The error ϵ is a small multiple of the rounding unit μ (defined in Section 3). One drawback of this algorithm is that the resulting hull is only approximately convex and therefore does not enjoy many of nice properties associated with convexity. In fact, considering the situation that the output of one robust algorithm may become the input to another robust algorithm, we may desire that the resulting approximate convex hull is not only weakly convex, even not only convex, but strongly convex, where an ϵ-strongly convex hull remains convex even if its vertices are perturbed by as much as ϵ, so that many of the desirable properties are preserved in some fashion even they are tested with floating-point arithmetic. Li and Milenkovic [15] present the first algorithm for computing an ϵ-strongly convex $(12\epsilon+\delta)$-hull of S, that is, the result is ϵ-strongly convex and no point of S lies farther than δ outside it, where $\delta = 0$ for exact arithmetic and $\delta = 288\sqrt{2}\mu$ for rounded arithmetic. Their algorithm runs in $O(n \log n)$ time. Guibas, Salesin and Stolfi [13] explain an $O(n^3 \log n)$ algorithm for constructing an ϵ-strongly $(6\epsilon+\beta)$-hull, where $\beta = 0$ for exact arithmetic and $\beta = 7\lambda$ (λ is defined as an error unit for a basic geometric operation) for imprecise computations including rounded arithmetic.

In this paper, we present parallel robust algorithms for convex hull problems in CREW PRAM. Many parallel convex hull algorithms have been developed in CREW PRAM [1-6,8], but none of them considers numerical issues. Our algorithms show the following results: (1) using exact arithmetic, an ϵ-strongly convex 6ϵ-hull of a set S of n points can be constructed in $O(\log n)$ time using n processors, (2) using rounded arithmetic, a convex $64\sqrt{2}\mu$-hull of S can be constructed in $O(\log^3 n)$ time using n processors, (3) using rounded arithmetic, an ϵ-strongly convex $(6\epsilon + 96\sqrt{2}\mu)$-hull of a convex polygon can be constructed in $O(\log n)$ time using n processors, and (4) using rounded arithmetic, an ϵ-strongly convex $(8\epsilon + 8\lambda + 128\sqrt{2}\mu)$-hull of a λ-weakly convex polygon can be constructed in $O(\log^2 n)$ time using n processors, in CREW PRAM. From the results (2) and (3), an ϵ-strongly convex $(6\epsilon + 160\sqrt{2}\mu)$-hull of S can be constructed in $O(\log^3 n)$ time using n processors in CREW PRAM. There are a number of reasons why these results are important. The main contribution is that we give the first efficient robust parallel methods for convex hull problems. Traditional parallel methods for convex hull problems cause large numerical errors internally and the known efficient robust sequential algorithms are difficult to be parallelized. In addition, our algorithms implies a new improved sequential algorithm for constructing a strongly convex hull. Guibas, Salesin and Stolfi's algorithm computes a more precise strongly convex hull than Li and Milenkovic's but it runs very slowly. Comparing with them, our algorithm computes a strongly convex hull whose precision is the same as that of the former when using exact algorithm and at least close to that of the former (since the error unit is defined abstractly in their algorithm, it is difficult to compare precisely) when using rounded arithmetic, but the cost (product of the running time and the number of the processors) of our algorithm is the same as the latter when using exact algorithm and close to the latter when using rounded arithmetic. We expect that our robust parallel methods can be generalized for many other geometric problems.

2 Definitions and Lemmas

Definition 1 *(1) A simple polygon P is a δ-hull of a set of points S if all vertices of P belong to S and if no point of S is farther than δ from the polygonal region bound by P.*

(2) A simple polygon P is ϵ-weakly convex ($\epsilon \geq 0$) if there exists some way of perturbing each vertex of P no farther than ϵ so that P becomes convex.

(3) A simple polygon P is ϵ-strongly convex ($\epsilon \geq 0$) if P is convex and remains convex even after each vertex of P is perturbed as far as ϵ. ∎

From Definition 1, we see that any convex polygon is a 0-weakly and 0-strongly convex polygon. For a polygonal chain C, $\overline{C}$ denotes the polygon bound by C and the edge connecting the first and last vertices of C. If $\overline{C}$ is a simple polygon, C is said *a simple polygonal chain.* We call the first and last vertices of C *end vertices* and the other vertices *internal vertices.* We use terms δ-chain, ϵ-weakly convex chain and ϵ-strongly convex chain for a polygonal chain C if the polygon consists of the internal vertices of C satisfies these properties. Let a simple polygon (or a simple polygonal chain) be represented by the sequence of the vertices in clockwise. The following

definition and lemma reduce strong convexity to a local property.

Definition 2 *Let A, B, and C be contiguous vertices in clockwise order of a simple polygon (or a simple polygonal chain). We say that vertex B is ϵ-convex if the distance from B to line AC is greater than ϵ.* ∎

No matter where we move B inside a disk of radius ϵ, it cannot cross line AC and angle ABC remains convex.

Lemma 1 *If each vertex of a simple polygon (or each internal vertex of a simple polygonal chain) P is 2ϵ-convex, then P is ϵ-strongly convex.* ∎

3 Exact and rounded arithmetic

Consider the following basic geometric operations: (I) $D(p,l)$: computes the distance from point p to line l, (II) $\Theta(AB,CD) \leq \theta$: determines if the angle of line segments AB and CD which equals to the change in orientation from AB to CD is not larger than θ, where θ is a very small angle, (III) $Slop(l)$: measures the slope of line l, and (IV) $Pos(A,B,C)$: determines the orientation of A, B, C which is counterclockwise, clockwise or collinear. We use rounded floating point arithmetic with a P-bit mantissa. Let $\mu = 2^{-P}M$ be the rounding unit of P-bit arithmetic, where M is a bound on the magnitude of the coordinates in the inputs, the following lemma holds.

