# Polar Split Tree as a Search Tool in Telecommunication

Farzad Bayat
Department of Electrical and Computer Engineering, Faculty of Engineering, Kharazmi University
std_farzadbayat@khu.ac.ir
Zahra Nilforoushan*
Department of Electrical and Computer Engineering, Faculty of Engineering, Kharazmi University
nilforoushan@khu.ac.ir

Received: 24/Feb/2018 Revised: 22/Aug/2018 Accepted: 16/Sep/2018

**Abstract**

Tree search algorithms are vital for the search methods in structured data. Such algorithms deal with nodes which can be taken from a data structure. One famous tree data structure is split tree. In this paper, to compute the split tree in polar coordinates, a method has been introduced. Assuming that the algorithm inputs (in form of points) have been distributed in the form of a circle or part of a circle, polar split tree can be used. For instance, we can use these types of trees to transmit radio and telecommunication waves from host stations to the receivers and to search the receivers. Since we are dealing with data points that are approximately circular distributed, it is suggested to use polar coordinates. Furthermore, there are several researches by search algorithms for the central anchor which leads to the assignment of a virtual polar coordinate system. In this paper, the structure of Cartesian split tree will be explained and the polar split tree will be implemented. Then, by doing nearest neighbor search experiments, we will compare the polar split tree and polar quad tree in terms of searching time and amount of distance to the closest neighbor and in the end, better results will be achieved.

**Keywords:** Split Tree; Polar Split Tree; Quad Tree; Polar Quad Tree; Nearest Neighbor Search.

## 1. Introduction

As it is apparent, nearest neighbor search can be used in a lot of cases relating to distance and it can also be used in classifying and clustering the data as a similarity criterion. Depending on the case, various algorithms can be used for nearest neighbor search. For instance we can find the nearest neighbor by searching trees including quad tree and split tree which will be briefly explained later. In some cases, depending on the case and the space in which the data are located, it is better to use polar coordinates. That's the reason why, we presented a method for the polar split tree. This algorithm has been compared with polar quad tree and the obtained results are compared in terms of searching time and the amount of distance to optimal response.

In the first section of this paper, searching trees and finding the nearest neighbor have been reviewed. In Section 2, quad tree and the method of computing it will be explained . Section 3 is devoted to the split tree . In Section 4, polar coordinates and some of its applications will be mentioned . In Section 5 we will indicate the way of construction and searching in polar quad trees. In Section 6, polar split tree is introduced and how to search the nearest neighbor by using this tree is defined . Section 7 is devoted to the investigating the efficiency of our proposed method and comparison of polar quad tree with polar split tree. Finally in Section 8, the conclusion and summing-up of the proposed algorithm are mentioned.

## 2. Quad Tree

The quad tree is a rooted tree whose internal node contains four children and each node represents one square. For a set P of points, the primary square of them is a square that contains all points of P and is the root of the tree that corresponds with partitions. If node *v* contains a child, the squares relating to its children will be the four areas of square *v*. That's why this tree is called quad tree. In other words, the leaf squares form a subset of root square and this subset is the subset of the quad tree. Fig. 1 indicates the quad tree and its subsets.

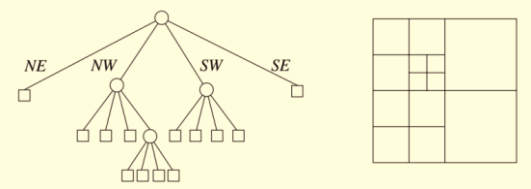


Fig. 1. A sample of quad tree and its subsets

The root children are called NE, NW, SW and SE that specify in which square they belong to. For example, NE belongs to northeast square, NW belongs to northwest square, etc.

Quad tree is used to save different kinds of data. For saving the a set of points in a plane, the square division will continue recursively as long as there is more than one point in a square. Thus the quad-tree for P inside the

* Corresponding Author

square $\sigma := [x_{min}: x_{max}] \times [y_{min}: y_{max}]$ is defined as follows:

- If the number of points of P are less than or equal to 1, the quad tree will just contain one leaf.
- Otherwise $\sigma_{NE}.\sigma_{NW}.\sigma_{SW}$ and $\sigma_{SE}$ show four areas of σ as follows:

$$x_{mid} = (x_{min} + x_{max})/2$$
$$y_{mid} = (y_{min} + y_{max})/2$$

$P_{NE} := \{p \in P: p_x > x_{mid} . p_y > y_{mid}\}$
$P_{NW} := \{p \in P: p_x \leq x_{mid} . p_y > y_{mid}\}$
$P_{SW} := \{p \in P: p_x \leq x_{mid} . p_y \leq y_{mid}\}$
$P_{SE} := \{p \in P: p_x > x_{mid} . p_y \leq y_{mid}\}$

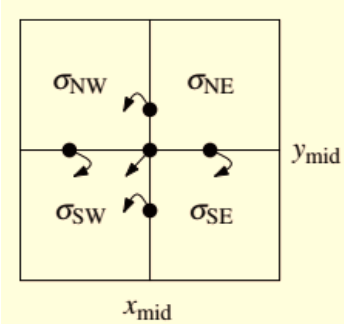

Fig. 2. A simple quad tree

**Lemma 2.1.** The depth of quad tree for the set P of points in the plane is at most *log(s/c)+3/2* in which *c* is the smallest distance between every two points in P and s is the lateral length of the primary square containing points of P . (For a proof refer to [3]).

