M.TECH. (COMPUTER SCIENCE) DISSERTATION SERIES

GUARD ZONE PROBLEM

A DISSERTATION SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE OF MASTER OF TECHNOLOGY IN COMPUTER SCIENCE

BY

J.SHIVAKUMAR

UNDER THE SUPERVISION OF

DR. BHARGAB B. BHATTACHARYA

AND

MR. SUBHAS. C. NANDY

INDIAN STATISTICAL INSTITUTE LIBRARY
CALCUTTA

INDIAN STATISTICAL INSTITUTE
203, BARRACKPUR TRUNK ROAD,
CALCUTTA - 700035.

ACKNOWLEDGEMENTS

I herewith express my deepest gratitude to my guides, Dr. Bhargab Bhattacharya and Mr. Subhas.C.Nandy for their valuable guidance and encouragement throughout the period of this dissertation. I would also like to thank my friends, who were always ready to help, especially in moments of crisis.

1.INTRODUCTION

In VLSI layout design where the major aim is to pack more and more circuitry in minimum chip area, the degree of closeness between two different modules is limited by the inductive effects of one circuit on another. It is thus required to separate different circuit components by a predefined distance δ determined by the design rules of the technology. In this dissertation we introduce the concept of a guard zone around the modules which will be used to prevent the violation of design rules.

The guard zone (of width δ) of a polygon is defined as a closed region consisting of straight lines and circular arcs bounding the polygon so that there exist no pair of points (p, q), where p lies on the polygon and q lies on the boundary of the guard zone, such that the euclidian distance between p and q is less than d. In this dissertation, our objective is to find the guard zone of a simple polygon having minimum area. Fig 1 demonstrates the guard zone around an isothetic polygon.

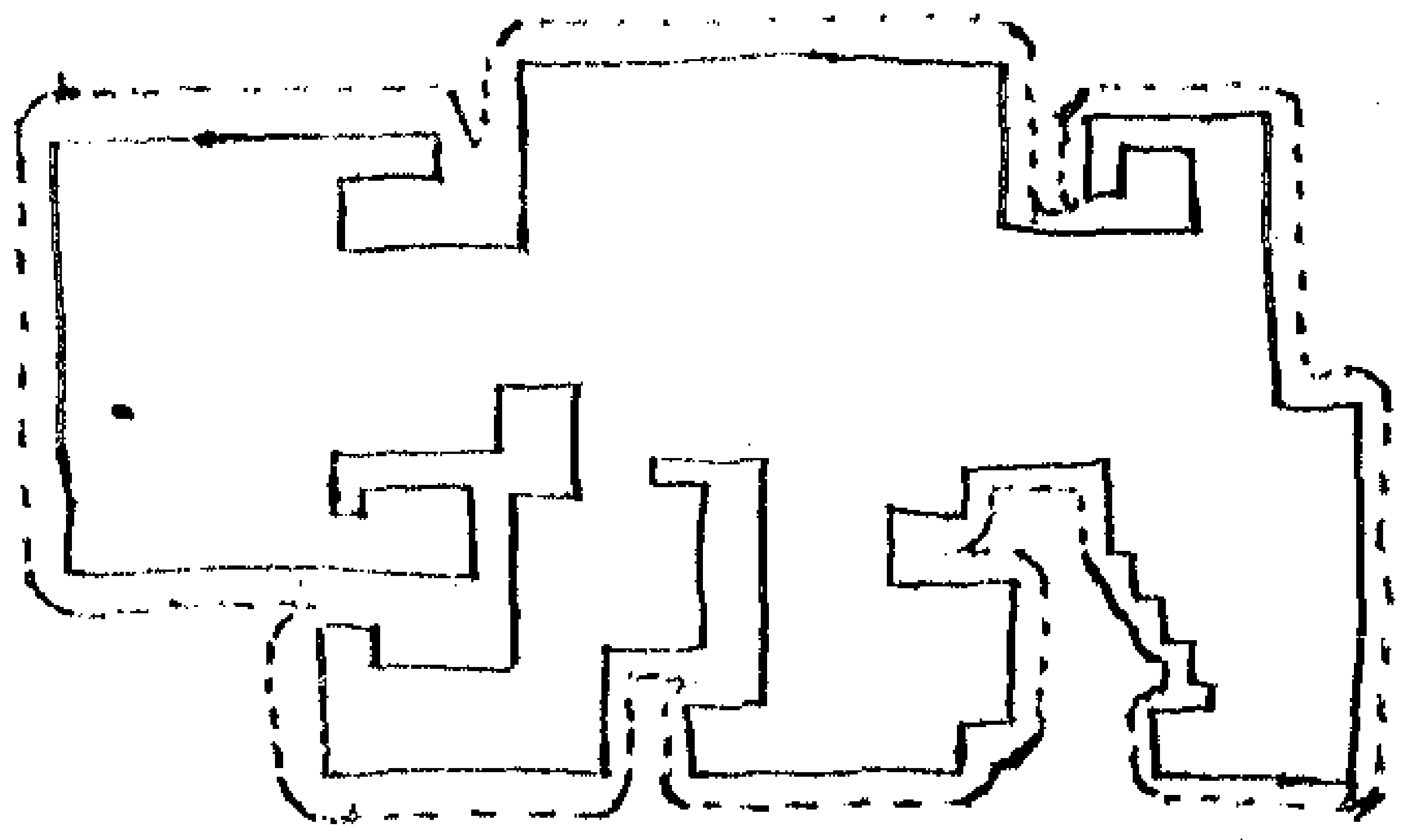


FIG 1

In addition to VLSI design the concept of guard zone has wide range of applications in graphics, machine tools and defence to name a few.

2. PROBLEM DESCRIPTION

The guard zone problem is an optimization problem of computational geometry, defined as follows.

DEFINITION: A guard zone G of a simple polygon P is a closed region bounded by straight line segments and circular arcs enclosing the polygon P and satisfying the following constraints.

1. For every point x on the boundary of G there does not exist any point p on the boundary of P such that dist(x,p) < δ for a given positive value of δ.

2. Area (G) is a minimum.

To develop the algorithm for finding the guard zone of a polygon, we now introduce the following concepts :

In our problem we consider a polygon as a sequence of vertices. By **clockwise traversal**, we mean an ordering of the vertices such that if one travels along the outer boundary of the polygon, then the interior of the polygon will lie to his right throughout.

During a clockwise traversal of the polygon, if we encounter a sequence of consecutive vertices (a, b, c), then vertex b is a **convex corner** provided there exists at least one point p ($\neq$ b) on ab and a point q ($\neq$ b) on bc such that the straight line segment pq lies completely inside the polygon. Otherwise b is said to be a **concave corner**. In the above two cases the angle abc is said to be a right/left turn respectively.

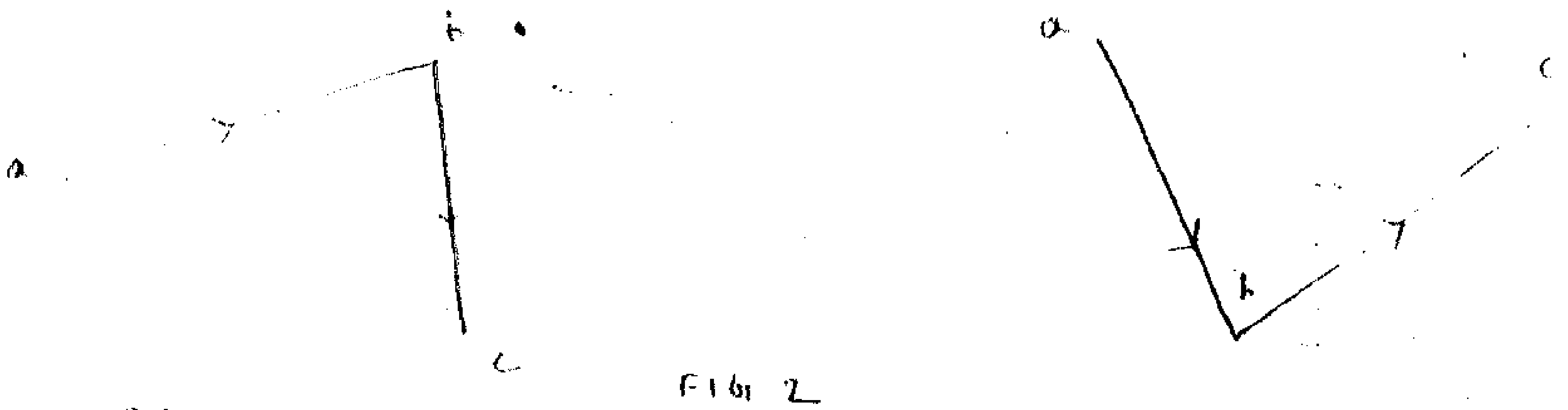


FIG 2

RIGHT TURN (CONVEX CORNER) (a)

(b) LEFT TURN (CONCAVE CORNER)

The **convex hull** [2] of a polygon P is the smallest convex polygon bounding the given polygon P completely. Note that the set of vertices of Q is a subset of the set of vertices of P, but all the edges of Q may not correspond to that of P.

Remark: If P is convex then all the edges of Q are edges of P and vice versa.

Let α and β be the two consecutive vertices of Q such that there exists a sequence of vertices $\Gamma = \{ \gamma 1, \gamma 2 ... \gamma k \}$ $(k \geq 1)$ such that the vertices of Γ lie between α and β in a clockwise traversal. Then the sequence of vertices in Γ forms a **notch** with lid (α, β).

Result: Let $\varkappa$ be the total number of edges of Q and $\varkappa^*$ be the number of edges which are aligned to the edges of P, then the total number of notches is $(\varkappa - \varkappa^*)$.

3. SOME IMPORTANT OBSERVATIONS

We can classify the edges of a polygon into the following eight types depending on their orientations.

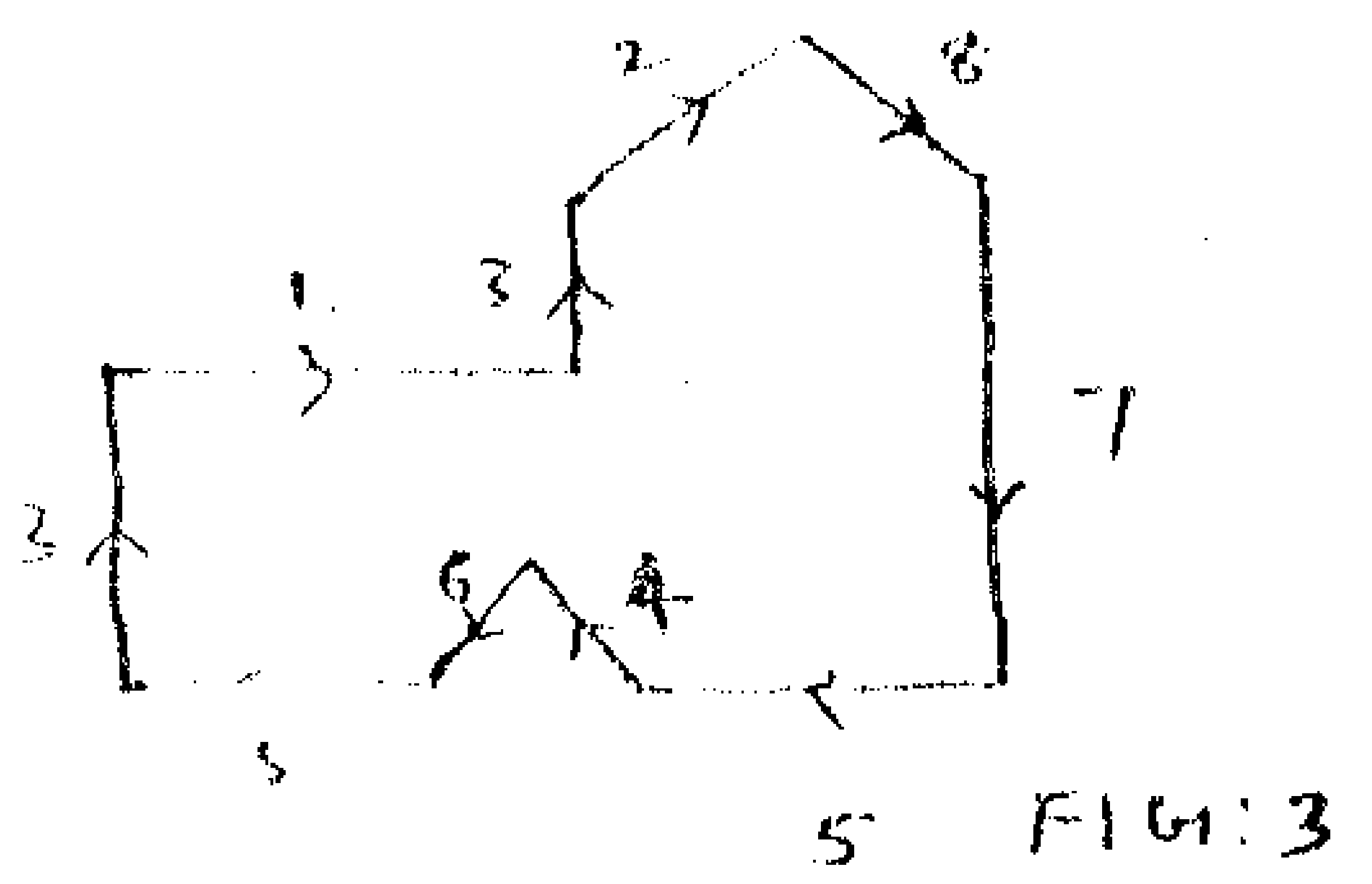


FIG: 3

THE NUMBERS SHOWN NEAR AN EDGE INDICATES ITS TYPE

An edge (α, β) is said to be of type-1 *(horizontal right)* if the starting vertex α and the ending vertex β have the same y-co-ordinates and the x-co-ordinate of α is less than that of β.

An edge (α, β) is said to be of type-2 (*with slope* $\in$ [0, $\pi/2$]) if both the x- and y-co-ordinates of the starting vertex α are less than those of the ending vertex β.

An edge (α, β) is said to be of type-3 (*vertical up*) if the vertices α and β have the same x-co-ordinate and the y-co-ordinate of α is less than that of β.

An edge (α, β) is said to be of type-4 (*with slope* $\in [\pi/2, \pi]$) if the x-co-ordinate α is greater than that of β but the y-co-ordinate of α is less than that of β.

An edge (α, β) is said to be of type-5 (*horizontal left*) if α and β have the same y-co-ordinates but the x-co-ordinate of α is greater than that of β.