Lemma 2 (15) *Using P-bit floating point arithmetic, the basic geometric operation (I) can be executed with absolute error at most $16\sqrt{2}\mu$, and operations (II), (III) and (IV) can be executed with absolute error at most $4 \cdot 2^{-P}$.* ∎

For convenience, we use the notations $(D(p,l))_P$, $(Slop(l))_P$, $(\Theta(AB,CD) \leq \theta)_P$ and $(Pos(A,B,C))_P$ to represent rounded arithmetic computations using P-bit.

There is another operation in our algorithms. Let $X(A)$ and $Y(A)$ denote the x and y coordinate of point A, respectively. Use $l(A,B)$ to denote the line passing through points A and B, where $X(A) \leq X(B)$. Point C is said above (below) $l(A,B)$ if $Pos(C,A,B)$ is counterclockwise (clockwise). Therefore, we can determine if C is above (below) $l(A,B)$ by the basic operation (IV). Let S be a set of points. We let $Abov(S,l)$ and $(Abov(S,l))_P$ denote the operation for computing the set $\{p \mid p \in S \text{ and } p \text{ is above } l\}$, using exact and rounded arithmetic, respectively. For any two points A and B of the input, $|AB| \leq 2\sqrt{2}M$ obviously. If an angle θ is very small, we can treat the value of $\sin\theta$ as the actual value of the angle. Therefore, if $\Theta(C,AB)$ is very small, $D(C,AB) \leq 2\sqrt{2}M\Theta(C,AB)$. Finally, notice that the pure comparison is not a geometric operation and arises no error.

4 Strongly convex hull of a weakly convex polygon

In what of the follows, the parallel computations are always considered in CREW PRAM.

4.1 Exact arithmetic method

Let $P = (p_1, p_2, \ldots, p_n)$ be a λ-weakly convex polygon (or polygonal chain). We use $P(i..j)$ to denote the vertices of P between p_i and p_j, i.e., the vertices p_i, p_{i+1}, $\ldots, p_j$. If vertieces are not ϵ-convex they are called *ϵ-flat* vertices, especially, we call the first and last vertices of a polygonal chain *ϵ-flat* vertices. We call the edges whose two endpoints are both ϵ-flat as *ϵ-flat* edges. In this section, we construct an ϵ-strongly convex γ-hull of a λ-weakly convex polygon P using exact arithmetic, where $\gamma = 6\epsilon$ for $\lambda = 0$ and $\gamma = 8\epsilon + 8\lambda$ for $\lambda > 0$. We first create *a ridgepiece* of P which plays an important role in our algorithms defined as follows.

Definition 3 *Let $P = (p_1, p_2, \ldots, p_n)$ be a λ-weakly convex polygon. Polygon $r(P)$ is said an (ϵ,λ)-ridgepiece of P if it satisfies the following conditions:*
(1) each vertex of $r(P)$ is a vertex of P,
(2) among any three contiguous vertices of $r(P)$, there exists at least one $(2\epsilon + 2\lambda)$-convex vertex, and
(3) when $\lambda = 0$, $r(P)$ is a 2ϵ-hull of P (Fig. 1(a)), and when $\lambda > 0$, $r(P)$ is a $2(2\epsilon + 2\lambda)$-hull of P, but the vertices of P between two endpoints of a $(2\epsilon + 2\lambda)$-flat edge e of $r(P)$ are at most $2\epsilon + 2\lambda$ away from e (Fig. 1(b)). ∎

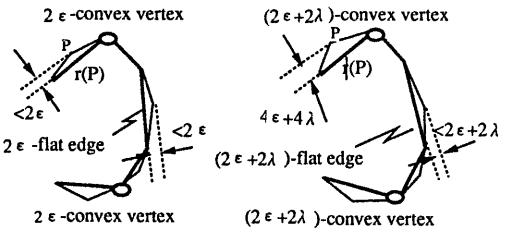

Fig. 1(a) ridgepiece for λ=0 Fig. 1(b) ridgepiece for λ>0

Figure 1: Definition of ridgepieces

Let F be a λ-weakly convex chain. Just by exchanging the word "polygon" into "chain" in the above definition, a (ϵ,λ)-ridgepiece of F, $r(F)$, can be defined similarly, except that the first and last vertices of $r(F)$ must be the first and last vertices of F. An (ϵ,λ)-ridgepiece is simply called ridgepiece if no confusion arises. The ridgepieces consisting of two or three vertices are the simplest ones. Following properties hold for them. Assuming $F = (u_1, u_2, \ldots, u_f)$, (1) $r(F)$ consists of two vertices *iff* $r(F) = (u_1, u_f)$ and each vertex of F is at most $2\epsilon + 2\lambda$ away from $r(F)$, and (2) $r(F)$ consists of

three vertices *iff* $r(F) = (u_1, u_i, u_f)$, where $1 < i < f$, u_i is a $(2\epsilon + 2\lambda)$-convex vertex in $r(F)$, and each vertex of F is at most γ away from (u_1, u_i, v_f), where $\gamma = 2\epsilon$ for $\lambda = 0$ and $\gamma = 2(2\epsilon + 2\lambda)$ for $\lambda > 0$. The following two lemmas suggest a divide-and-conquer method for computing a ridgepiece of F.