**Theorem 2.2.** A quad tree with the depth of *d* is for saving *n* points containing *O((d+1)n)* nodes and can be constructed with time complexity of *O((d+1)n).* (For a proof refer to [3]).

## 3. Split tree

The split tree for a set of points *P* is a rooted binary tree data structure containing the points of *P* in its leaves. Split tree can be used as a searching tree in finding the nearest neighbor [10]. In split tree algorithm, each time the bounding box of inputs is considered and then the longest edge of it will be halved and the bounding box will be drawn for the left and right sub tree as long as the points inside the box is more than one point. In other words, if there is one point inside the bounding box, then the split tree consists of one single node that stores that point and the algorithm will stop.

**Theorem 3.1.** A split tree is constructed for saving *n* points. Its time complexity in the worst case and in the best case are $\theta(n^2)$ and *O(nlogn)* respectively and its height is $\theta(n)$. [10]

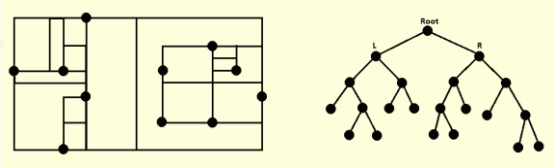

Fig. 3. A sample of split tree algorithm

For improving the complexity of split tree, partial split tree algorithm has been introduced whose threshold is *n/2*. It means that in each step a bounding box will be drawn for left and right children only if the numbers of their points are more than *n/2*. Thus, the number of children in the last level of tree can be between *1* and *n/2*.

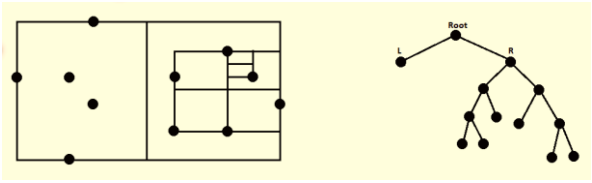

Fig. 4. A sample of partial split tree algorithm

## 4. Polar Coordinate and its Applications

polar coordinates is a two-dimensional coordinate system where each point *p* is represented by $(r.\theta)$. The distance of each point to the center of the coordinate is *r* and $\theta$ is the angle between *x*-axis and the line connecting *p* and the center of the coordinate [1,7 20].

A point can be converted from Cartesian coordinate to polar coordinate (and vice versa) as follows:

$x = rcos\theta \qquad y = rsin\theta$
$r^2 = x^2 + y^2 \qquad \theta = tan^{-1}\, y/x$

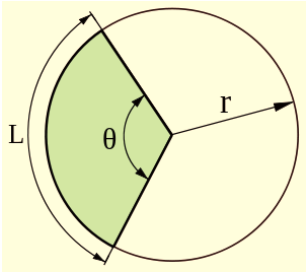

Fig. 5. A sector of the circle with angle θ and arc length L

For computing the length of arc L relating to angle $\theta$ in a circle with the radius *r,* the formula $L = r\theta$ must be used.

Polar coordinate can be used in most of physics equations such as circular movement of central force and planets rotation. Furthermore, in cases in which the data have been distributed in the form of a circle or part of a circle, polar coordinate can be used. For instance, when the data are in form of radio or telecommunication waves, as we are dealing with shapes which are approximately circular, using polar coordinate is recommended [2,17,19,21].

In the following parts, polar quad tree and polar split tree will be investigated.

## 5. Polar Quad Tree

A polar quad tree is a tree data structure in two-dimensional polar metric space in which each internal node has exactly four children. It is most often used to partition a two-dimensional space by recursively subdividing it into four subsectors or regions. A polar quad tree with $k$ levels has $4^k$ leaf cells, defined by $2^k$ angular and $2^k$ radial divisions. There are many applications for polar quad trees [4,8,13,18,22]. For drawing quad trees, the square covering input points is drawn and as long as the number of points in each square is more than 1, each square will be recursively divided into four parts. For drawing polar quad tree (PQT), the same steps must be followed as well but we are dealing with arc and part of radius instead of square sides.