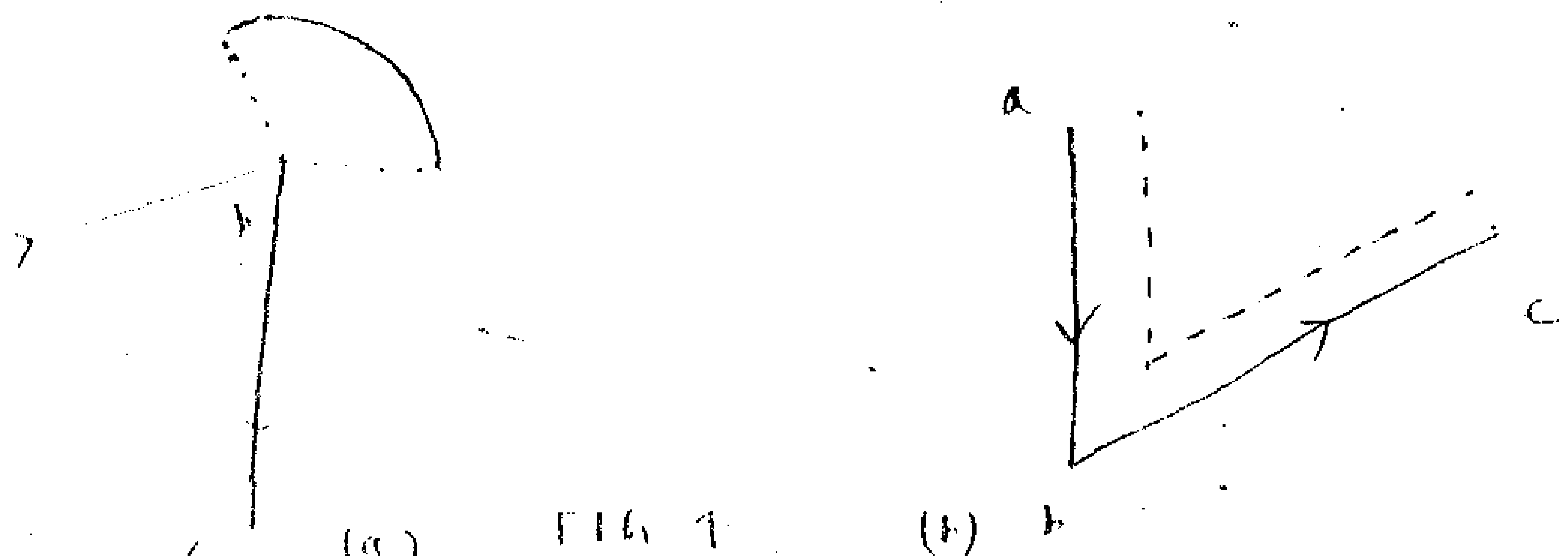
An edge (α, β) is said to be of type-6 (*with slope* $\in [\pi, 3\pi/2]$) if the x-and y-co-ordinates of α are both greater than that of β.

An edge (α, β) is said to be of type-7 (*vertical down*) if the y co-ordinate of vertex α is greater than that of thevertex β but their x-co-ordinates are same.

An edge (α, β) is said to be of type-8 (with slope $\in [3\pi/2, 2\pi]$) if the x-co-ordinate of α is less than that of β but the y-co-ordinate of β is less than that of α.

The above types of edges are exhaustive and if the polygon is oriented in a clockwise direction, the guard zone will lie to the left of all types of edges. In appendix 1, we show the different edge types (ab) and present the analytical techniques used in the computation of the extents (m,n) of the portion of the guard zone running parallel to it.

The shape of the guard zone around a convex corner is a circular arc of radius δ and it is a sharp turn around a concave corner. (see fig below)



(a) FIG 1 (b)

A CONVEX CORNER

AROUND A CONCAVE CORNER

LEMMA 1: The length of a circular arc in a guard zone can be no greater than $\pi\delta$ where δ is the guard zone distance.

PROOF:

Consider the guardzone around the convex corner 'A' of figure.

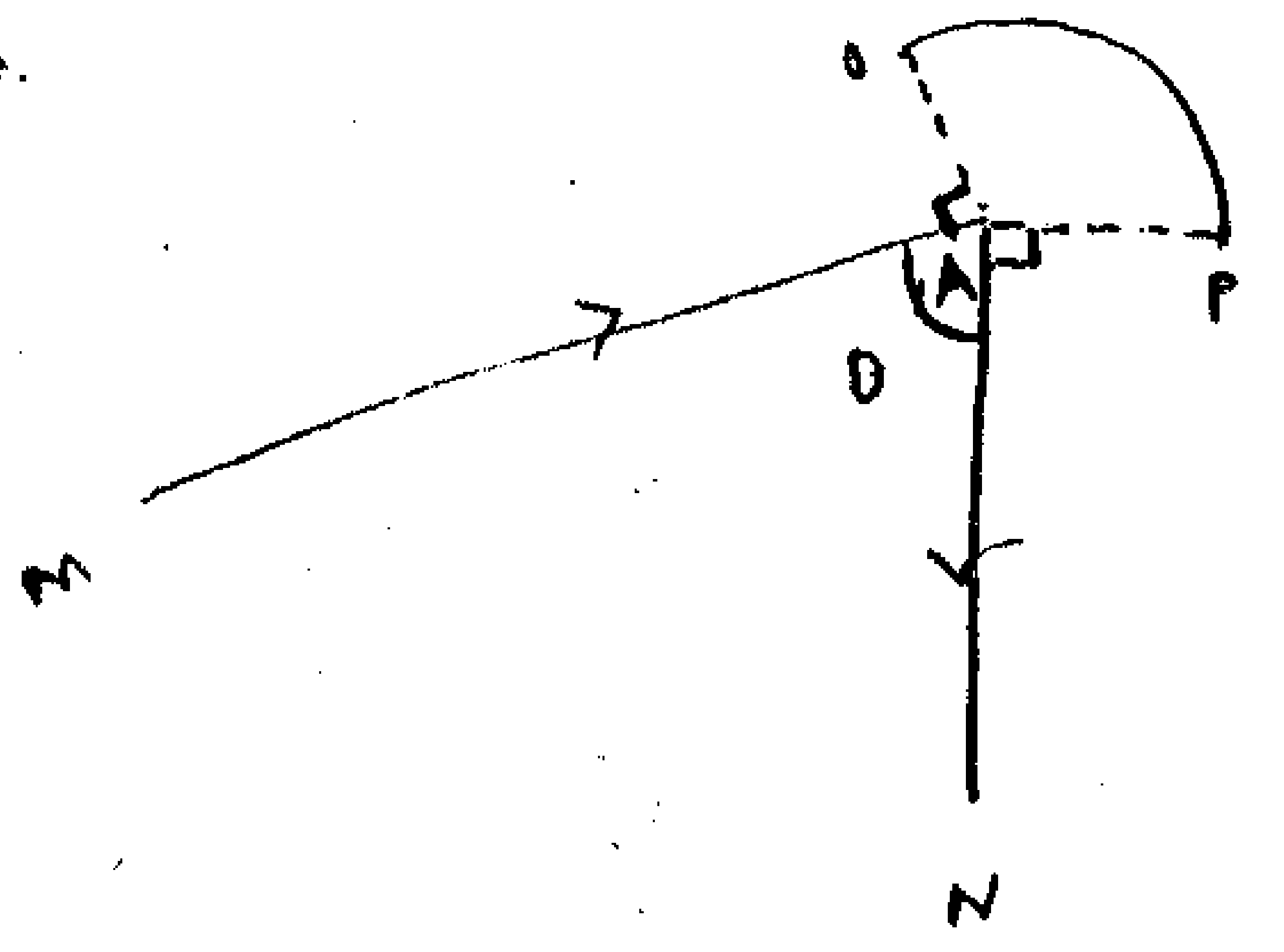


FIG 5

Let $\angle$ MAN be θ.

If OA $\perp$ MA and PA $\perp$ NA then $\angle$ OAP =360-θ-180 = 180- θ.

The minimum value θ can take is 0 and so $\angle$ OAP can atmost be 180.

It follows that the length of the circular arc can be no greater than $\pi\delta$ where δ is the guard zone distance.

A guard zone enters a notch if the lid of the notch is of length > 2*δ. The amount of penetration of the guard zone inside the notch and the shapes it can assume depend on the shape of the notch. Typical examples are shown in the figures below.

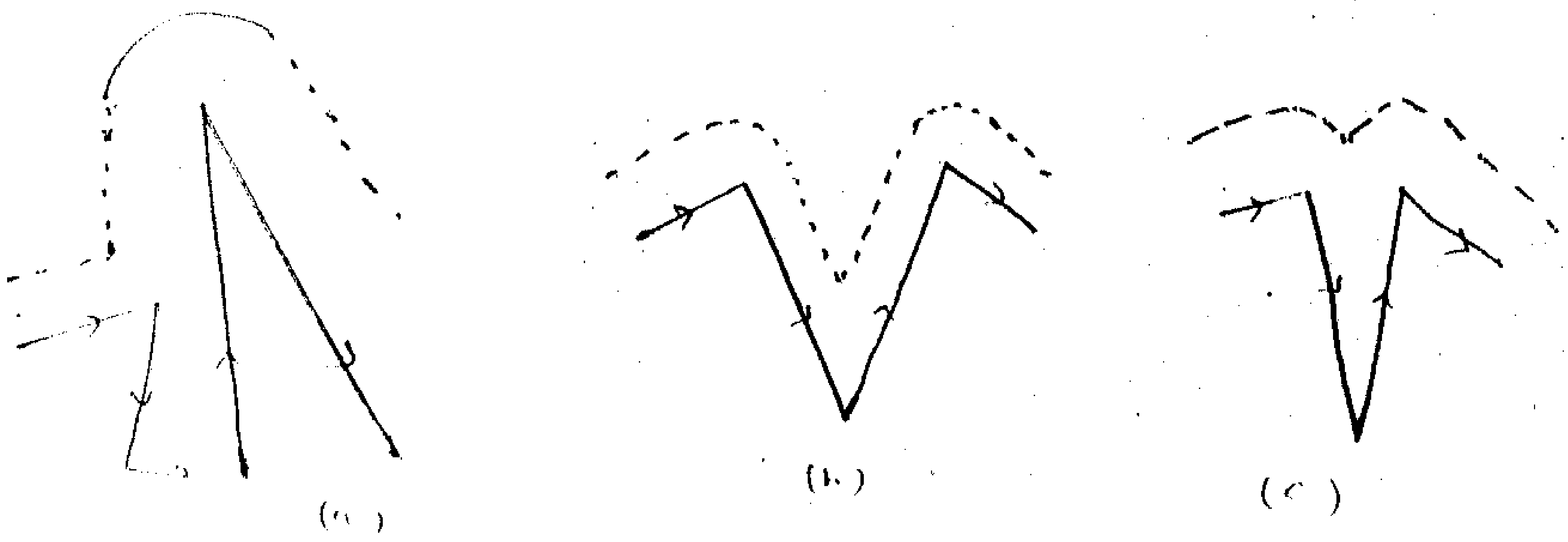


FIG 6

4.METHOD

4.1 DATA STRUCTURES: We propose the following data structure to hold a polygon and its guard-zone:

A doubly-linked circular list of the polygonal vertices is required. Each vertex in the list is characterized by its cartesian co-ordinates. With each vertex, we associate two fields start and end which are two points around it. The figure below shows a polygon and its guard zone along with the proposed data structure. It can be seen that the guard zone is efficiently represented in the data structure. Further we associate with each vertex a pointer next_hull_vertex which, if the current vertex is a convex hull vertex, would point to the next hull vertex. Our data structure is thus equipped to store the convex hull of the polygon also.

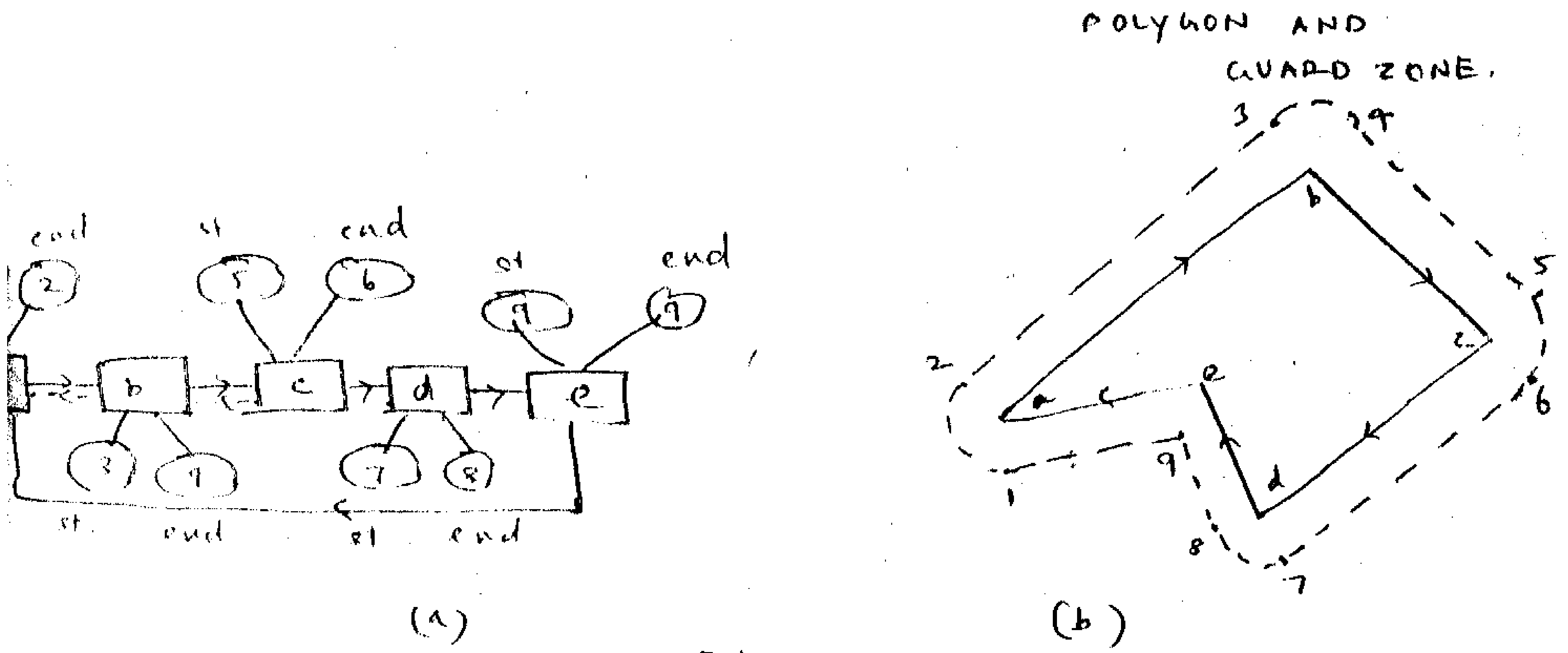


Fig 7

Plot of the guard zone

Here we describe the method of plotting the guard zone from the data structure prescribed above. The polygon can be drawn by drawing straight line edges between the successive vertices in the linked list. To construct the guard zone we adopt the following procedure:

1) For every vertex do the following :

a) If the start and end fields are different, then with the vertex co-ordinates as the centre, draw a circular arc oriented in the clockwise direction from the start point to the end point. The radius of this circular arc will be the guard zone distance δ.

b) Draw a straight line edge from the end co-ordinate of the vertex to the start co-ordinate of the next vertex in the list.