Lemma 3 *Let $F = (u_1, u_2, \ldots, u_f, w, v_1, v_2, \ldots, v_g)$ be a λ-weakly convex chain. If (u_1, w) is a ridgepiece of $U = (u_1, u_2, \ldots, u_f, w)$ and (w, v_g) is a ridgepiece of $V = (w, v_1, v_2, \ldots, v_g)$, a ridgepiece of F consisting of at most three vertices can be found in $O(1)$ time using $O(|F|)$ processors when $\lambda = 0$ and in $O(\log|F|)$ time using $O(|F|/\log|F|)$ processors when $\lambda > 0$.*

(Proof:) First consider the case that $\lambda = 0$. In this case F is a convex polygonal chain. From the convexity, distance function $D(x, u_1u_g)$ for $x \in F$ is unimodal. Compute $D(x, u_1v_g)$ for each $x \in F$. If all vertices of F are at most 2ϵ away from u_1v_g, (u_1, v_g) is a ridgepiece of F. Assume that there exists at least one vertex which is farther than 2ϵ from u_1v_g. If $D(w, u_1v_g) \geq 2\epsilon$, (u_1, w, v_g) is a ridgepiece of F. Assume $D(w, u_1v_g) < 2\epsilon$. From the unimodality, the vertices of F father than 2ϵ from u_1v_g are contiguous (Fig.2). Let these vertices be $X = (x_1, x_2, \ldots, x_k)$ $(k \geq 1)$. Since w is not the vertex of X it must be the vertex before of x_1 or after of x_k. If w is before of x_1, (u_1, x_1, v_g) is a ridgepiece of F since x_1 is 2ϵ-convex, and for any vertex $x \in F$ if x is between u_1 and x_1, $D(x, u_1x_1) \leq D(x, u_1v_g) \leq 2\epsilon$ else if x is between x_1 and v_g, $D(x, x_1v_g) \leq D(x, wv_g) \leq 2\epsilon$. Similarly, we can prove that if w is after of x_k, (u_1, x_k, v_g) is a ridgepiece of F. From unimodality, all the above computations can be executed in constant time using $|F|$ processors.

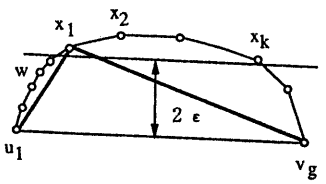

Figure 2: Finding a 2ϵ-convex vertex

Next consider the case that $\lambda > 0$. Find a vertex y such that $D(y, u_1v_g) = \max\{D(x, u_1v_g) \mid x \in F\}$. If $D(y, u_1v_g) \leq 2\epsilon + 2\lambda$, (u_1, v_g) is a ridgepiece of F, else (u_1, y, v_g) is a ridgepiece of F since y is $2\epsilon + 2\lambda$-convex, and all vertices of F are at most $2(2\epsilon + 2\lambda)$ away from (u_1, y, v_g). This ends the proof since the maximum computation can be executed in $O(\log|F|)$ time using $|F|/\log|F|$ processors [10]. ∎

Lemma 4 *Let $F = (u_1, u_2, \ldots, u_f, w, v_1, v_2, \ldots, v_g)$ be a λ-weakly convex chain. Given Z_1 and Z_2, the ridgepieces of $U = (u_1, u_2, \ldots, u_f, w)$ and $V = (w, v_1, v_2, \ldots, v_g)$, respectively, a ridgepiece of F can be computed in $O(1)$ time using $O(|F|)$ processors when $\lambda = 0$ and in $O(\log|F|)$ time using $O(|F|/\log|F|)$ processors when $\lambda > 0$.*

(Proof:) Let $Z_1 = (z_1, z_2, \ldots, z_k, z)$ and $Z_2 = (z, z_{k+1}, \ldots, z_h)$, where $z_i \in U$ for $1 \leq i \leq k$, $z_i \in V$ for $k + 1 \leq i \leq h$, $z = w$, $z_1 = u_1$ and $z_h = v_g$. $Z = (Z_1, Z_2)$ $(= (z_1, z_2, \ldots, z_k, z, z_{k+1}, \ldots, z_h))$ satisfies all conditions as a ridgepiece of F except that there may appear three contiguous $(2\epsilon + 2\lambda)$-flat vertices at the junction of Z_1 and Z_2. To see this, let z^* be the last $(2\epsilon + 2\lambda)$-flat vertex of Z_1 and z^{**} be the first $(2\epsilon + 2\lambda)$-flat vertex of Z_2. By the definition of a ridgepiece, z^* can be z_k or z and z^{**} can be z or z_{k+1}. When $z^* = z_k$ and $z^{**} = z_{k+1}$, three contiguous vertices z_k, z, z_{k+1} of Z are all $(2\epsilon + 2\lambda)$-flat (Fig. 3). We solve this problem as follows. Let the vertices of F between z_k and z be $F(z_k..z)$ and the vertices of F between z and z_{k+1} be $F(z..z_{k+1})$. Since (z_k, z) and (z, z_{k+1}) are the ridgepieces of $F(z_k..z)$ and $F(z..z_{k+1})$, using Lemma 3 we can find, $r(F(z_k..z_{k+1}))$, a ridgepiece of $F(z_k..z_{k+1})$ which is either (z_k, z_{k+1}) or (z_k, y, z_{k+1}), where $y \in F(z_k, z_{k+1})$, in $O(1)$ time using $O(|F|)$ processors when $\lambda = 0$ and in $O(\log|F|)$ time using $O(|F|/\log|F|)$ processors when $\lambda > 0$. If $r(F(z_k..z_{k+1})) = (z_k, z_{k+1})$, delete z from Z, and if $r(F(z_k..z_{k+1})) = (z_k, y, z_{k+1})$, delete z and insert y to Z. The resulting Z is a ridgepiece of F. ∎

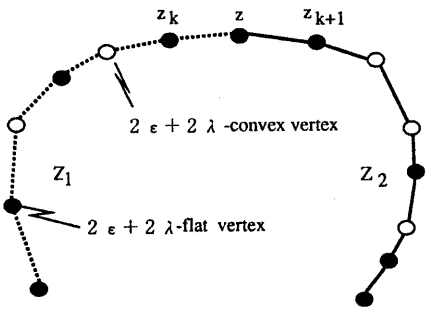

Figure 3: Method for marging two ridgepieces

The following algorithm MakeRidgeChain(F) constructs a ridgepiece for a λ-weakly convex chain F.