### 5.1 Construction of Polar Quad Tree

The following steps shall be done for constructing the polar quad tree for input points *P*:

1. Draw the smallest circle containing the points.
2. Divide the circle containing input points into four equal parts by two perpendicular diameters.
3. Divide each quarter of the circle into four parts recursively. For dividing one quarter of the circle into four parts, we first halve the radius and then halve the related arc. The numbering could be done similar to Fig. 6.

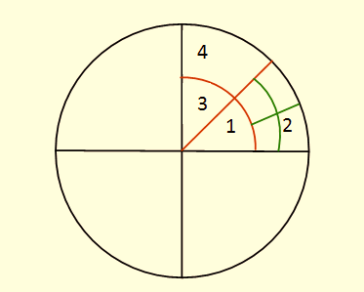


Fig. 6. Numbering polar quad tree

4. In Fig. 6, for dividing the children similar to areas corresponding 1 and 3, follow the previous steps. But for dividing the children similar to 2 and 4, half both the related arc and the part of radius located in that cell.
5. Repeat steps (1) to (4) as long as there is more than one point.

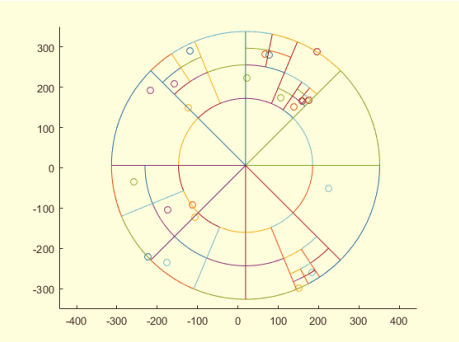


Fig. 7. A sample of polar quad tree

The pseudo code of the polar quad tree construction is as following:

**Polar_Quad_Tree_Construction (DataPoints P)**:

1. **Compute** smallest circle $C$ enclosing $P$ with *Nimrod Mojido Algorithm* [9].
2. **Divide** the circle into 4 parts: Part1, Part2, Part3, Part4; Fig. 6.
3. **If** part1 has more than one data, then call **Polar_Quad_Tree_Construction (Part1)** and add it as first child of $C$.
4. **If** part2 has more than one data, then call **Polar_Quad_Tree_Construction (Part2)** and add it as second child of $C$.
5. **If** part3 has more than one data, then call **Polar_Quad_Tree_Construction (Part3)** and add it as third child of $C$.
6. **If** part4 has more than one data, then call **Polar_Quad_Tree_Construction (Part4)** and add it as forth child of $C$.
7. **Return** $C$.
8. **End**.

Note that the smallest enclosing circle for a set of n points in the plane can be computed in O(n) expected time [9].

Hence the time complexity of constructing polar quad tree is:

$T(n)=\max\{O(n), 4T(n/4)+O(n)\}=O(n\log n)$.

For similar tree structure see[6,11,14,16].

### 5.2 The Nearest Neighbor Search in Polar Quad Tree

After constructing the polar quad tree (dividing each cell into four parts) next job is to find ourthe nearest neighbor for each query point in the polar quad tree.To this end, we average the x and y coordinates of the points inside quarter 1 and 4. Let the average of the points inside quarter 1 be $(x_1', y_1')$ and the average of the points inside quarter 4 be $(x_4', y'_4)$. If we want to search a query point $(x_q, y_q)$, the coordinate of it is compared using the following formula:

$$x_m = (x_1' + x_4')/2 \, , y_m = (y_1' + y_4')/2$$

Then we decide which subset of the tree should be chosen to continue the search.

- If $x_m > x_q$ and $y_m > y_q$, quarter 4 will be chosen for searching,
- If $x_m > x_q$ and $y_m \leq y_q$, quarter 3 will be chosen for searching,
- If $x_m \leq x_q$ and $y_m > y_q$, quarter 2 will be chosen for searching,
- If $x_m \leq x_q$ and $y_m \leq y_q$, quarter 1 will be chosen for searching.

By itterating this searching method, we find a leaf in the polar quad tree that the query point belongs to.

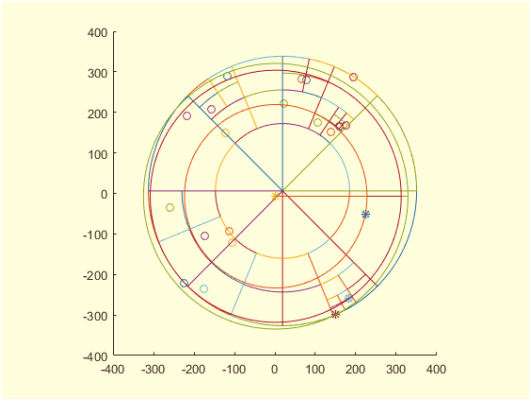


Fig. 8. Nearest neighbor search using polar quad tree

The correctness of this algorithm follow from the fact that for a given query point $Q = (r_q, \theta_q)$ inside the smallest enclosing circle $C$ of data, according to the values of radius $r_q$ and angle $\theta_q$ of $Q$, the point $Q$ is located in one of the four division regions. If the region containing $Q$ does not have more than one data, it will no longer be divided into four parts, and corresponding to it in the tree is a leaf indicating the region containing the $Q$. In this case, the search will terminate. If the region containing $Q$ contains more than one data, it is divided into four parts again (in the same way that each side of it is halved), and $Q$ is placed in one of the four regions according to the two values $r_q$ and $\theta_q$.