In the figure below we show a non-convex polygon and its guard zone. We also show how the data structure can hold the guard zone. It is thus clear that the proposed data structure can hold guard zones of arbitrary polygons.

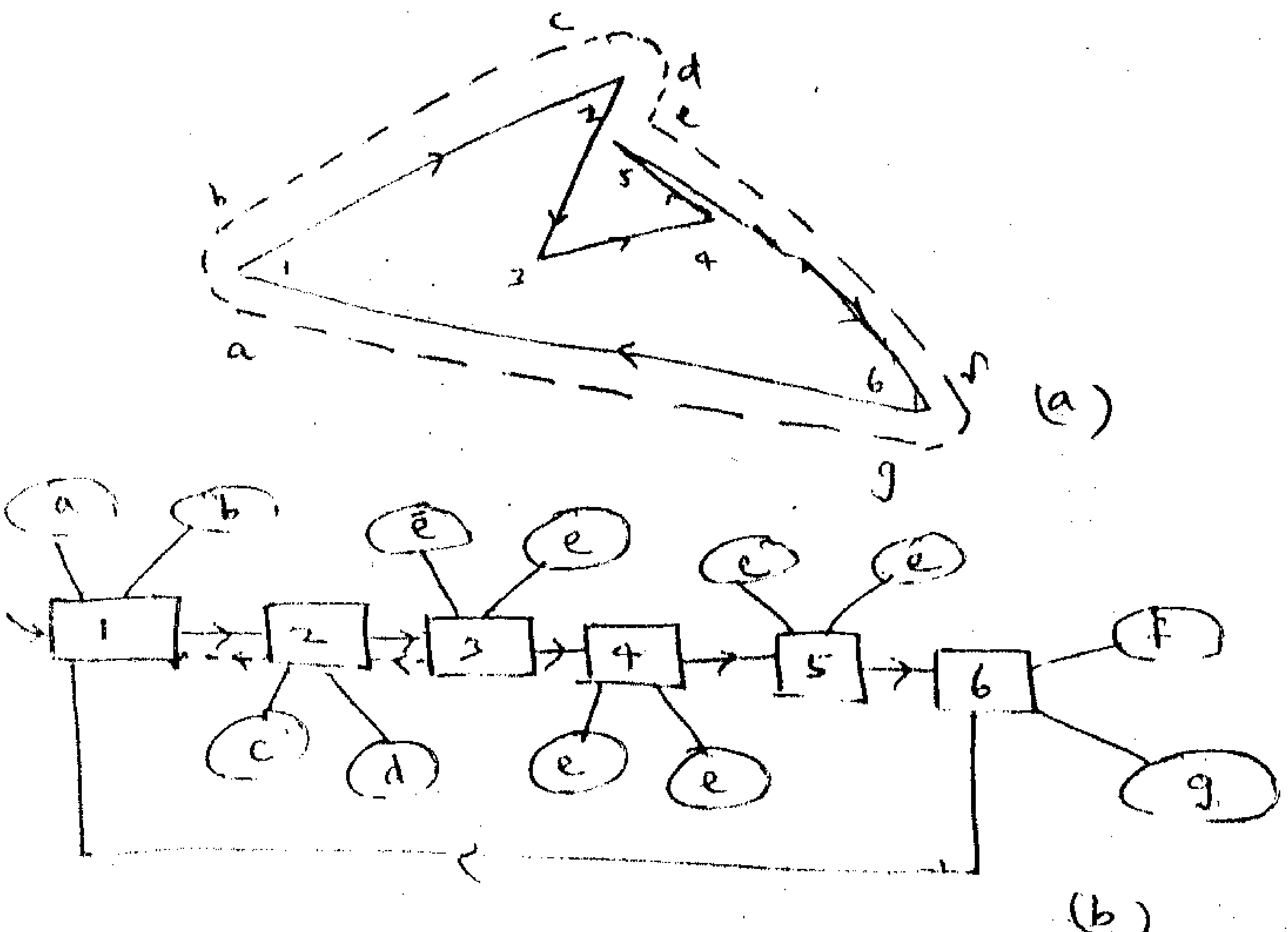


FIG 8

4.2 THE CONVEX GUARD ZONE OF A POLYGON:

DEFINITION: The convex guard zone of a polygon is the guard zone of its convex hull. The figure below shows a non-convex polygon and its convex guard zone.

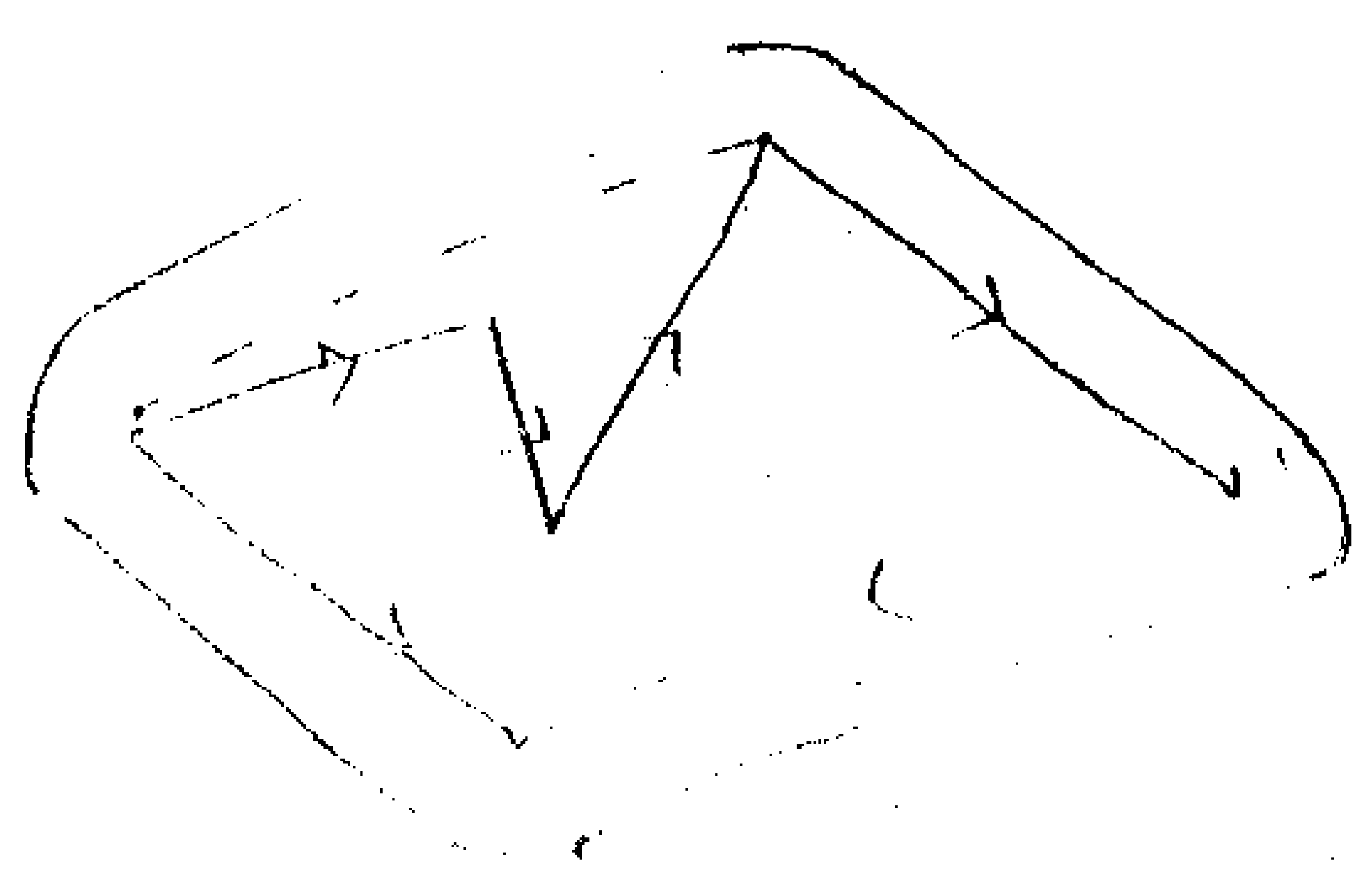


FIG 9

METHOD:

1) Store the polygon in the prescribed data structure.

2) For the above polygon find the convex hull using GRAHAM'S SCAN algorithm. (see next side heading).

3) For each edge (a,b) of the hull do the following.

a) Find the type of edge it is. (OBSV 1)

b) For the edge (a,b) find the extent (m,n) of the guard zone running parallel to it. (OBSV 2)

c) Set the end field of a to m.

d) Set the start field of b to n.

e) If there are any vertices of the polygon lying between vertices a and b for all these vertices set the start and end fields to m.

4) Plot the guard zone from the data structure using the method outlined earlier.

/* Note that in steps 3c and 3d we set the extents of the lines and arcs comprising the guard zone. In step 3e we process outside a notch. We set the data structure fields of the vertices inside the notch in such a way that the guard zone does not enter the notch. */

GRAHAM'S SCAN ALGORITHM [2]

1. Start at a vertex of the polygon that is assured to be a hull vertex. The leftmost highest vertex would be a suitable vertex.

2. Starting from the above vertex, we examine triples of consecutive vertices in clockwise order. If (ABC) is such a triple,

2.1 if ∠ ABC is convex we advance the scan and check for the next triple (BCD) where D is the vertex next to C.

2.2 If ∠ ABC is a left turn, we eliminate vertex B, and check for the triple (ZAC) where Z is the non-eliminated vertex prior to A.

It is an immediate consequence of convexity that in traversing a convex polygon in clockwise order we make only right turns.

The GRAHAM's SCAN method gives us the convex hull of a polygon in one scan around the polygon.

The complexity of finding the convex hull of a simple polygon using the above algorithm is O(n) time, where n is the number of vertices of the polygon. The convex guard zone can be formed in one scan around the convex hull. Thus the first phase of our problem, of finding the convex guard zone, can be accomplished in linear time.

DETECTION OF NOTCHES: It follows from the above method that we can detect a notch in the polygon in the following manner. If there exists an edge(A,B) of the convex hull that is not an edge of the polygon, then (A,B) is the lid of a notch. All the vertices of the polygon including and lying between A and B in a clockwise scan form the notch.

The area enclosed by the convex guard zone can be reduced further if we generate the general guard zone of the polygon. The general guard zone is formed when we allow the guard zone to enter into notches having sufficient width.

4.3 GUARD-ZONE INSIDE A NOTCH

We proceed now to generate the guard zone within a notch.

We explain below a method to generate the guard zone inside a notch. This method takes $O(n^2)$ time but is very space efficient. It builds the guard zone directly from the data structure proposed earlier. The method is outlined below in stages:

```
Step 1. For every vertex $v$ of the notch do
/* let $u$ be the vertex preceding $v$ and $w$ be the vertex next to $v$
in the list of vertices */
  begin
      Construct the portion of the guard zone part parallel to
      $(u,v)$ and call it $(m,n)$.
      Construct the portion of the guard zone part parallel to
      $(v,w)$ and call it $(o,p)$.
      if $v$ is blunt and if the line segments $(m,n)$ and $(o,p)$
      intersect at $x$,
          set the start and end points of the arc around the
          vertex $v$ to $x$. /* The arc length is zero */
      else
          Set the start and end points of the arc around vertex
          $v$ to $n$ and $o$ respectively.
  end;
```

```
Step 2. for every vertex $v$ of the notch do
    begin
        if vertex $v$ is sharp, find the first intersection $x$ of
        the already plotted guard zone with the arc around
        $v$.
            If the intersection $x$ exists then delete all
            portions of the guard zone lying beyond the
            intersection point.
        find the first intersection point of the already
        plotted guard zone with the guard zone line $(o,p)$
        parallel to $(v,w)$ .
            If the intersection point exists then delete the
            portion of the guard zone after the intersection.
    end.
```

Once the above steps are accomplished for each of the vertices of the notch our data structure would contain the precise guard-zone .We can plot it now using the method described earlier.

We can argue that the current guard zone encloses minimum area in the following manner. Being a closed curve any attempt to further minimize the area would require compressing the guardzone. According to our method of construction, this attempt would cause a design rule violation. Thus our guard zone bounds the least area possible.
The method of finding intersections are explained later in Appendix 2.

We next try to minimize the time the algorithm takes to construct the guard zone within a notch. Here we assume that the lid of a notch is horizontal. If it is not we can rotate the notch so that its lid becomes parallel to the horizontal axis. We generate the guard zone inside the notch using the method outlined, and then rotate the notch back to its original orientation.
Here in order to find the guard-zone inside a notch we first slice the notch into maximal empty horizontal strips. It is mentioned earlier that we align the face of the notch with the horizontal axis. For the other vertices of the polygon inside the notch, we may assume that no two verties have the same y-co-ordinates with respect to the reorientation of the notch. In the case any contradiction arises, we resolve it by adding a very small positive quantity e to any of them. Thus a slice is defined as follows:

Definition: A slice inside a notch of the polygon is an empty trapezium or an empty triangle whose two non-horizontal sides are arms of the notch and the horizontal sides correspond to the vertices of the notch. The figure below shows a sliced notch.