Algorithm MakeRidgeChain(F)

(Step 1) Let $F = (u_1, u_2, \ldots, u_m)$. If $m = 2$, (u_1, u_2) is a ridgepiece of F. This completes the algorithm. Else execute the following steps.

(Step 2) Divide F into F_1 and F_2 such that $F_1 = (u_1, u_2, \ldots, u_{m/2})$ and $F_2 = (u_{m/2}, u_{m/2+1}, \ldots, u_m)$. In parallel compute a ridgepiece of F_1 and a ridgepiece of F_2, recursively.

(Step 3) Marge the ridgepiece of F_1 and the ridgepiece of F_2 into a ridgepiece of F by Lemma 4. ∎

Algorithm MakeRidgeChain(F) uses divide-and-conquer technique. From Lemma 4, each recursive step runs in $O(1)$ time using $O(m)$ processors when $\lambda = 0$ and

in $O(\log m)$ time using $O(m)$ processors when $\lambda > 0$. Therefore, the following theorem holds.

Theorem 1 *Let F be a λ-weakly convex chain with m vertices. A ridgepiece of F can be constructed in $O(\log m)$ time using m processors when $\lambda = 0$ and in $O(\log^2 m)$ time using m processors when $\lambda > 0$.*

Theorem 2 *Let $P = (p_1, p_1, \ldots, p_n)$ be a λ-weakly convex polygon with n vertices. A ridgepiece of P can be constructed in $O(\log n)$ time using n processors when $\lambda = 0$ and in $O(\log^2 n)$ time using n processors when $\lambda > 0$.*

Proof: Polygon P can be changed into a closed polygonal chain $P' = (p_1, p_1, \ldots, p_n, p_{n+1})$ by adding a vertex p_{n+1} which is equal to p_1. Construct a ridgepiece of P', $r(P') = (q_1, q_2, \ldots, q_m)$, where $q_1 = p_1$ and $q_m = p_{n+1}(= q_1)$. Let R be the polygon bound by $r(P')$. Polygon R satisfies all conditions as a ridgepiece of P, except the problem that there may exist three contiguous $(2\epsilon + 2\lambda)$-flat vertices at the junction, that is, all of q_{m-1}, q_1 $(= q_m)$ and q_2 may be $2\epsilon + 2\lambda$-flat. Since the same problem has been happened in Lemma 4, we solve the problem in the same way. That is, let the vertices of P between q_{m-1} and q_1 be $P(q_{m-1}..q_1)$ and the vertices of P between q_1 and q_2 be $P(q_1..q_2)$. Using Lemma 3 find $r(P(q_{m-1}..q_2))$, a ridgepiece of $P(q_{m-1}..q_2)$, which is either (q_{m-1}, q_2) or (q_{m-1}, y, q_2), where $y \in P(q_{m-1}..q_2)$. If $r(P(q_{m-1}..q_2)) = (q_{m-1}, q_2)$ delete q_1 from R and if $r(P(q_{m-1}..q_2)) = (q_{m-1}, y, q_2)$ delete q_1 and insert y to R. The resulting R is a ridgepiece of P. ∎

For any three points u, v and w, where $X(u) < X(v) < X(w)$, angle uvw is said β-convex if u, v and w are in clockwise and $D(v, uw) \geq \beta$. The following lemma guarantees that the polygon obtained by removing all $(2\epsilon + 2\lambda)$-flat vertices from a ridgepiece is 2ϵ-convex.

Lemma 5 *Let $P = (p_1, p_2, \ldots, p_n)$ be a λ-weakly convex polygon. For any three vertices u,v and w, if there exist vertex p between u and v and vertex q between v and w such that angle pvq is $(2\epsilon + 2\lambda)$-convex then angle uvw is 2ϵ-convex.*

Proof: Assume that angle uvw is not 2ϵ-convex. See Fig. 4, where l_1 is a line parallel to the line segment pq and 2ϵ away from v, l_2 is the line passing through vertex v vertical to l_1, a is the intersection of l_1 and l_2, and b is the intersection of l_2 and pq, respectively. It is easy to see that if angle uvw is not 2ϵ-convex, at least one of u and w is above l_1. Without loss of generality, let u be above l_1. If $|vp| \geq |up|$, then $|vp| \geq |up| > |ab| \geq 2\lambda$ holds. This contradicts with the condition that P is λ-weakly convex since however perturbing each vertex of u, p and v within λ distance angle upv can not become convex. If $|vp| < |up|$, $|up| > |vp| > |ab| \geq 2\lambda$ holds. This conducts the same contradiction. ∎

Theorem 3 *Given a λ-weakly convex polygon P, an ϵ-strongly convex 6ϵ-hull can be computed in $O(\log n)$ time using n processors when $\lambda = 0$, and an ϵ-strongly convex $4(2\epsilon + 2\lambda)$-hull can be computed in $O(\log^2 n)$ time using n processors when $\lambda > 0$.*