This process is recursively repeated until we reach to a leaf on the corresponding tree; that leaf has labeled with the name of region containing $Q$.

In particular, in Fig. 9, the point $Q$ with the * sign is contained in the gray region which corresponds to the gray color leaf of the corresponding tree.

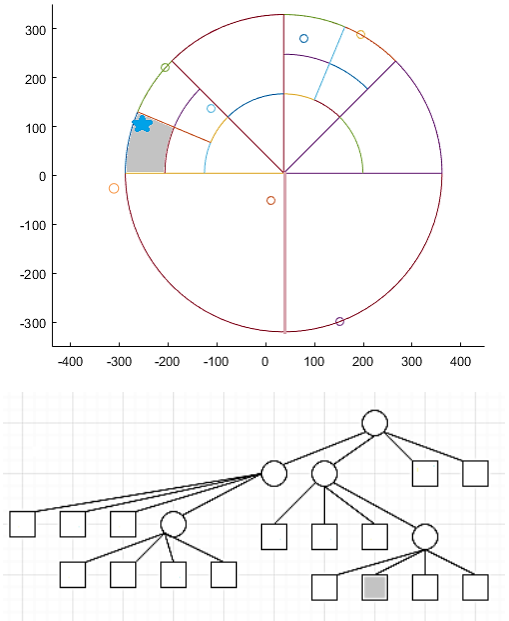


Fig. 9. A sample of nearest neighbor search for query point * using polar quad tree and corresponding search tree

As a result, finding the nearest neighbor in a polar quad tree can be computed in average case

S(n)=T(n)+logn=O(nlogn),

where T(n) is the average time complexity of polar quad tree construction.

## 6. Polar Split Tree

The polar split tree is a hierarchical rooted binary tree data structure in two- dimensional polar metric space which can be used as a searching tree in finding the nearest neighbor in polar metric spaces. Using polar split trees will usually improve the results obtained by the polar quad trees [5,15]. There are two steps for investigating polar split tree (PST), first the polar split tree is calculated and then the procedure of finding the nearest neighbor search in polar split tree is explained. In the following these two steps are explained.

### 6.1 Construction of Polar Split Tree

For construction the polar split tree, it is assumed that input data have been distributed in a circular shape. At first, based on Nimrod Megiddo algorithm, the smallest circle containing all points must be drawn [12] hich takes $O(n)$ time. Then similar to split tree in Cartesian coordinates, the longest edge must be halved. In order to do that, two components of angle and arc of the circle perimeter are compared and the one which is bigger, will be halved. For starters, we halve the perimeter of the circle. Then, we draw one of the circle's diameters and as a result the circle containing the points will be halved. Therefore the covering circle will be drawn for the left and right sub trees (part of a circle which is located at the center of the primary circle) and the length of radius sector will be compared with the angle sector and the one which is greater will be halved. The steps will continue recursively as long as there is just one point in the covering circle. For drawing the tree, we use the convention that, initially the horizontal diameter parallel to x-axis is drawn and the upper part and lower part of the circle will be the right and left child respectively.

In next steps, the right upper part will be the right child, the left upper part will be the left child, the right lower part will be the right child and the left lower part will be the left child. Finally, there will be a polar split tree. Fig. 10 indicates a sample of a polar split tree for set of 20 random input points. The pseudo-code for polar split tree construction is as follows:

**Polar_Split_Tree_Construction (DataPoints P)**

**1 Compute** the smallest circle C containing P using *Nimrod Mojido Algorithm* **[9].**

**2 Divide** the circle into two parts: Part1 and Part2.

**3 Compute** the smallest subsector covering points of Part1 and Part2, say S1 and S2.

**4 Compare** the length of arc and radius of S1 (and similarly for S2) and half the bigger one into Part 3 and Part4.

**5 If** Part 3 has more than one data, then go to line 4.

**6 If** Part 4 has more than one data, then go to line 4.

**7 End**

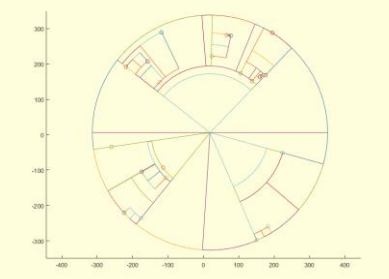

Fig. 10. A sample of polar split tree

## 6.2 Complexity of Polar Split Tree

As a polar split tree can be obtained by drawing a circle and we can draw a circle at least by three points, there will be two situations. Two points will be in one quarter and the other one will be in another quarter or every of these three points will be located in individual quarters. Therefore, considering the location of the points, the maximum depth of the tree will be $O(n)$. On the other hand, since this tree is a binary tree, the minimum depth of the tree will be the depth of a complete tree, i.e. $O(logn)$, and we have $O(logn) \leq h_{PST} \leq O(n)$. If there are points and the worst case of the tree happens, the depth of the tree will be $O(n^2)$ and the time constructing tree in the worst case will be and in the best case the height will be $O(logn)$ and the time of constructing the tree will be $O(nlogn)$.

## 6.3 The Nearest Neighbor Search in the Polar Split Tree

After construction the polar split tree for the set of input points, some random query points are tested to see whether this tree specifies the nearest neighbor to the point correctly or not. If for finding the nearest neighbor to the query point we just suffice to the search in the tree, in some cases the answer will be right and in some cases it will be wrong. The wrong answer will happen when the nearest point to the query point is in a cell to which the query point doesn't belong. But since the recursive functions use a depth first search mechanism, the point which is in the same cell as the query point will be chosen as the nearest neighbor and this answer is wrong. As it can be seen in Fig. 11, the query point which has been shown by * is nearer to point A but the tree recognizes point B as the nearest point because the query point is located in the cell belonging to B.

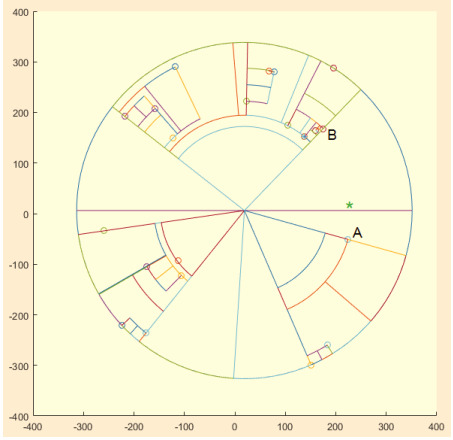


Fig. 11. Wrong output in the nearest neighbor search

For solving this problem, we will perform the search in kd-tree algorithm described in [23]. In kd-tree algorithm, when point x is found as the nearest point to the query point p, the distance between x and p is calculated and called r, i.e., d(x,p)=r. Then a circle with center p and radius r is drawn and the cells which are completely or partially inside this circle are investigated. Then the distance between the points inside every of these cells and p is calculated. If this distance is less than r, we update r and the nearest neighbor point. A circle is drawn with radius r and center p. Then the previous steps will be done recursively as long as there won't be any change in r and the nearest neighbor point. In the last step, the calculated point x is the nearest neighbor to the query point p. [12]

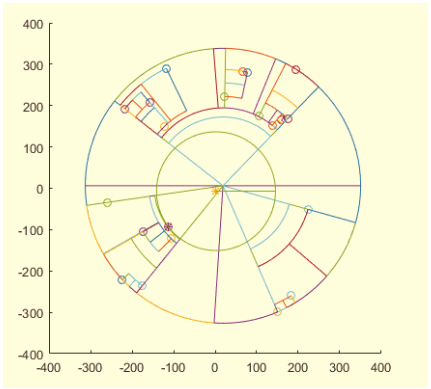

Fig. 12. Nearest Neighbor Search by polar split tree

The nearest neighbor searching pseudo code in polar split tree will be as following :

**Spt_NNS (Root root, Query QPoint)**

**1** CurNode = root
**2** distance = biggest integer number
**3 Start** from root.
**4 If** root has no children, then **return** null
**5 If** root has a child $C$ which is a leaf and its value is lower than distance, then distance = value of $C$, and **return** distance
**6 If** QPointX<=CurNodeX
**7** **If** QPointY-distance<=CurNodeY
**8** **If** it has a left child, then **return** Spt_NNS with left child
**9** **If** QPointY-distance>CurNodeY
**10** **If** it has a right child, then **return** Spt_NNS with right child
**11 Else**
**12** **If** QPointY-distance>CurNodeY
**13** **If** it has a right child, then **return** Spt_NNS with right child
**14** **If** QPointY-distance<=CurNodeY
**15** **If** it has a left child, then **return** Spt_NNS with left child
**16 End**

In order to see the correctness of this algorithm note that for a given query point $Q$ inside the smallest enclosing circle $C$ of data, if $Q$ is in the northern or southern semicircle of $C$, in the corresponding tree $T$ we

move from root to the left or right respectively. In each semicircle that $Q$ is located, we calculate the bounding box $B$ for the data of that semicircle according to the Polar_Splite_Tree_Construction algorithm and split the largest side into two halves. If the line that divides $B$ into two halves is an arc line, $B$ is divided into two upper and lower halves, and if $Q$ is in the upper or lower part of $B$, in $T$ we move to the left or right. If the line that divides $B$ into two halves is a radial line, $B$ is divided into two halves left and right, and if $Q$ is in the left or right part of $B$, in $T$ we move to the left or right.