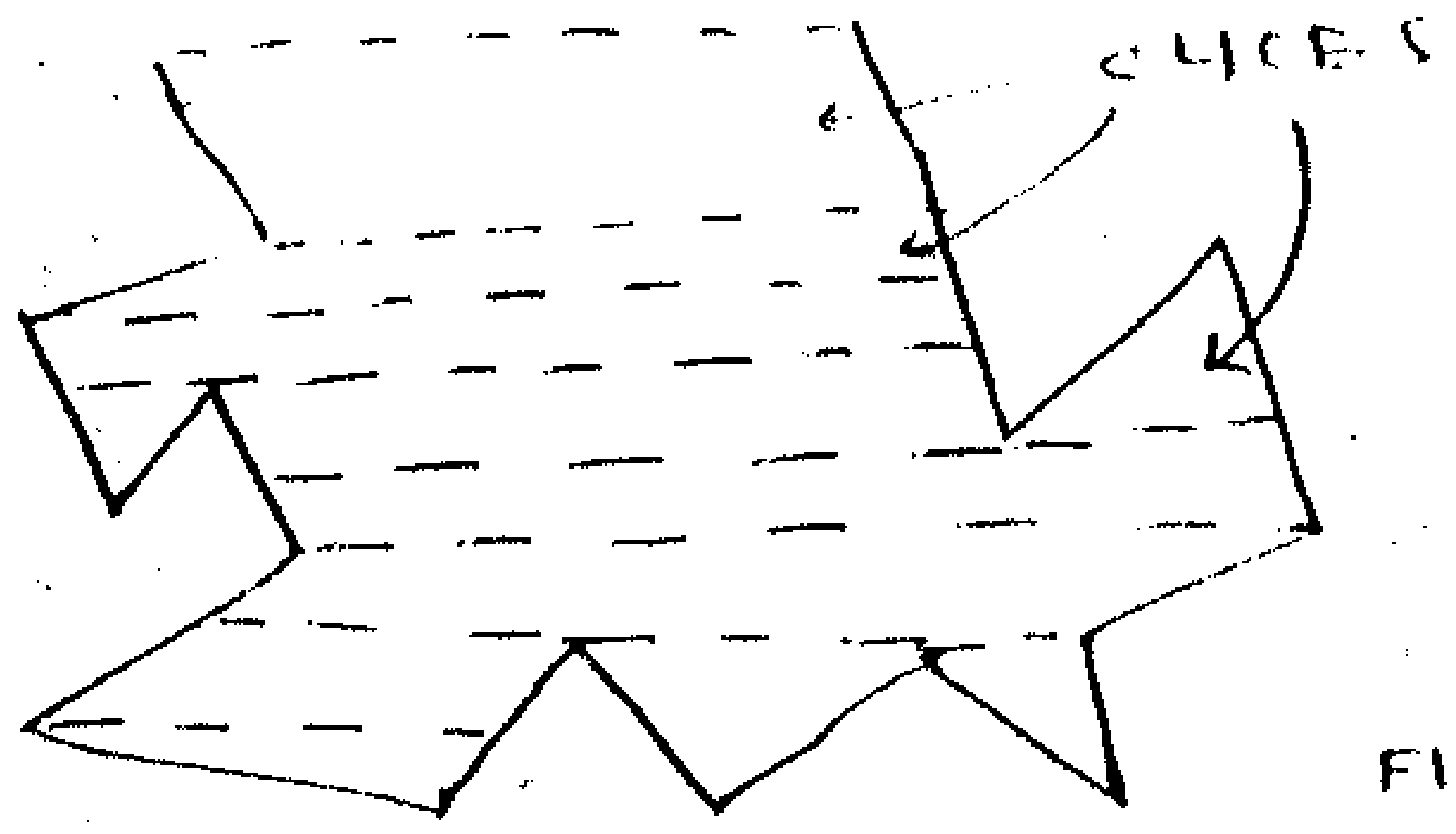


FIG 10

Now we describe certain observations of the slices based on which we describe the method of slicing a notch. In the next section we shall describe the method of finding the guard-zone inside the notch using the slicing structure.

1. A **slice_line passes** through a single vertex of the polygon. The vertex (υ) may be at one end of the slicing line, (fig below) or at the middle of the slicing line (fig below).

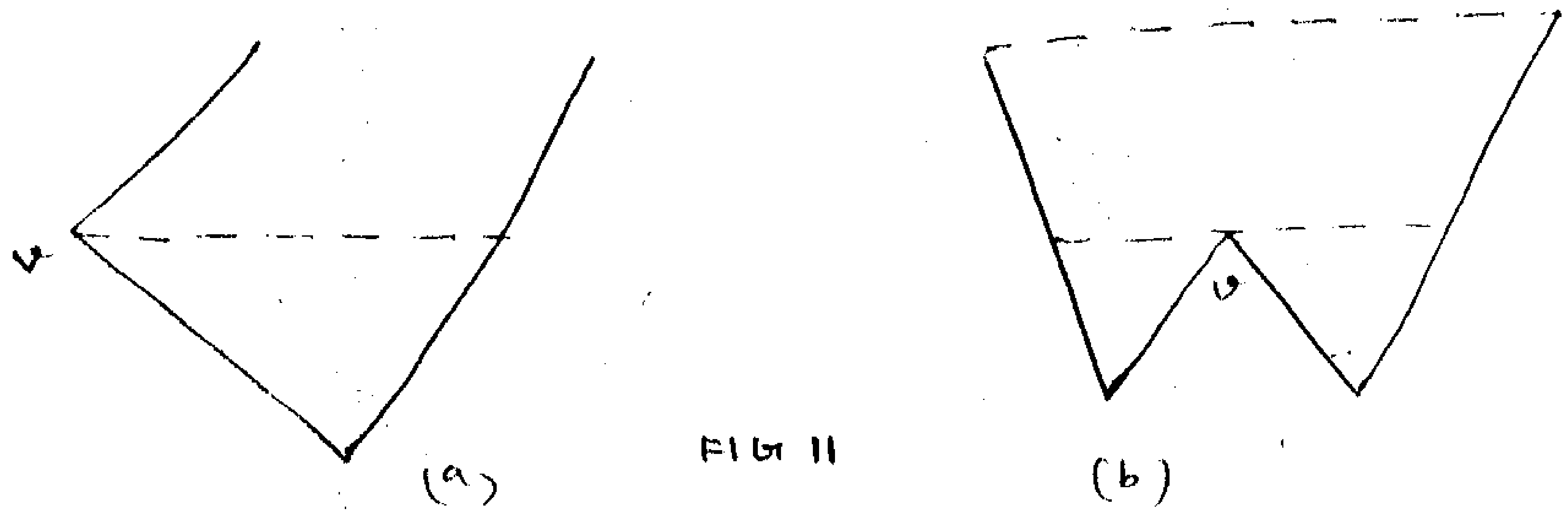


FIG 11

If the guard-zone **enters a slice** it **passes** through it as a sequence of straight lines and circular arcs. The figures below depict different shapes the guardzone can take inside a slice. Such a sequence of contiguous parts of the guard zone is referred to as a chain.

2.If the guard zone of a notch enters a slice along the top boundary it will surely exit from the slice along the top boundary. Similarly, if it exits from a slice along the bottom boundary, it will enter it again along the bottom boundary.

3. At most four chains of the guardzone may be present in a slice. The figures below show examples of slices containing one, two, three, four chains.

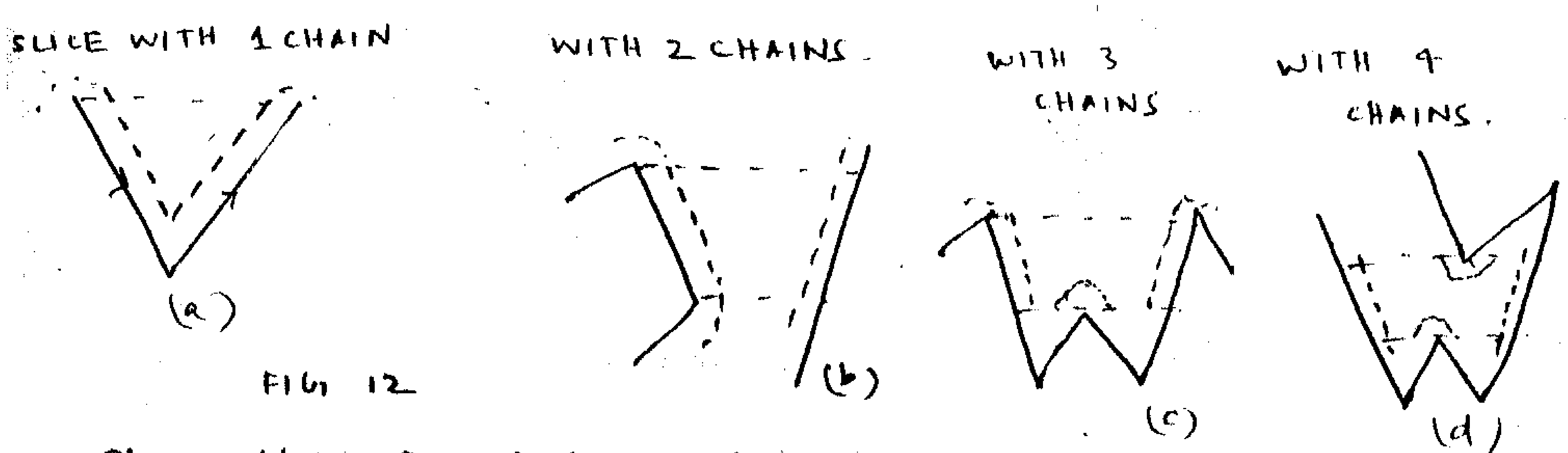


FIG 12

Observations 2 and 3 are helpful while drawing the guardzone inside a notch.

Generation of slicing structure: In order to build the slicing structure we first have to sort the vertices of the notch in decreasing order of their y-co-ordinates. The lid of the notch consists of the start and end vertices of the notch and it is considered as the top of the first slice.

When the top of a slice is generated it becomes active. The slice bounded by two (usually nonvertical) edges to its left and right may be swept downward until a point appears inside it. The point inside it may be on one of the bounding edges or may fall in the middle of its bounding edges. In either case the slice is closed and preserved in the slicing datastructure described below. In the former case, it gives birth to the top of a new slice,and in the latter case the top of two new slices are generated. The newly generated slice(s) are linked to their parent slice.

The vertices of the notch are processed in decreasing order of their y-co-ordinates. At a particular instant of time, more than one slice may be open, whose top boundaries are disjoint. Thus there exists a linear order among the slices and can be preserved in a height-balanced binary tree. Let the slices a1, a2, a3 be active at a particular instant of time. In figure 13 (a) below the vertex p lies in slice a2. Here the active slices are

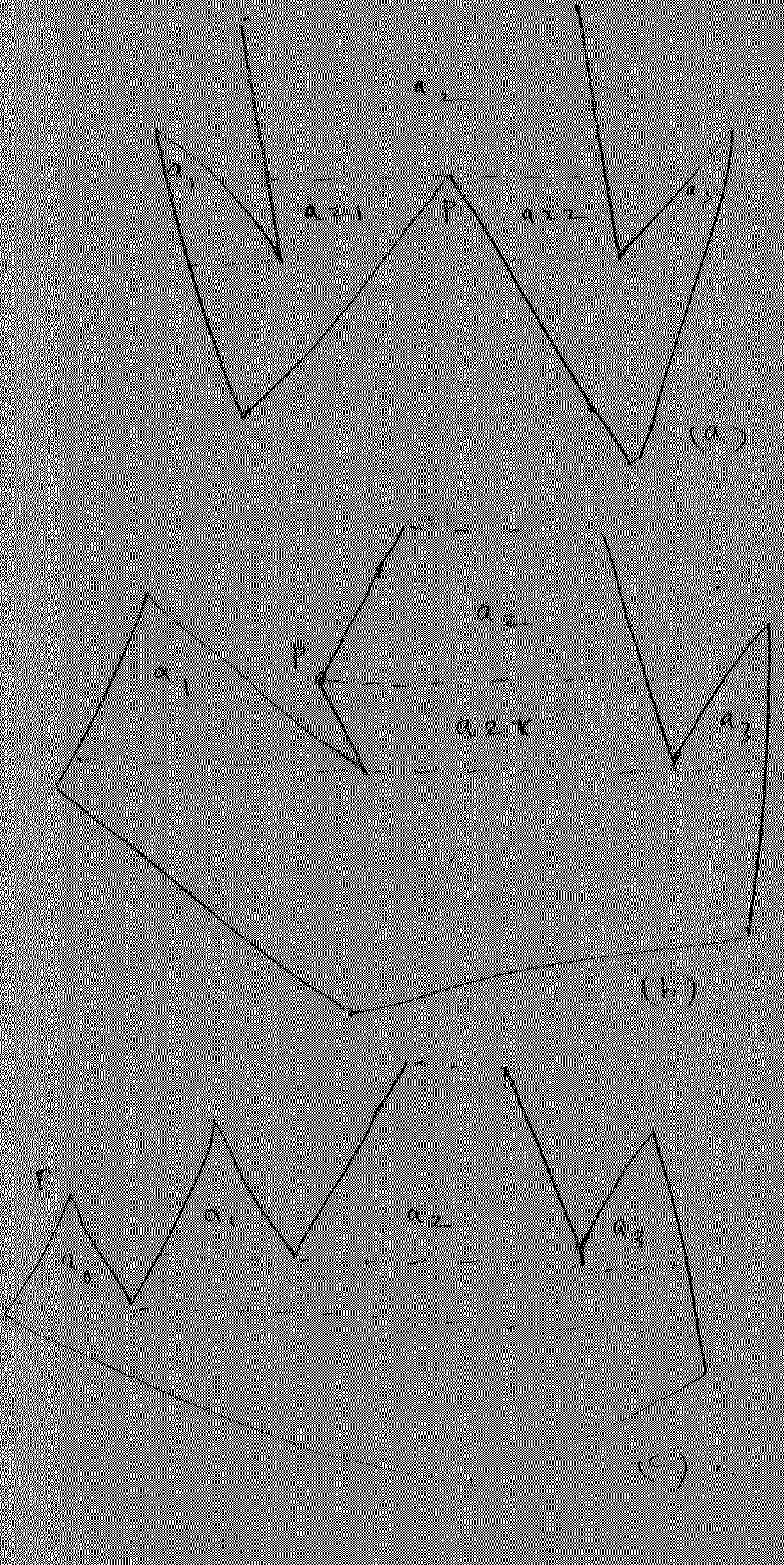


FIG 13

13 (b)

a1,a21,a22,a3.In figure below the vertex p lies on the left edge of slice 2, so the active slices are a1, a2*, a3.In figure 13 (c) the vertex p lies to the left of slice a1,hence the currently active slices are a0, a1, a2, a3. After processing a new slice, the affected slice is deleted and the newly generated slice(s) are inserted dynamically in O(log k) time, where k is the number of active slices at that particular instant of time. The slicing data structure consists of the following fields.

a) The left and right arms.
b) The top and bottom arms.
c) The slice width.
d) A pointer to those slices that are adjacent to the current slice and the edges along which we can enter one slice from the other.
e) A pointer to the four linked lists pointing to the first elements of the chain that are currently inside the slice.