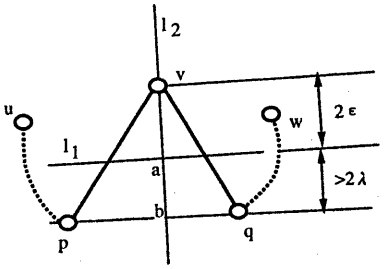

Figure 4: Proof of Lemma 5

Proof: Construct $r(P)$, a ridgepiece of P. From the definition of the ridgepiece, $r(P)$ has the following properties: (i) among any three contiguous vertices of $r(P)$ there exists at least one $(2\epsilon + 2\lambda)$-convex vertex, and (ii) when $\lambda = 0$, $r(P)$ is a 2ϵ-hull of P and when $\lambda > 0$, $r(P)$ is a $2(2\epsilon + 2\lambda)$-hull of P but the vertices of P between the endpoints of a $(2\epsilon + 2\lambda)$-flat edge e are at most $(2\epsilon + 2\lambda)$ away from e. Let R be the polygon obtained by removing the $(2\epsilon + 2\lambda)$-flat vertices from $r(P)$. Lemma 5 guarantees that R is ϵ-strongly convex (for any three $(2\epsilon + 2\lambda)$-convex vertices a, b, c of $r(P)$ in clockwise, let a, b, c correspond to u, v, w of Lemma 5). Between any two adjacent $(2\epsilon + 2\lambda)$-convex vertices of $r(P)$ there exist at most two $(2\epsilon + 2\lambda)$-flat vertices. Since removing one flat vertex from $r(P)$ let each vertex of $r(P)$ at most $2\epsilon + 2\lambda$ away from R, R is an ϵ-strongly convex 6ϵ-hull of P when $\lambda = 0$, and an ϵ-strongly convex $4(2\epsilon + 2\lambda)$-hull of P when $\lambda > 0$. ∎

When $\lambda = 0$, λ-weakly convex polygon P is a convex polygon for which we have the following theroem.

Theorem 4 *Using exact arithmetic an ϵ-strongly convex 6ϵ-hull of a set S of n points in the plane can be computed in $O(\log n)$ time using n processors.*

Proof: First, construct $CH(S)$, the convex hull of S, in $O(\log n)$ time using n processors []. From Theorem 3, an ϵ-strongly convex 6ϵ-hull of $CH(S)$ can be computed in $O(\log n)$ time using n processors. This completes the proof. ∎

4.2 Rounded arithmetic method

We implement the algorithm of Section 4.1 using rounded P-bit arithmetic. the only geometric operation needed in Section 4.1 is to determine if $D(C, AB) \leq 2\epsilon + 2\lambda$ (in other words if C is $(2\epsilon + 2\lambda)$-convex). From Lemma 2, if $(D(C, AB) \geq 2\epsilon + 2\lambda + 16\sqrt{2}\mu)_P$, then $D(C, AB) \geq 2\epsilon + 2\lambda$ holds. Therefore, replacing $2\epsilon + 2\lambda$ by $2\epsilon + 2\lambda + 16\sqrt{2}\mu$ in the algorithm of Section 4.1 wherever it appears (in the case of $\lambda = 0$, replacing 2ϵ by $2\epsilon + 16\sqrt{2}\mu$) assures that the rounded version will still generate an ϵ-strongly convex polygon. Since determining whether

$(D(C,AB) \geq 2\epsilon + 2\lambda + 16\sqrt{2}\mu)_P$ brings an adding error of $16\sqrt{2}\mu$, the output is a $(3(2\epsilon + 32\sqrt{2}\mu)$-hull when $\lambda = 0$ and $(4(2\epsilon + 2\lambda + 32\sqrt{2}\mu)$-hull when $\lambda > 0$.

Theorem 5 *Given a λ-weakly convex polygon P, an ϵ-strongly convex $(6\epsilon + 96\sqrt{2}\mu)$-hull can be computed in $O(\log n)$ time using n processors when $\lambda = 0$, and an ϵ-strongly convex $(8\epsilon + 4\lambda + 128\sqrt{2}\mu)$-hull can be computed in $O(\log^2 n)$ time using n processors when $\lambda > 0$.* ∎

5 Robust convex hull algorithm

In this section we construct a convex $64\sqrt{2}\mu$-hull of a set of n points using rounded arithmetic in $O(\log^3 n)$ time using n processors. Therefore an ϵ-strongly $(6\epsilon + 160\sqrt{2}\mu)$-hull can be constructed in $O(\log^3 n)$ time using n processors from Theorem 5. Since $\log_2(160\sqrt{2}) < 7$, the robust algorithm requires only $(N+7)$-bit arithmetic.

First, extract four sequences from S which cover the vertices of the convex hull of S (Fig. 5). For each sequence, the x and y coordinates are monotonic function of indices. To generate H, one of the sequences, where both x and y coordinates are in increasing order, sort S by increasing x coordinate in $O(\log n)$ time using n processors[]. For each point p of sorted S, find point q before p such that q has the largest y coordinate, and then draw out q. This can be executed by prefix maxima computation and the doubling technique in $O(\log n)$ time using n processors [10]. We can generate the other three sequences similarly.

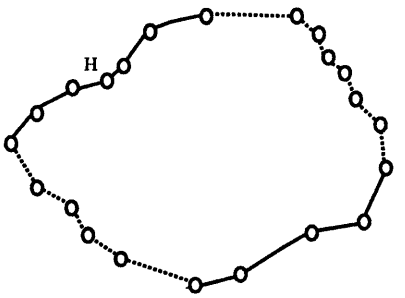

Figure 5: Monotonic sequences covering the vertices of the convex hull

In the following, we describe how to construct a convex $64\sqrt{2}$-hull for each monotonic sequence. We only construct a convex $64\sqrt{2}$-hull for sequence H. The work for other three sequences can be done in the same way. Our algorithm guarantees that four extreme points of S are included in these four convex $64\sqrt{2}$-hulls. Putting the four convex $64\sqrt{2}$-hulls together, we get a convex $64\sqrt{2}$-hull of S.

A line l is defined as *a support line* of H if l passes through at least one vertex of H and all other vertices of H are below of l. A line l is said *an α-support line* ($\alpha \geq 0$) of H if l passes through at least one vertex of H and any vertex of H above l is at most α away from l. Let H_1 and H_2 be two subsequences of H separated by a vertical line. A line segment ab is said *a bridge* of H_1 and H_2 if (i) $a \in H_1$ and $b \in H_2$, and (ii) each vertex of H_1 and H_2 is under line $l(a,b)$. A line segment $a'b'$ is said *an α-bridge* of H_1 and H_2 if (i) $a' \in H_1$ and $b' \in H_2$, (ii) each vertex of H_1 and H_2 above line $l(a',b',)$ is at most α away from $l(a',b')$.