These steps are recursively repeated until we reach to a leaf of $T$ which has labeled by the name of area containing $Q$.

In particular, in the Fig. 13, the point $Q$ is displayed by * and the area containing it (the gray zone) is corresponds to the gray color leaf of $T$.

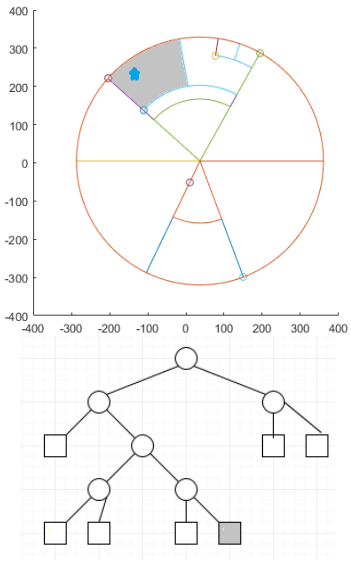


Fig. 13. A sample of nearest neighbor search for query point * using polar split tree and corresponding search tree

| | | | | Polar Split Tree (100) | | | Polar Quad Tree (100) | | |
|---|---|---|---|---|---|---|---|---|---|
| Row# | DataSet# | QuerySet# | QueryPoint | 1NN | Distance | Time(ns) | 1NN | Distance | Time(ns) |
| 1 | 1 | 1 | (2,-7) | (4,-25) | 18.11077028 | 0.09375 | (6,-22) | 15.5241747 | 0.046875 |
| 2 | 1 | 2 | (-9,8) | (-7,-27) | 35.05709629 | 0.015625 | (-31,-22) | 37.20215048 | 0.046875 |
| 3 | 1 | 3 | (0,-5) | (4,-25) | 20.39607805 | 0 | (6,-22) | 18.02775638 | 0.03125 |
| 4 | 1 | 4 | (9,1) | (4,-25) | 26.47640459 | 0.03125 | (-31,-22) | 46.14108798 | 0 |
| 5 | 1 | 5 | (2,-4) | (4,-25) | 21.09502311 | 0.03125 | (6,-22) | 18.43908891 | 0.03125 |
| 6 | 1 | 6 | (5,8) | (-6,33) | 27.31300057 | 0.015625 | (-31,-22) | 46.86149806 | 0 |
| 7 | 1 | 7 | (-7,-6) | (-7,-27) | 21 | 0.015625 | (6,-22) | 20.61552813 | 0 |
| 8 | 1 | 8 | (-6,7) | (4,-25) | 33.52610923 | 0 | (-31,-22) | 38.28837944 | 0 |
| 9 | 1 | 9 | (-4,0) | (4,-25) | 26.2488095 | 0.015625 | (6,-22) | 24.16609195 | 0 |
| 10 | 1 | 10 | (5,9) | (-6,33) | 26.40075756 | 0.03125 | (-31,-22) | 47.50789408 | 0.015625 |
| 11 | 2 | 1 | (2,-7) | (-14,0) | 17.4642492 | 0.03125 | (-7,-3) | 9.848857802 | 0.03125 |
| 12 | 2 | 2 | (-9,8) | (-14,0) | 9.433981132 | 0 | (-32,1) | 24.04163056 | 0 |
| 13 | 2 | 3 | (0,-5) | (-14,0) | 14.86606875 | 0.015625 | (-7,-3) | 7.280109889 | 0 |
| 14 | 2 | 4 | (9,1) | (-14,0) | 23.02172887 | 0.03125 | (-32,1) | 41 | 0 |
| 15 | 2 | 5 | (2,-4) | (-14,0) | 16.4924225 | 0.03125 | (-7,-3) | 9.055385138 | 0 |
| 16 | 2 | 6 | (5,8) | (-14,0) | 20.61552813 | 0.03125 | (-32,1) | 37.65634077 | 0.03125 |
| 17 | 2 | 7 | (-7,-6) | (-14,0) | 9.219544457 | 0.015625 | (-7,-3) | 3 | 0 |
| 18 | 2 | 8 | (-6,7) | (-14,0) | 10.63014581 | 0 | (-32,1) | 26.68332813 | 0.015625 |
| 19 | 2 | 9 | (-4,0) | (-14,0) | 10 | 0 | (-32,1) | 28.01785145 | 0.03125 |
| 20 | 2 | 10 | (5,9) | (-14,0) | 21.02379604 | 0.03125 | (-32,1) | 37.85498646 | 0.03125 |
| 21 | 3 | 1 | (2,-7) | (-7,-3) | 9.848857802 | 0.015625 | (14,-17) | 15.62049935 | 0.015625 |
| 22 | 3 | 2 | (-9,8) | (-8,5) | 3.16227766 | 0.015625 | (-46,-21) | 47.01063709 | 0 |