4.4 GENERATION OF THE GUARD ZONE

In our algorithm of drawing the guard zone, the edges and vertices of the polygon are processed in their usual order. When an edge of the polygon is processed, its parallel guard-zone is constructed. For each slice, through which it passes, the following checkings are to be done:
a) whether it intersects any of the chains that are alreadv present in the slice.
b) whether it hits a boundary of the polygon.
In both the cases appropriate end points of the guard-zone line is decided, and the process continues from that point.
When a vertex of the polygon is processed the polygonal chain around it, is decided. Now some checkings as in the previous case are made here also.
Complexity analysis:

The construction of the slicing data structure requires O(nlogn) time. (This is due to the sorting step involved).While drawing the guard zone of a notch, in the worst case all the vertices and edges are to be processed. To draw the guard zone around a vertex or along an edge of the line we may have to check the intersection with atmost three chains. The number of elements in a chain is very less in comparison with the total number of parts in the guard zone. So it is assumed to be constant. Hence the complexity of our algorithm is

$$O(n\log n) + \sum_{i=1}^{k} O(n_i)$$ where k is the number of notches

n_i is the number of vertices inside the i th notch.

Thus the overall complexity of our algorithm is O(nlogn).

Appendix 1

We discuss below, techniques to determine the extent of the portion of a guard zone (m,n) lying parallel to an edge (a,b) of the polygon. The quantity δ denotes the guard zone distance. The co-ordinates of vertex a are (x1,y1) and those of vertex b are (x2,y2). We consider all the eight different orientations possible for edge (a,b) as outlined in observation 1.

.Type 1 edge :

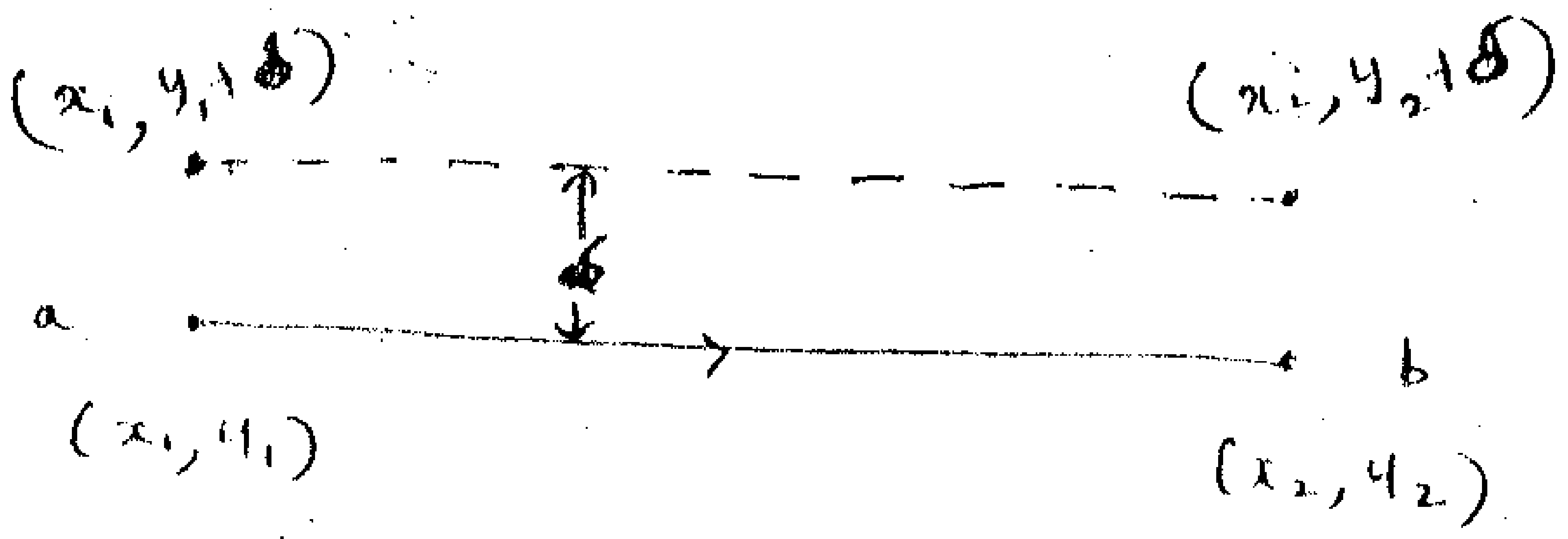
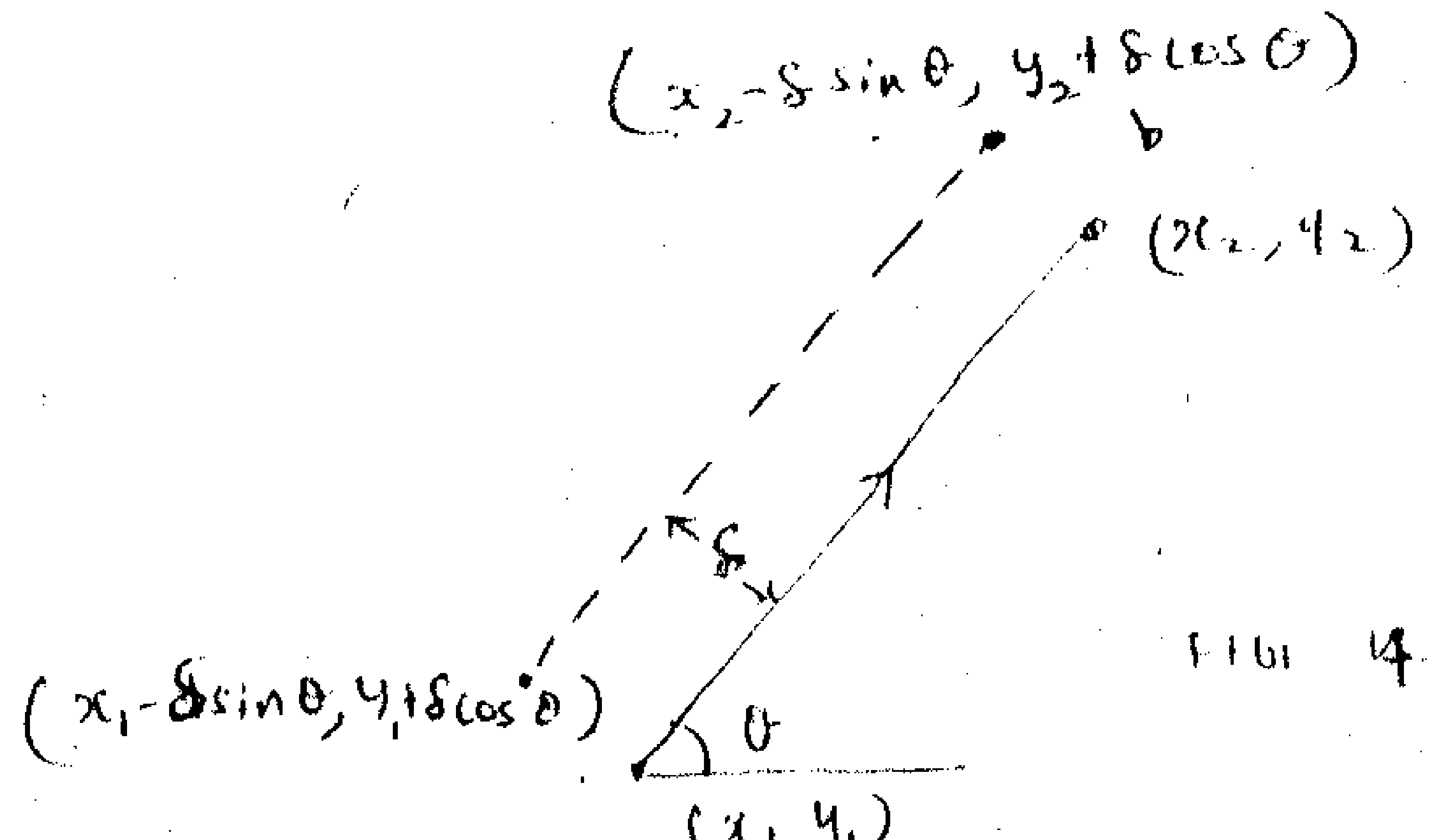


FIG 14 (a

.Type 2 edge:



$(x_2-\delta\sin\theta, y_2+\delta\cos\theta)$
b
(x_2, y_2)
δ
$(x_1-\delta\sin\theta, y_1+\delta\cos\theta)$
θ
(x_1, y_1)
a

FIG 14 (b)

Note The $\angle\theta$ in the above case is given by $\tan^{-1}((y2-y1)/(x2-x1))$

.Type 3 edge:

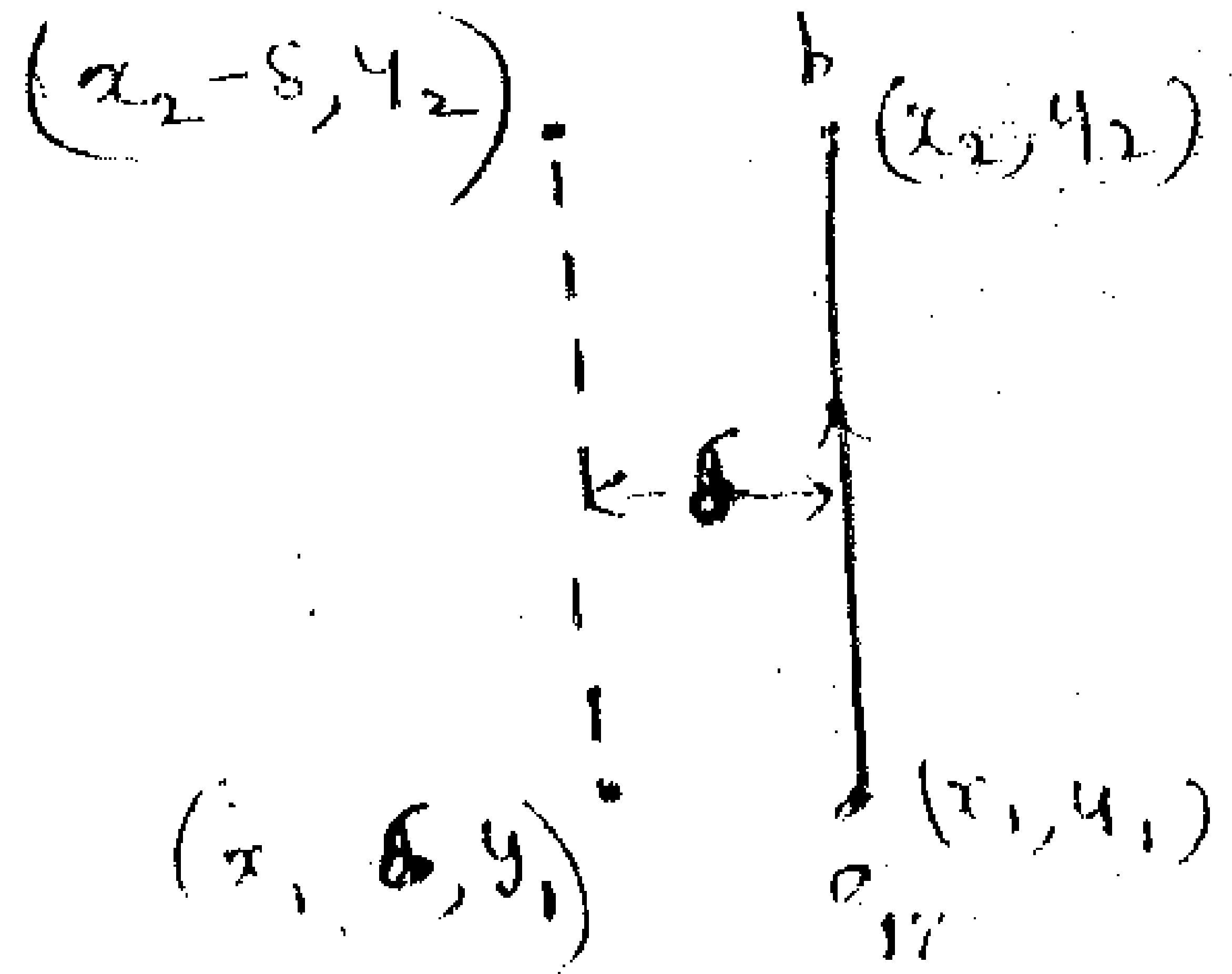


FIG 14-(c)

ype 4 edge :