Lemma 6 *Let $H = (u_1, u_2, \ldots u_m)$ be a monotonic polygonal chain, where both x and y coordinates of the vertices are the increasing order. For the polygonal chains $H_1 = (u_1, u_2, \ldots u_{m/2})$ and $H_2 = (u_{m/2+1}, u_2, \ldots, u_m)$, either the following (1) or (2) holds.*

(1) A $48\sqrt{2}\mu$-bridge of H_1 and H_2 can be found using rounded arithmetic in $O(\log m)$ time using m processors.

(2) Either $|H_1|/4$ vertices of H_1 or $|H_2|/4$ vertices of H_2 which does not contain the endpoints of the bridge of H_1 and H_2 can be found using rounded arithmetic in $O(\log m)$ time using m processors.

Proof: Construct $|H|/2$ edges by matching each pair of two vertices of H. Let E be the set of these edges. Find an edge e from E such that *medi* $= (Slop(e))_P$ is the mediam of $\{(Slop(e'))_P \mid e' \in E\}$. Denote the line passing through e as $l(e)$. Find a $16\sqrt{2}\mu$-support line of H whose slope is *medi* as follows. Compute set $A = (Abov(A, l(e)))_P$, the set of the vertices above line $l(e)$, and then find vertex $p \in H$ such that $(D(p, l(e)))_P = \max\{(D(p', l(e)))_P \mid p' \in A\}$ (Fig. 6). Let L be the line passing through p with slope *medi*. Including the absolute error causing by rounded arithmetic, L is a $16\sqrt{2}\mu$-support line of H. That is, if vertex u of H is above L, u is at most $16\sqrt{2}\mu$ away from L. We prove the lemma by considering the following two cases.

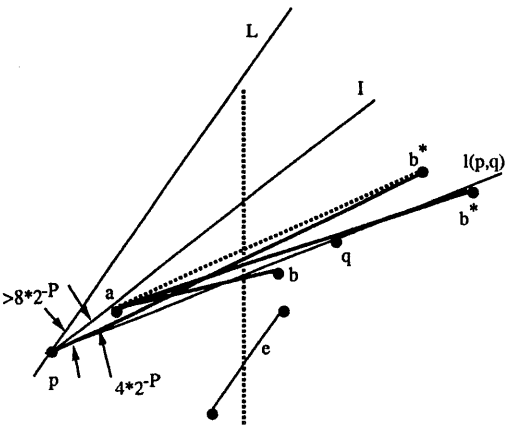

Figure 6: Vertex a: not an endpoint of the bridge

(Case 1) Vertex p belongs to H_1. Find vertex $q \in H_2$

such that $(Slop(pq))_P = \max\{(Slop(pq'))_P \mid q' \in H_2\}$ and determine whether $(\Theta(qp, L) \leq 12 \cdot 2^{-P})_P$.
(i) When $(\Theta(qp, L) \leq 12 \cdot 2^{-P})_P$, pq is a $48\sqrt{2}\mu$-bridge of H_1 and H_2. To show this see that any vertex under L is at most $16\sqrt{2}\mu$ away from pq, since $\Theta(qp, L) \leq 16 \cdot 2^{-P}$ holds from $(\Theta(qp, L) \leq 12 \cdot 2^{-P})_P$ by Lemma 2. For any point $g \in H$ above $l(p, q)$, if g is under L, $D(g, pq) \leq 2\sqrt{2}M \cdot 16 \cdot 2^{-P} = 32\sqrt{2}\mu$, and if g is above L, $D(g, pq) \leq D(g, L) + 2\sqrt{2}M \cdot 16 \cdot 2^{-P} \leq 48\sqrt{2}\mu$.

(ii) When $(\Theta(qp, L) > 12 \cdot 2^{-P})_P$, for any line segment $ab \in E$ if $a \in H_1$ and $(Slop(ab))_P \geq medi$ (note that there are at least $|H_1|/4$ such vertices), vertex a can not be an endpoint of the bridge of H_1 and H_2. We show this as follows. From conditions $(\Theta(qp, L) > 12 \cdot 2^{-P})_P$ and $(Slop(ab))_P \geq (Slop(L))_P$, we have $\Theta(qp, L) > 8 \cdot 2^{-P}$ and $Slop(ab) \geq Slop(L) - 4 \cdot 2^{-P}$, respectively. Let I be the line passing through p and $\Theta(pq, I) = 4 \cdot 2^{-P}$. See that $Slop(I) \leq Slop(L) - 4 \cdot \sqrt{2}\mu$ holds. Assume that a is an endpoint of the bridge ab^*. Obviously, $Slop(ab) \leq Slop(ab^*)$ holds. If b^* is under $l(p, q)$, a must be above $l(pq)$ else ab^* can not be the bridge. This means $Slop(ab) \leq Slop(ab^*) \leq Slop(pq) \leq Slop(I) < Slop(L) - 4 \cdot 2^{-P}$ which is a contradiction with $Slop(ab) \geq Slop(L) - 4 \cdot 2^{-P}$. If b^* is above $l(p, q)$, from the selection of q, b^* must under I, in the meanwhile, a must be above pb^* else ab^* can not be the bridge. This means $Slop(ab) \leq Slop(ab^*) \leq Slop(pb^*) \leq Slop(I) < Slop(L) - 4 \cdot 2^{-P}$. This is also a contradiction with $Slop(ab) \geq Slop(L) - 4 \cdot 2^{-P}$.
(Case 2) Vertex p belongs to H_2. Using the proof similar to that for Case 1, either we can find a $48\sqrt{2}\mu$-bridge of H_1 and H_2, or we can prove that for any line segment $ab \in E$, if $b \in H_2$ and $(Slop(ab))_P \leq medi$, b is not an endpoint of the bridge of H_1 and H_2.