| 23 | 3 | 3 | (0,-5) | (-7,-3) | 7.280109889 | 0 | (-46,-21) | 48.70318265 | 0.015625 |
| 24 | 3 | 4 | (9,1) | (-7,-3) | 16.4924225 | 0.015625 | (-46,-21) | 59.23681288 | 0.03125 |
| 25 | 3 | 5 | (2,-4) | (-7,-3) | 9.055385138 | 0 | (-46,-21) | 50.92150823 | 0.046875 |
| 26 | 3 | 6 | (5,8) | (-7,-3) | 16.2728206 | 0 | (-46,-21) | 58.66856058 | 0 |
| 27 | 3 | 7 | (-7,-6) | (-7,-3) | 3 | 0 | (14,-17) | 23.70653918 | 0.03125 |
| 28 | 3 | 8 | (-6,7) | (-8,5) | 2.828427125 | 0.015625 | (-46,-21) | 48.82622246 | 0 |
| 29 | 3 | 9 | (-4,0) | (-7,-3) | 4.242640687 | 0 | (-46,-21) | 46.95742753 | 0.03125 |
| 30 | 3 | 10 | (5,9) | (-7,-3) | 16.97056275 | 0.015625 | (-46,-21) | 59.16924877 | 0 |
| 31 | 4 | 1 | (2,-7) | (14,-17) | 15.62049935 | 0.015625 | (18,-35) | 32.24903099 | 0 |
| 32 | 4 | 2 | (-9,8) | (14,-17) | 33.9705755 | 0.03125 | (-7,-14) | 22.09072203 | 0.03125 |
| 33 | 4 | 3 | (0,-5) | (14,-17) | 18.43908891 | 0 | (18,-35) | 34.98571137 | 0 |
| 34 | 4 | 4 | (9,1) | (14,-17) | 18.68154169 | 0.03125 | (-7,-14) | 21.9317122 | 0.03125 |
| 35 | 4 | 5 | (2,-4) | (14,-17) | 17.69180601 | 0.015625 | (18,-35) | 34.88552709 | 0.03125 |
| 36 | 4 | 6 | (5,8) | (14,-17) | 26.57066051 | 0.03125 | (-7,-14) | 25.05992817 | 0.03125 |
| 37 | 4 | 7 | (-7,-6) | (14,-17) | 23.70653918 | 0 | (18,-35) | 38.28837944 | 0.03125 |
| 38 | 4 | 8 | (-6,7) | (14,-17) | 31.2409987 | 0.015625 | (-7,-14) | 21.02379604 | 0.03125 |
| 39 | 4 | 9 | (-4,0) | (14,-17) | 24.75883681 | 0 | (18,-35) | 41.34005322 | 0 |
| 40 | 4 | 10 | (5,9) | (14,-17) | 27.51363298 | 0.015625 | (-7,-14) | 25.94224354 | 0.03125 |
| 41 | 5 | 1 | (2,-7) | (-7,-14) | 11.40175425 | 0.015625 | (18,-35) | 32.24903099 | 0 |
| 42 | 5 | 2 | (-9,8) | (-7,-14) | 22.09072203 | 0.03125 | (-7,-14) | 22.09072203 | 0.0625 |
| 43 | 5 | 3 | (0,-5) | (-7,-14) | 11.40175425 | 0.015625 | (18,-35) | 34.98571137 | 0 |
| 44 | 5 | 4 | (9,1) | (3,24) | 23.76972865 | 0.015625 | (-7,-14) | 21.9317122 | 0.03125 |
| 45 | 5 | 5 | (2,-4) | (-7,-14) | 13.45362405 | 0.015625 | (18,-35) | 34.88552709 | 0 |
| 46 | 5 | 6 | (5,8) | (3,24) | 16.1245155 | 0.015625 | (-7,-14) | 25.05992817 | 0.015625 |
| 47 | 5 | 7 | (-7,-6) | (-7,-14) | 8 | 0 | (18,-35) | 38.28837944 | 0 |
| 48 | 5 | 8 | (-6,7) | (-7,-14) | 21.02379604 | 0.03125 | (-7,-14) | 21.02379604 | 0.03125 |
| 49 | 5 | 9 | (-4,0) | (-7,-14) | 14.31782106 | 0.015625 | (18,-35) | 41.34005322 | 0.03125 |
| 50 | 5 | 10 | (5,9) | (3,24) | 15.13274595 | 0.015625 | (-7,-14) | 25.94224354 | 0.046875 |
| 51 | 6 | 1 | (2,-7) | (-7,-14) | 11.40175425 | 0 | (14,-30) | 25.94224354 | 0.09375 |