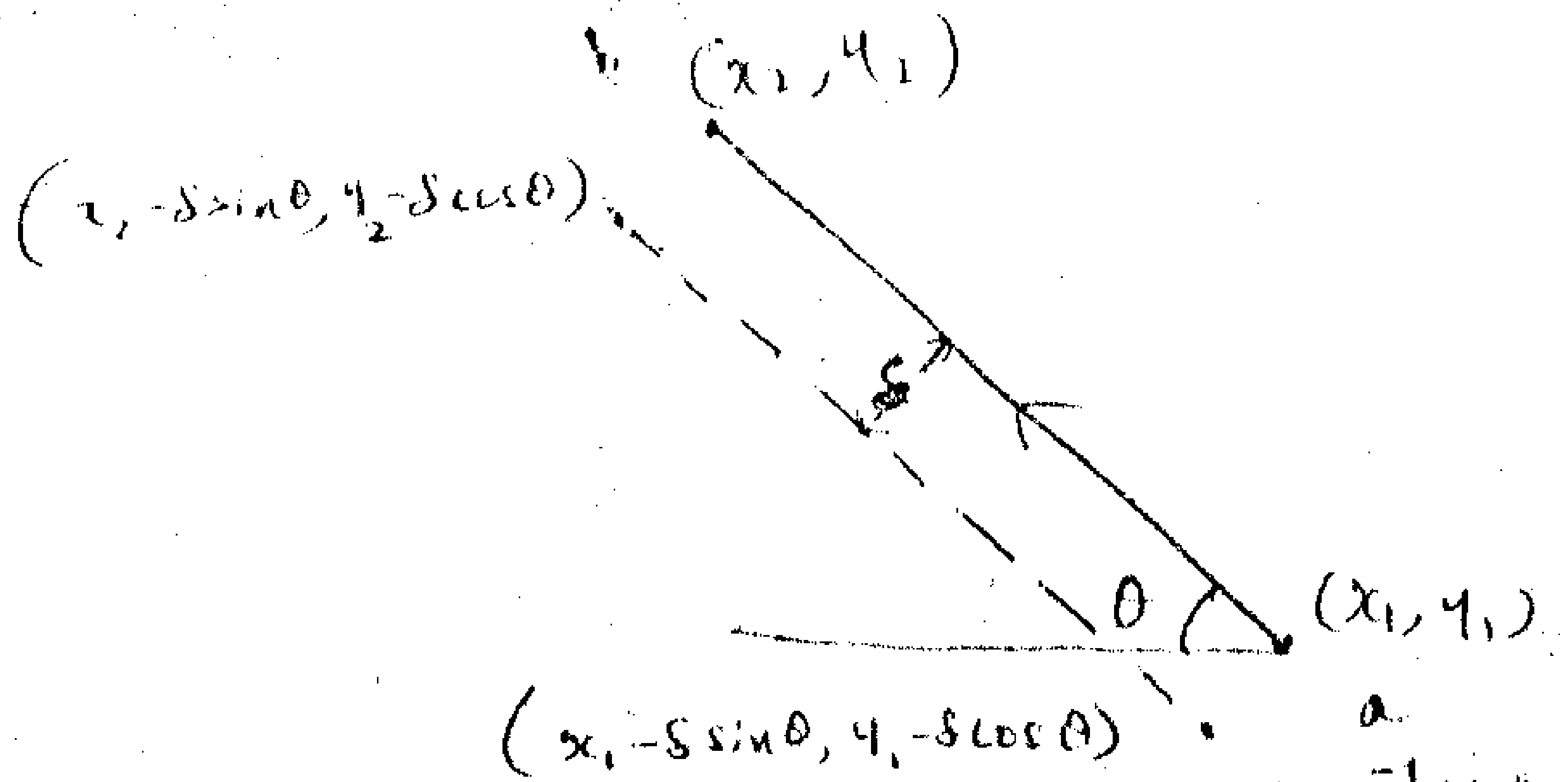

b
(x_2, y_2)
$(x_2 - \delta \sin\theta, y_2 - \delta\cos\theta)$
δ
θ
(x_1, y_1)
a
$(x_1 - \delta\sin\theta, y_1 - \delta\cos\theta)$

FIG 14(d)

ote The $\angle\ \theta$ in the above case is given by $\tan^{-1}((y2-y1)/(x1-x2))$.

Type 5 edge:

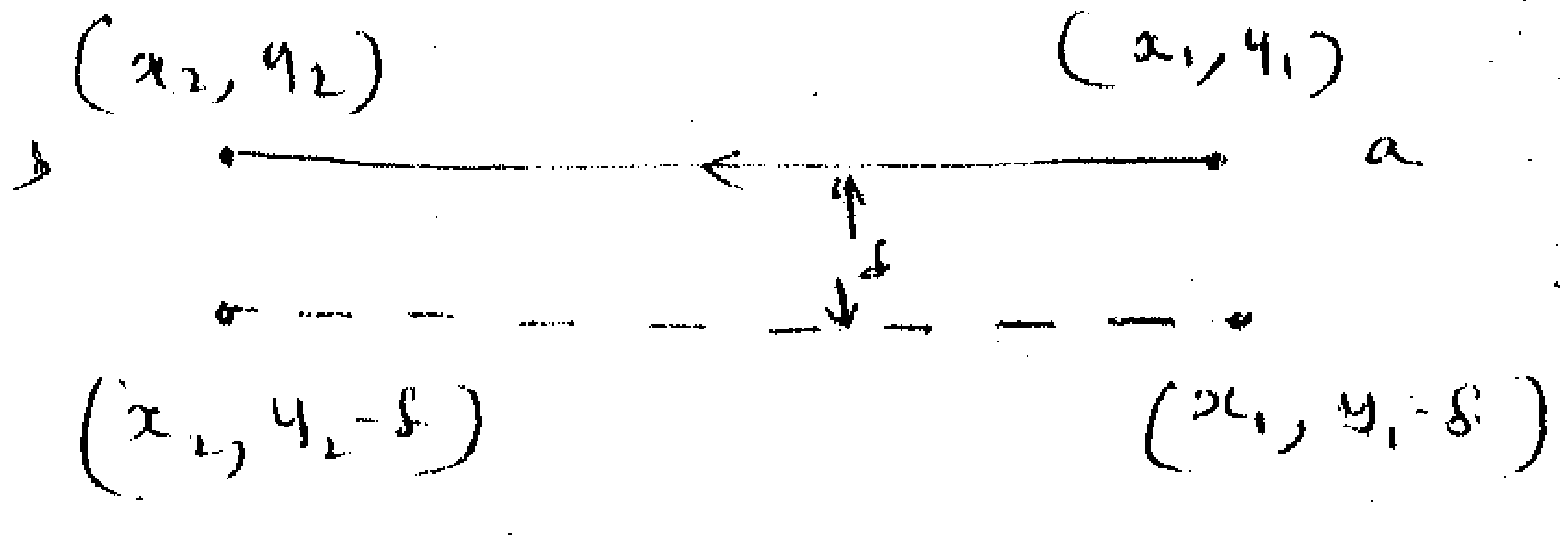
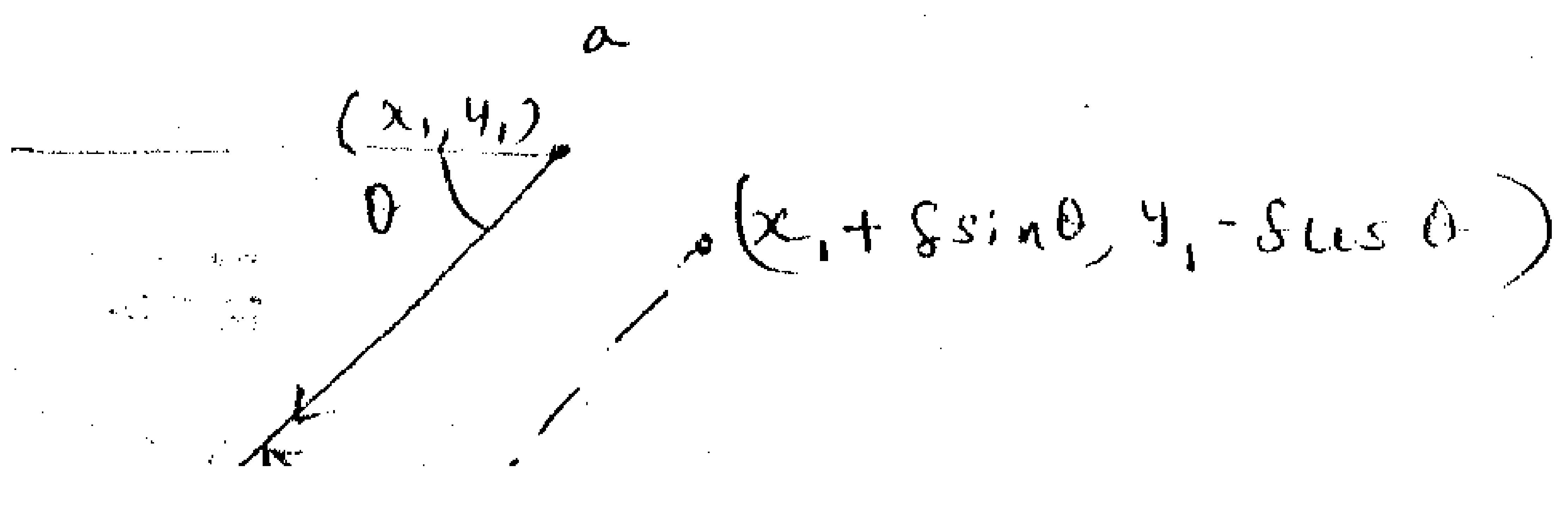


FIG 14(e)

Type 6 edge :



Note:The $\angle \theta$ in the above case is given by $\tan^{-1}((y1-y2)/x1-x2))$.

.Type 7 edge:

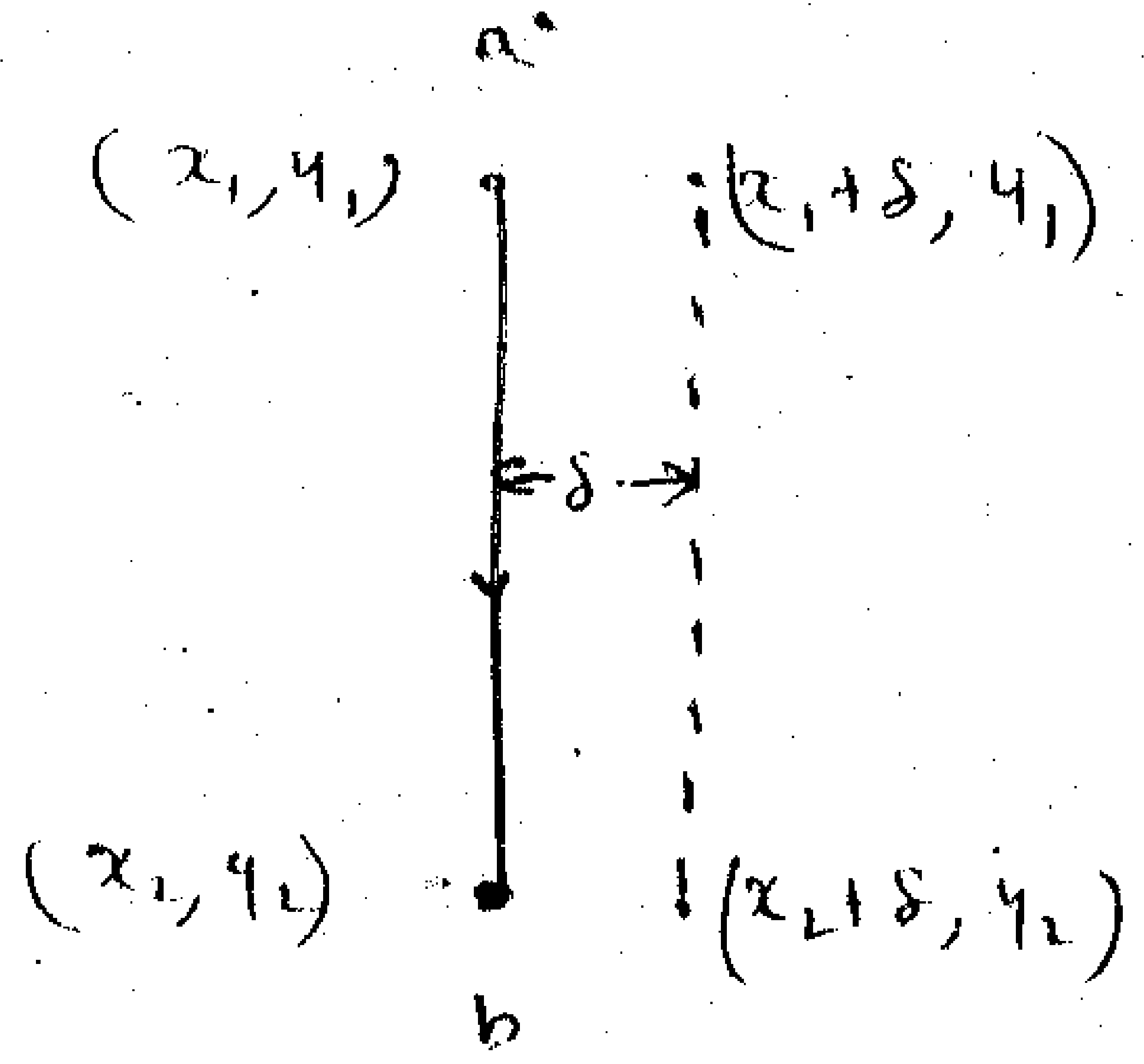


FIG 14 (g)

.Type 8 edge:

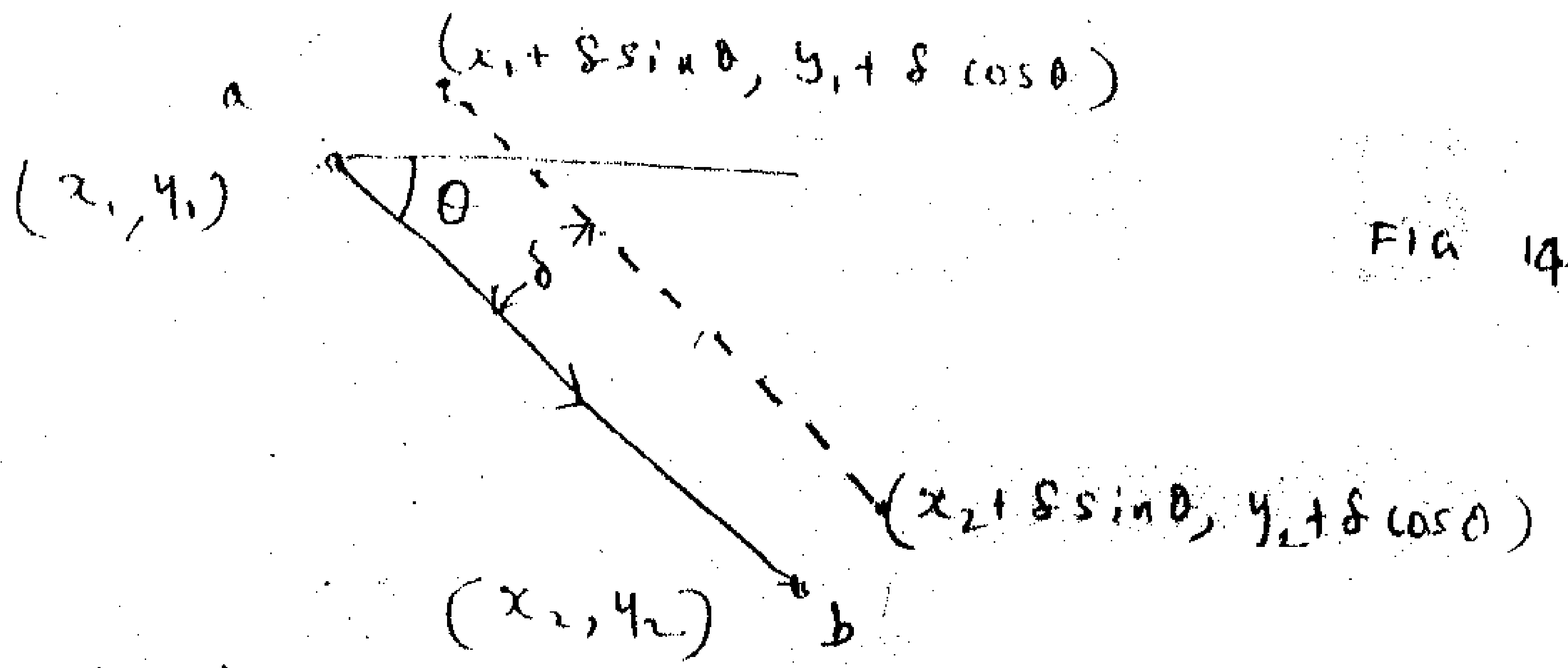

$(x_1+\delta\sin\theta, y_1+\delta\cos\theta)$
a
(x_1,y_1)
θ
δ
$(x_2+\delta\sin\theta, y_2+\delta\cos\theta)$
(x_2,y_2)
b

FIG 14 (h)

Note The $\angle \theta$ in the above case is given by $\tan^{-1}((y1-y2)/x2-x1))$.