The maximum and mediam computations can be executed in $O(\log m)$ time using m processors [10]. It is easy to see that other operations can be executed in $O(1)$ time using m processors. ∎

Lemma 7 *Let H, H_1 and H_2 be the polygonal chains in Lemma 6. A $48\sqrt{2}\mu$-bridge of H_1 and H_2 can be computed Using rounded arithmetic in $O(\log^2 m)$ time using m processors.*

Proof: Using Lemma 6, we can either find a $48\sqrt{2}\mu$-bridge of H_1 and H_2, or delete $|H_1|/4$ points of H_1 or $|H_2|/4$ points of H_2 which does not contain the endpoints of the bridge of H_1 and H_2. After using Lemma 6 $O(\log n)$ times we can find a $48\sqrt{2}\mu$-bridge of H_1 and H_2 finally. Therefore, a $48\sqrt{2}\mu$-bridge can be computed in $O(\log^2 m)$ time using m processors. ∎

The following lemma suggests a divide-and-conquer method for constructing an approximate convex hull of H.

Lemma 8 *Let $H = (u_1, u_2, \ldots, u_m)$ be the polygonal chain in Lemma 6, and $b = u_s u_t$ be a $48\sqrt{2}\mu$-bridge of $H_1 = (u_1, u_2, \ldots, u_{m/2})$ and $H_2 = (u_{m/2+1}, u_{m/2+2}, \cdots, u_m)$. Let F_1 be a convex $64\sqrt{2}\mu$-hull of $H^1 = (u_1, u_2, \ldots, u_s)$ containing vertices u_1 and u_s, and F_2 be a convex $64\sqrt{2}\mu$-hulls of $H^2 = (u_t, u_{t+1}, \ldots, u_m)$ containing vertices u_t and u_m. Given bridge b, F_1 and F_2, a convex $64\sqrt{2}\mu$-hull of H which contains vertices u_1 and u_m can be computed using rounded arithmetic in $O(\log n)$ time using n processors.*

Proof: Compute $B_1 = \{u \mid (\Theta(u_s u_t, u u_s) > 4 \cdot 2^{-P})_P$ and $u \in F_1\}$. For each vertex u of B_1, since $(\Theta(u_s u_t, u u_s) > 4 \cdot 2^{-P})_P$ implies $\Theta(u_s u_t, u u_s) > 0$, u is under line $l(u_s, u_t)$. Find vertex u_a of B_1 such that its x coordinate is minimum (Fig. 7). Similarly, compute $B_2 = \{u \mid (\Theta(u_t u, u_s u_t) > 4 \cdot 2^{-P})_P$ and $u \in F_2\}$ and find vertex u_b of B_2 such that its x coordinate is maximum. Let u^* be the right neighbour of u_a in F_1 and u^{**} be the left neighbour of u_b in F_2. Let $\overline{F_1}$ be the convex polygon by deleting the vertices of F_1 which are in the right of u^* and $\overline{F_2}$ be the convex polygon by deleting the vertices of F_2 which are in the left of u^{**}. We show that convex polygon F consisting of $\overline{F_1}$, $\overline{F_2}$, $u^* v^{**}$ is a convex $62\sqrt{2}\mu$-hull of H. First, we prove F that is convex. Each vertex u of F_1 except u^* is under line $l(u_s, u_t)$. and each vertex of u of F_2 except u^{**} is under line $l(u_s, u_t)$. Therefore, line seqment $u_a u_b$ is under line $l(u_s, u_t)$. Together with the conditions that F_1 and F_2 are convex we conduct that F is convex. Obviously, F contains vertices u_1 and u_m.

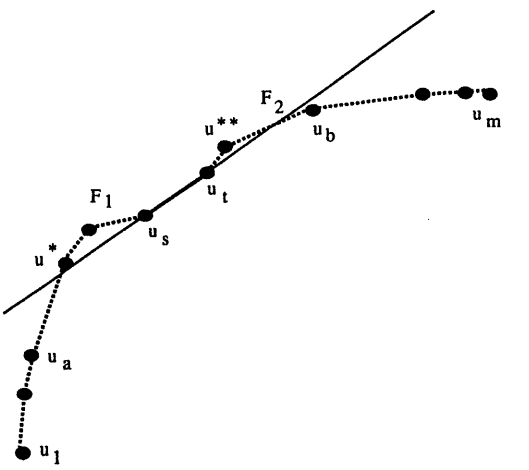

Figure 7: Robust method for marging two convex hulls with an approximate bridge

Next we prove that F is a $64\sqrt{2}\mu$-hull of H. For vertex u of H which is above F, there are two cases.
(Case 1) $X(u_1) < X(u) < X(u^*)$ or $X(u^{**}) < X(u) < X(u_m)$. Since F_1 and F_2 are $64\sqrt{2}\mu$-hulls of H_1 and H_2, respectively. u is at most $64\sqrt{2}\mu$ away from $\overline{F_1}$ or $\overline{F_2}$.
(Case 2) $X(u^*) < X(u) < X(u^{**})$. Since $(\Theta(u_s u_t, u_a u_s) \leq 4 \cdot 2^{-P})_P$ and $(\Theta(u_t u_b, u_s u_t) \leq 4 \cdot 2^{-P})_P$ imply $-4 \cdot 2^{-P} \leq \Theta(u_s u_t, u_a u_s) \leq 8 \cdot 2^{-P}$ and $-4 \cdot 2^{-P} \leq \Theta(u_t u_b, u_s u_t) \leq$

$8 \cdot 2^{-P}$, respectively, any point on line $l(u_s, u_t)$ is at most $16\sqrt{2}\mu$ ($= 2\sqrt{2}M \cdot 8 \cdot 2^{-P}$) away from line $l(u^*, u^{**})$, in the meanwhile, $u_s u_t$ is a $48\sqrt{2}\mu$-bridge. Therefore, vertex u is at most $64\sqrt{2}\mu$ away from segment $u^* u^{**}$. ∎

The following algorithm MakeCH(H) constructs a convex $64\sqrt{2}\mu$-hull of H using Lemma 8, where H is a monotonic polygonal chain whose x and y coordinates are in increasing order.

Algorithm MakeCH(H)

(Step 1) If $|H| \leq 3$ return H. This completes the algorithm. Else do the following steps.

(Step 2) Let $H = (u_1, u_2, \ldots, u_m)$. Divide H into two equally sized polygonal chains H_1 and H_2 such that $H_1 = (u_1, u_2, \ldots, u_{m/2})$ and $H_2 = (u_{m/2+1}, u_{m/2+1}, \ldots, u_m)$. Use Lemma 7 compute $\overline{u_s u_t}$, a $48\sqrt{2}\mu$-bridge of H_1 and H_2.

(Step 3) For $H^1 = (u_1, u_2, \ldots, u_s)$ and $H^2 = (u_t, u_{t+1}, \ldots, u_m)$, the subchains of H_1 and H_2, respectively, recursively construct CH^1 and CH^2, the convex $64\sqrt{2}\mu$-hulls of H^1 and H^2 respectively, in parallel.

(Step 4) construct a convex $64\sqrt{2}\mu$-hull of H from $u_s u_t$, CH^1 and CH^2 by Lemma 8. ∎

Theorem 6 *Algorithm MakeCH(H) constructs a convex $64\sqrt{2}\mu$-hull of H using rounded arithmetic in $O(\log^3 m)$ time using m processors.*

Proof: The correctness of the algorithm is proved in Lemma 8. The algorithm constructs a convex $64\sqrt{2}\mu$-hull of H using divide-and-conquer technique. Since each recursive level runs in $O(\log^2 m)$ time using m processors, the algorithm runs in $O(\log^3 m)$ time using m processors. ∎

Theorem 7 *A convex $64\sqrt{2}\mu$-hull of a set of n points can be constructed using rounded arithmetic in $O(\log^3 n)$ time using n processors.* ∎

References

(1) A. Aggarwal, B. Chazelle, L. Guibas, C. O'Dunlaing, and C. Yap: Parallel computational geometry. Algorithmica, 3, 293-327, 1988.

(2) M. J. Atallah and M. T. Goodrich: Efficient parallel solutions to some geometric problems. Journal of Parallel and Distributed Computing, 3, 492-507, 1986.

(3) M. J. Atallah and M. T. Goodrich: Parallel algorithms for some functions of two convex polygons. Algorithmica, 3, 535-548, 1988.

(4) W. Chen,K. Nakano, T. Masuzawa, and N. Tokura: Optimal parallel algorithms for finding the convex hull of a sorted point set, Trans. IEICE, J74-D-I, 12, 814-825, 1991.

(5) W. Chen, K. Nakano, T. Masuzawa and N.Tokura: A parallel method for the prefix convex hulls problem. IEICE Trans. Fundamentals, E77-A, 10, 1675-1683, 1994.

(6) R. Cole and M. T. Goodrich: Optimal parallel algorithms for polygon and point set problems. In Proc. of the 4th Annual ACM Symposium on Computational Geometry, 1988.

(7) D. Dobkin and D. Silver: Recipes for geometry and numerical analysis—part I: An empiricial study. In Proc. of the 4th Annual ACM Symposium on Computational Geometry, 93-105, 1988.

(8) P-O. Fjällström,J. Katajainen, C. Levcopoulos and O. Petersson: A sublogarithmic convex hull algorithm. Bit, 30, 378-384, 1990.

(9) S. Fortune: Stable maintenance of point set triangulations in two dimensions. In Proc. of the 30th Annual Symposium on Foundations of Computer Science, 494-499, 1989.

(10) A. Gibbons and W. Rytter: Efficient parallel algorithms. Cambridge University Press, 1988.

(11) D. H. Greene and F. F. Yao: Finite-resolution computational geometry. In Proc. of the 27th IEEE Symposium on Foundations of Computer Science, 143-152, 1986.

(12) L. Guibas, D Salesin and J. Stolfi: Epsilon geometry: building algorithms from imprecise computations. In Proc. of the 5th Annual ACM Symposium on Computational Geometry, 208-217, 1989.

(13) L. Guibas, D Salesin and J. Stolfi: Constructing strongly convex approximate hulls with inaccurate primitives. Algorithmica, 9, 534-560, 1993.

(14) C. Hoffman: The problem of accuracy and robustness in geometric computation. Computer, 22, 31-42, 1989.

(15) Z. Li and V. J. Milenkovic: Constructing strongly convex hulls using exact or rounded arithmetic. In Proc. the 6th Annual ACM Symposium on Computational Geometry, 35-243, 1990.

(16) V. J. Milenkovic: Verifiable implementations of geometric algorithms using finite precision arithmetic. Artificial Intelligence, 37, 377-401, 1988.

(17) V. J. Milenkovic: Calculating approximate curve arrangements using rounded arithmetic. In Proc. the 5th Annual ACM Symposium on Computational Geometry, 197-207, 1989.

(18) V. J. Milenkovic: Double precision geometry: a general technique for calculating line and segment intersections using rounded arithmetic. In Proc. of the 30th Annual Symposium on Foundations of Computer Science, 500-505, 1989.

(19) T. Ottmann, F. Thiemt and C. Ullrich: Numerical stability of geometric algorithms. In Proc. the 3th Annual ACM Symposium on Computational Geometry, 119-125, 1987.