Appendix 2

Techniques for finding intersections:

1. Intersection of two line segments: Two non-parallellines will intersect at a point which can be determined by solving their equations. If the equations of two lines (a,b) and (c,d) are respectively

$a1\ x + b1\ y + c1 = 0$ and

$a2\ x + b2\ y + c2 = 0.$

then their point of intersection (x',y') is given by the following expressions

$x' = ((b1\ c2 - b2\ c1)/(a1\ b2 - a2\ b1))$

and $y' = ((c1\ a2 - c2\ a1)/(a1\ b2 - a2\ b1))$

If a1 b2 =a2 b1, then the lines are parallel and do not intersect in the finite domain.

If we consider two line segments, the intersection point must lie between the bounds of the two lines.

To sum up if E (x1,y1) and F (x2,y2) constitute one line segment and if G (x3,y3) and H (x4,y4) constitute another, the intersection point determined as mentioned above must have its x-co-ordinate lying between (x1 and x2) and (x3 and x4). It's y-co-ordinate must lie between (y1 and y2) and between (y3 and y4).

2. Intersection of a line segment and a circular arc segment:

A line segment and a circular arc segment can intersect atmost at two points. We specify the procedure to determine the intersection points below.

We first determine the intersection points of the line and the circle having same centre and radius as the circular arc by solving their equations.

The equation of the line is of the form ax + by + c = 0 and that of the circle is of the form

$(x-\alpha)^2 + (y-\beta)^2 = \delta^2$; (α,β) being the centre of the circle and δ it's radius.

If the intersection points are imaginary then the line does not intersect the circular arc. If not, they intersect, the points of intersection (x',y') and (x'',y'') given by the solution of the two equations.
Next we have to determine whether the points of intersection lie within the line bounds and the arc bounds. We can determine if a point of intersection lies between the line bounds using the methods described in previous section. We explain below, how to determine whether a point (*p*) lies between the bounds of an arc of the guardzone whose extremes are given by the points *arcstart* and *arcend* respectively. In this context we use the consequences of lemma 1.

Consequence of lemma 1: The Euclidean distance between *arcstart* and *arcend* of an arc of a guardzone is always greater than the Euclidean distance between *arcstart* and any intermediate point *p* on the arc.

We generate the diameter through *arcstart* and check whether the point *p* lies on the same side of this line as *arcend*. If not we discard the point. If yes, then we check whether the euclidean distance Δ between *arcstart* and *arcend* is greater than the euclidean distance Δ^* between *arcstart* and *p*. If Δ is greater than Δ^* then the point *p* lies within the arc bounds ,else if Δ is less than Δ^* then *p* lies beyond the arc bounds and we discard it.We select those points lying between the bounds of both, the line and the arc.

3. Intersection of two circular arc segments:
Two circular arc segments of a guard zone may atmost intersect at two points. To determine the points of intersection we proceed as follows.

We first check whether the distance between the two centres
greater than 2δ. If yes, then the two circles are well-separate
and they will not intersect. If not, then we determine the point
of intersection of the two circles of which the circular arcs for
a part in the following way.

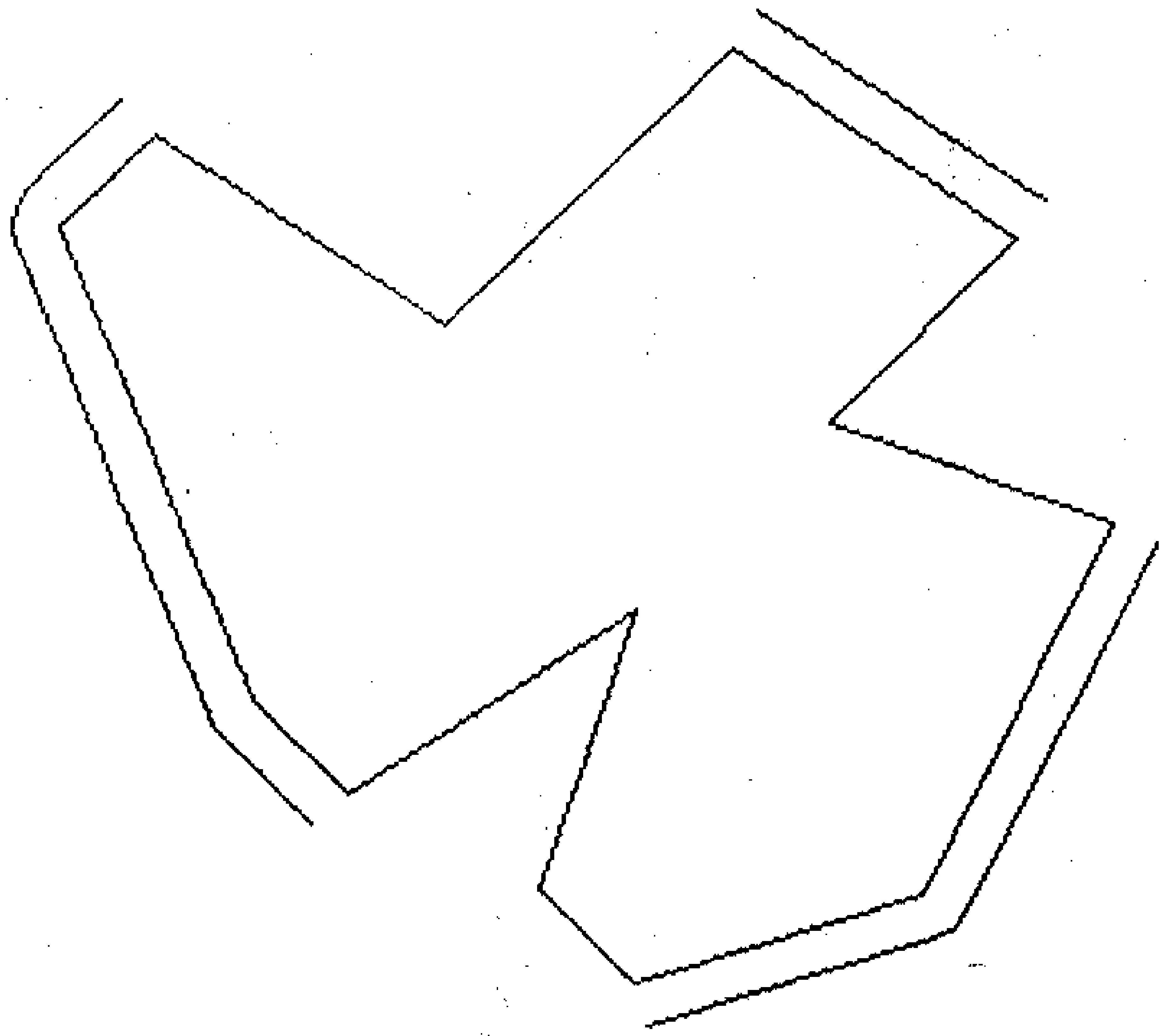
We generate the line passing through the mid-point of the lin
connecting the centres and lying perpendicular to it. We check th
intersection points of this line with one of the circles, usin
the methods of the previous section. We check whether the obtaine
points lie between the bounds of both the circular arcs, agai
using the methods described in the previous section. We onl
accept those points which lie between the bounds of both the arcs.

REFERENCES:

[1] A.Asano, M.Sato and T.Ohtsuki,"Computational Geometry algorithms", in Layout Design and Verification, Advances in CAD for VLSI Vol-4 (Ed.T.Ohtsuki),North Holland 1986.

[2] F.P.Preparata & M.I.Shamos, Computational Geometry : An Introduction, Springer Verlag, NY, 1985.

PLATE 1

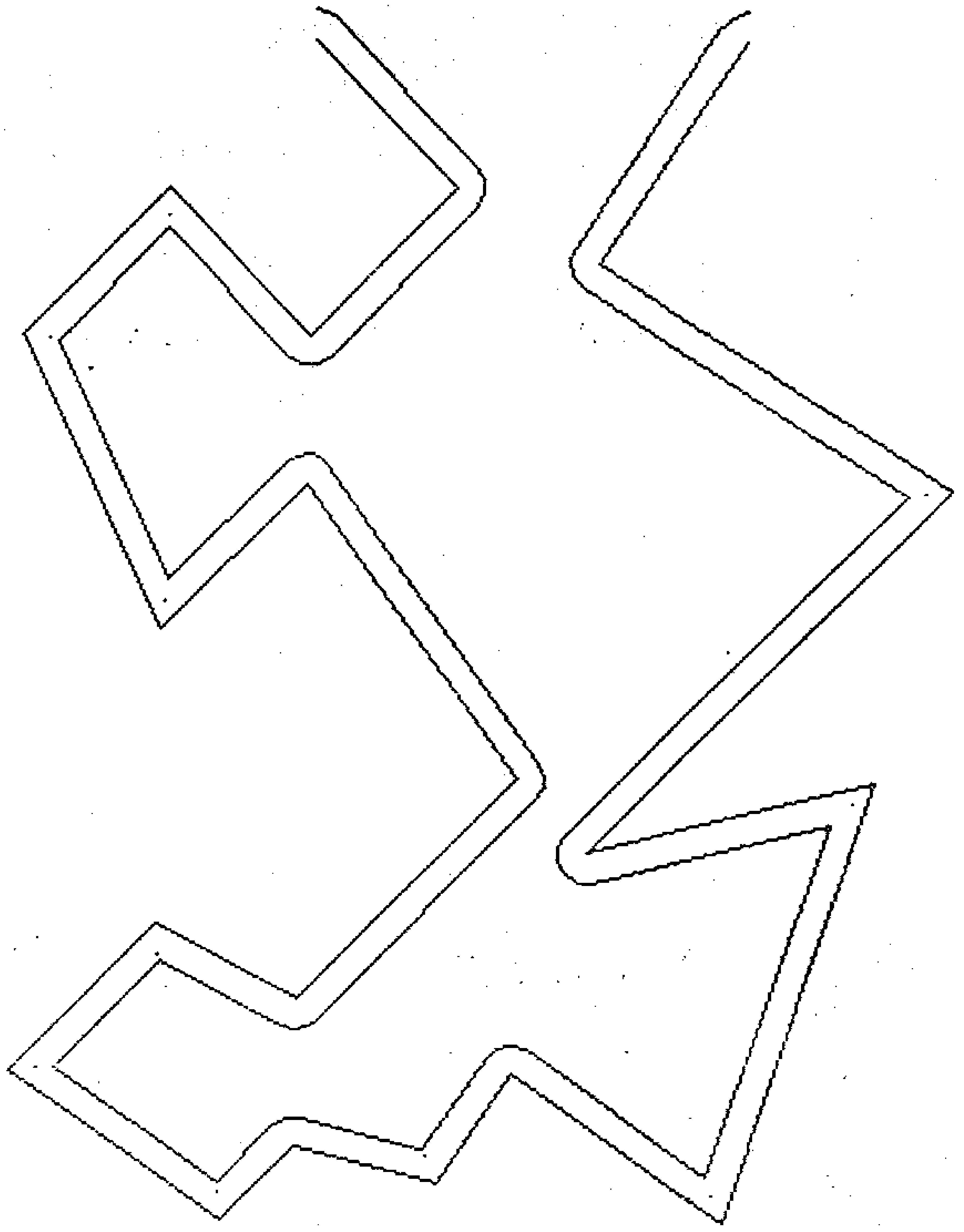


SHOWS THE GUARD ZONE AROUND THE CONVEX PART OF THE POLYGON.

NOTE: THE GUARD ZONE INSIDE THE NOTCHES ARE NOT SHOWN

THE PLATES IN THIS REPORT SHOW THE OUTPUTS OBTAINED, BY IMPLEMENTING THE ALGORITHMS EXPLAINED EARLIER, IN C LANGUAGE, ON THE VAX 8650 SYSTEM.

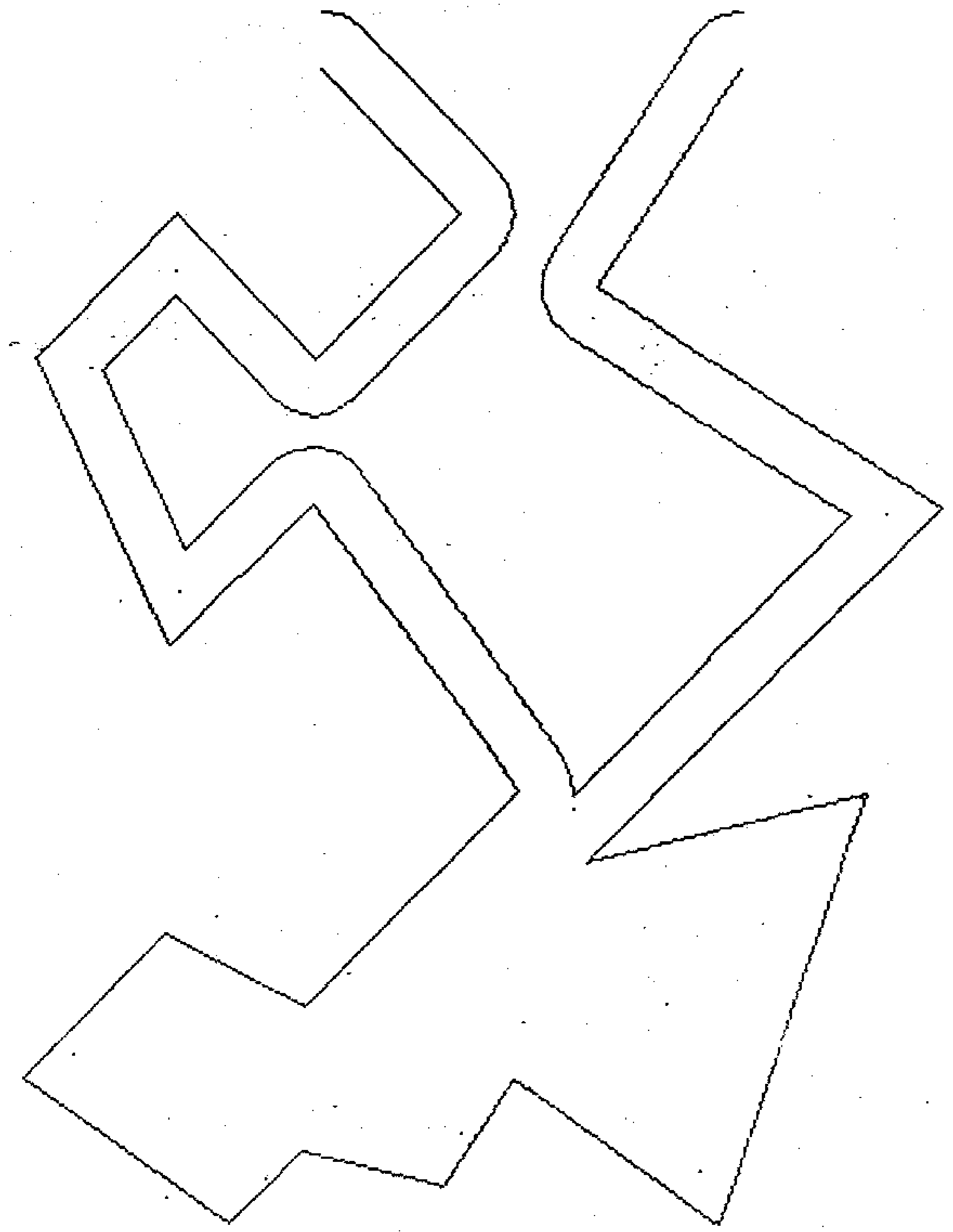
PLATE 2



(A)

A NOTCH AND THE GUARD ZONE WITHIN IT FOR A VERY SMALL DISTANCE δ

PLATE 2



(B)

THE SAME NOTCH

AND THE GUARDZONE

WHEN THE WIDTH & IS

INCREASED)

JS>

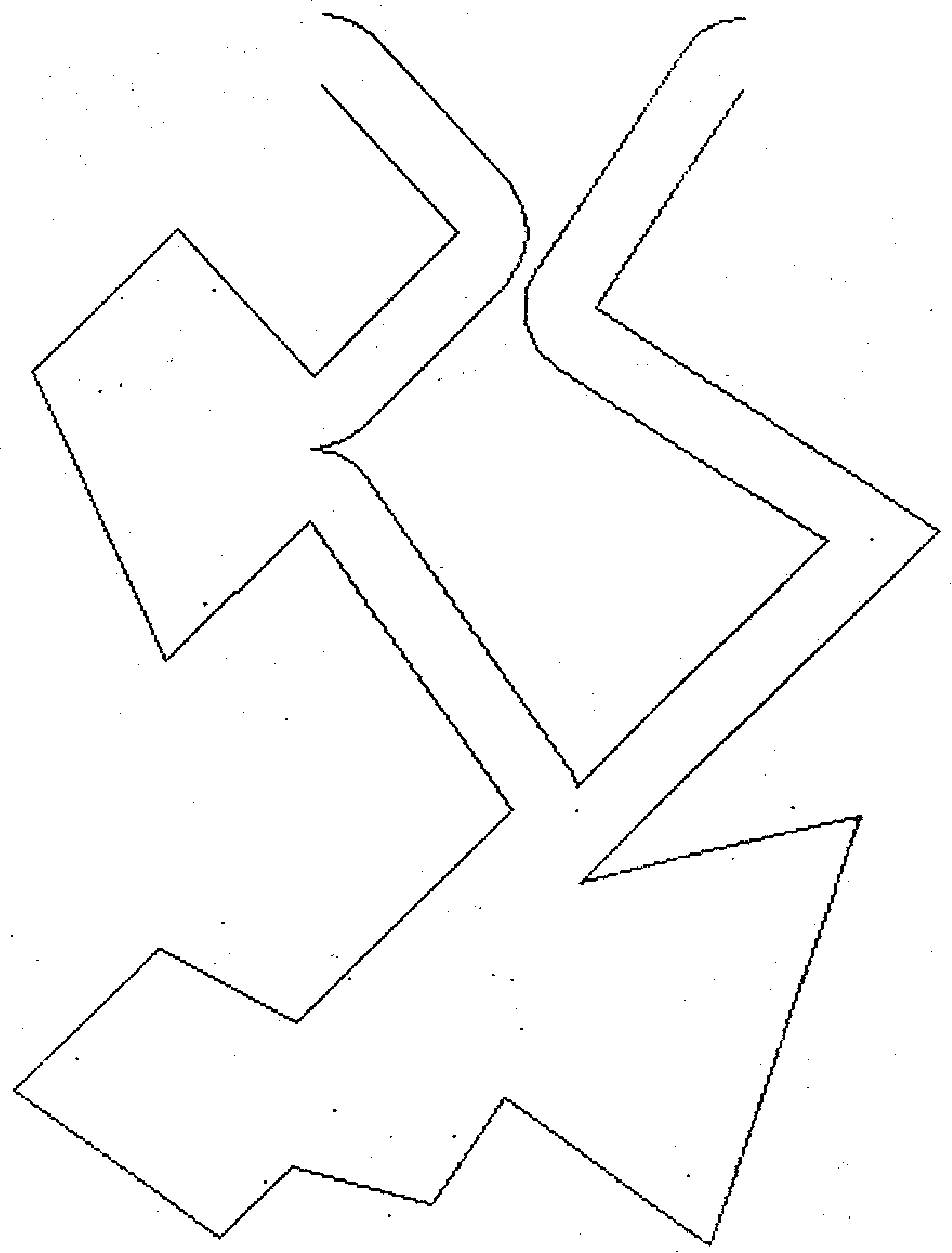


PLATE 2

(c)

A FURTHER INCREASE IN WIDTH & CHANGES THE GUARD ZONE SHAPE

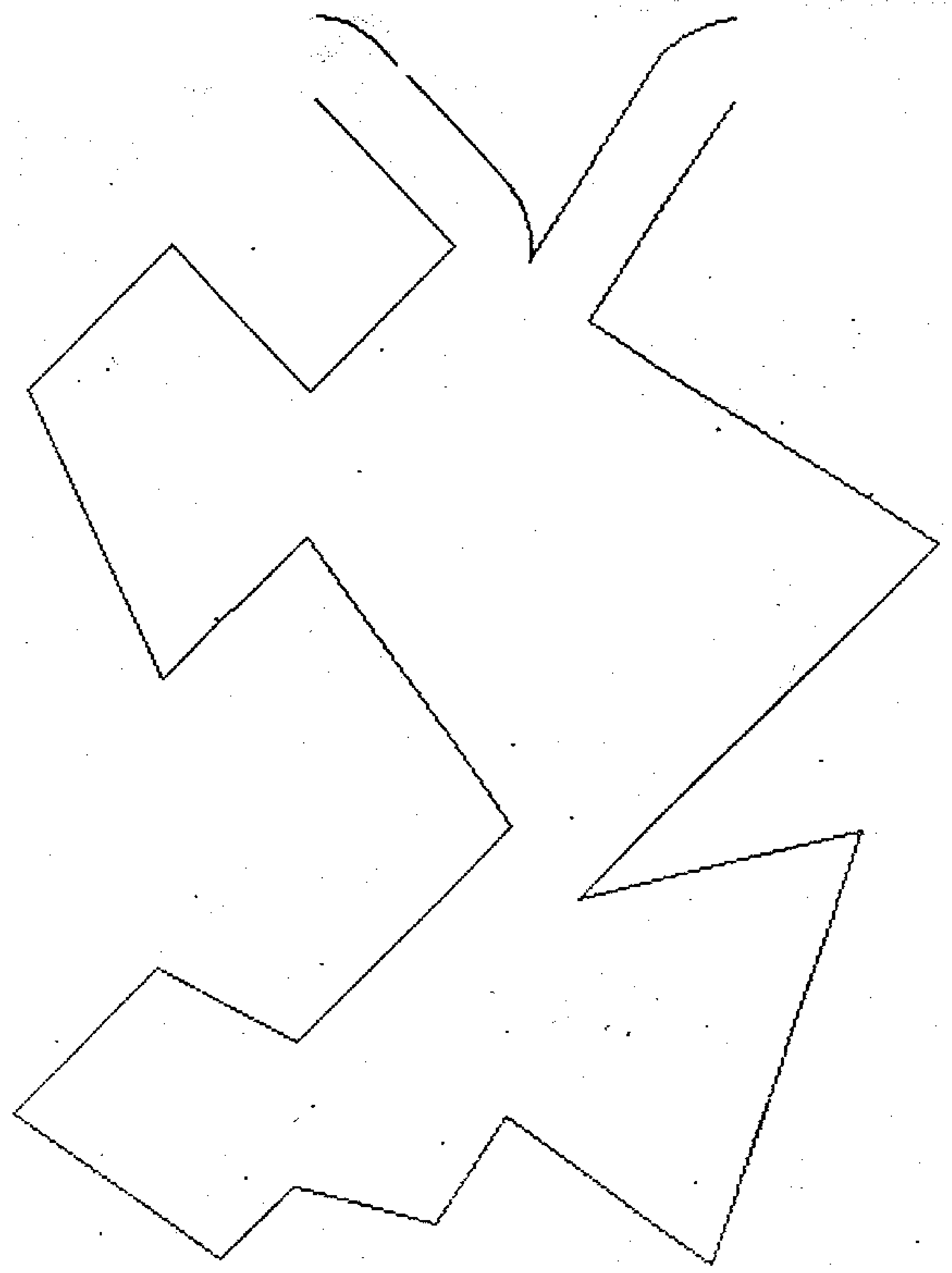


PLATE 2

(D)

INCREASING

THE WIDTH

FURTHER

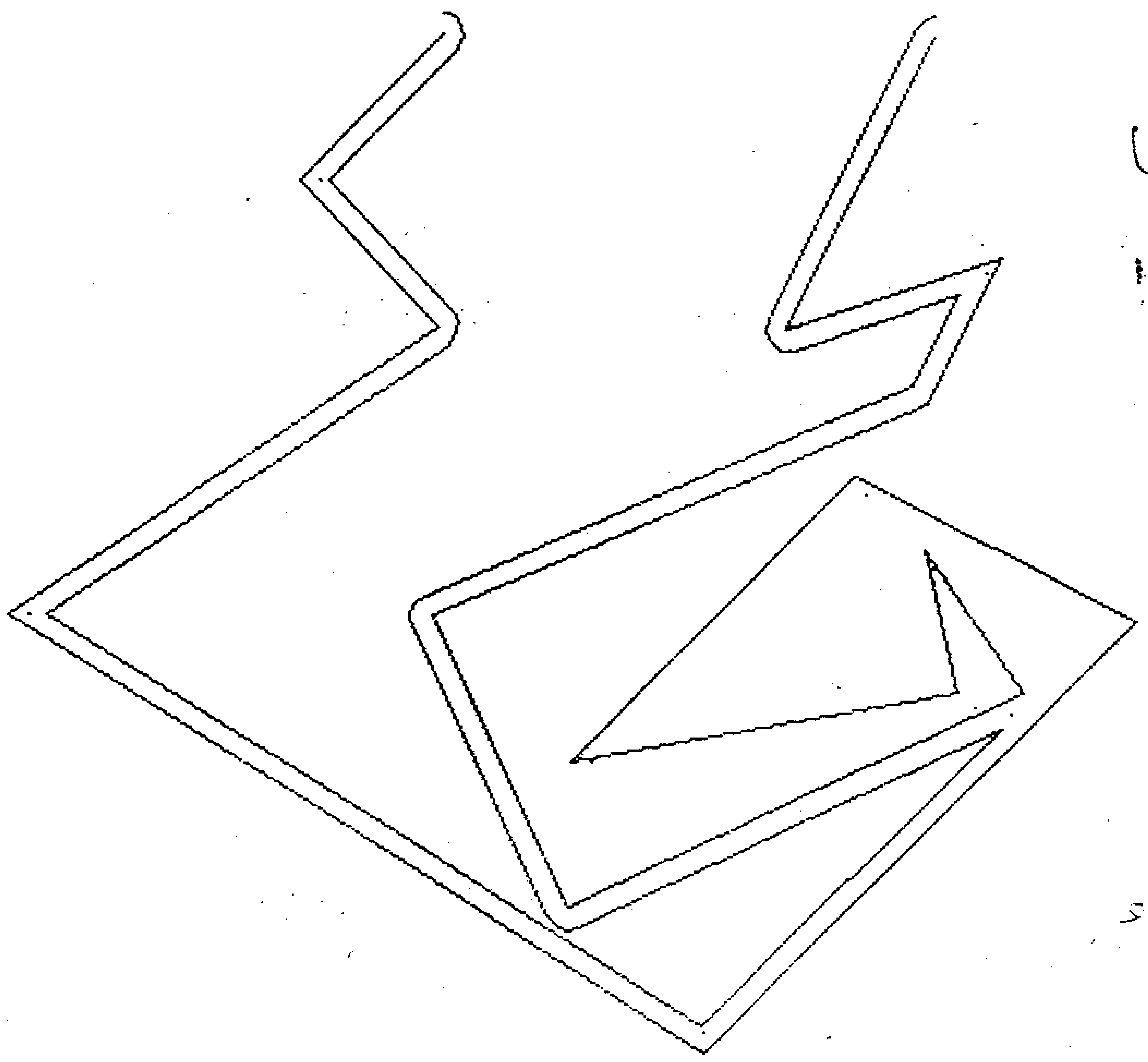
CAUSES THE

AMOUNT OF

PENETRATION

TO DECREASE

PLATE 3



(A)

IN (A) (B) (C) AND (D) THE EFFECT OF THE INCREASE IN WIDTH ON THE QUADRATIC SHAPE ARE DEPICTED