| 52 | 6 | 2 | (-9,8) | (-7,-14) | 22.09072203 | 0.015625 | (-23,-16) | 27.78488798 | 0 |
| 53 | 6 | 3 | (0,-5) | (-7,-14) | 11.40175425 | 0 | (14,-30) | 28.65309756 | 0.03125 |
| 54 | 6 | 4 | (9,1) | (3,24) | 23.76972865 | 0.015625 | (-23,-16) | 36.23534186 | 0 |
| 55 | 6 | 5 | (2,-4) | (-7,-14) | 13.45362405 | 0 | (14,-30) | 28.63564213 | 0.03125 |
| 56 | 6 | 6 | (5,8) | (3,24) | 16.1245155 | 0.015625 | (-23,-16) | 36.87817783 | 0 |
| 57 | 6 | 7 | (-7,-6) | (-7,-14) | 8 | 0 | (1,-36) | 31.04834939 | 0 |
| 58 | 6 | 8 | (-6,7) | (-7,-14) | 21.02379604 | 0.015625 | (-23,-16) | 28.60069929 | 0 |
| 59 | 6 | 9 | (-4,0) | (-7,-14) | 14.31782106 | 0.03125 | (-23,-16) | 24.8394847 | 0 |
| 60 | 6 | 10 | (5,9) | (3,24) | 15.13274595 | 0.015625 | (-23,-16) | 37.53664876 | 0 |
| 61 | 7 | 1 | (2,-7) | (7,-11) | 6.403124237 | 0 | (9,-15) | 10.63014581 | 0.03125 |
| 62 | 7 | 2 | (-9,8) | (7,-11) | 24.8394847 | 0 | (-9,-6) | 14 | 0 |
| 63 | 7 | 3 | (0,-5) | (-6,-1) | 7.211102551 | 0.03125 | (9,-15) | 13.45362405 | 0.015625 |
| 64 | 7 | 4 | (9,1) | (7,-11) | 12.16552506 | 0 | (-2,-6) | 13.03840481 | 0 |
| 65 | 7 | 5 | (2,-4) | (-6,-1) | 8.544003745 | 0.03125 | (9,-15) | 13.03840481 | 0.03125 |
| 66 | 7 | 6 | (5,8) | (-6,-1) | 14.2126704 | 0 | (-2,-6) | 15.65247584 | 0 |
| 67 | 7 | 7 | (-7,-6) | (7,-11) | 14.86606875 | 0.03125 | (9,-15) | 18.35755975 | 0.03125 |
| 68 | 7 | 8 | (-6,7) | (7,-11) | 22.20360331 | 0 | (-9,-6) | 13.34166406 | 0 |
| 69 | 7 | 9 | (-4,0) | (-6,-1) | 2.236067977 | 0 | (-2,-6) | 6.32455532 | 0.03125 |
| 70 | 7 | 10 | (5,9) | (-6,-1) | 14.86606875 | 0 | (-2,-6) | 16.55294536 | 0.03125 |
| 71 | 8 | 1 | (2,-7) | (9,-15) | 10.63014581 | 0 | (-2,-33) | 26.30589288 | 0 |
| 72 | 8 | 2 | (-9,8) | (-9,-6) | 14 | 0 | (-23,-18) | 29.52964612 | 0 |
| 73 | 8 | 3 | (0,-5) | (9,-15) | 13.45362405 | 0.03125 | (-2,-33) | 28.0713377 | 0 |
| 74 | 8 | 4 | (9,1) | (22,16) | 19.84943324 | 0.015625 | (10,-14) | 15.03329638 | 0 |
| 75 | 8 | 5 | (2,-4) | (9,-15) | 13.03840481 | 0 | (-2,-33) | 29.27456234 | 0.03125 |
| 76 | 8 | 6 | (5,8) | (-6,17) | 14.2126704 | 0.015625 | (-23,-18) | 38.20994635 | 0 |
| 77 | 8 | 7 | (-7,-6) | (-9,-6) | 2 | 0 | (-2,-33) | 27.45906044 | 0.015625 |
| 78 | 8 | 8 | (-6,7) | (-9,-6) | 13.34166406 | 0.03125 | (-23,-18) | 30.23243292 | 0 |
| 79 | 8 | 9 | (-4,0) | (-9,12) | 13 | 0 | (-2,-33) | 33.06055051 | 0.015625 |
| 80 | 8 | 10 | (5,9) | (-6,17) | 13.60147051 | 0.03125 | (-23,-18) | 38.89730068 | 0.015625 |
| 81 | 9 | 1 | (2,-7) | (-9,-23) | 19.41648784 | 0 | (-2,-33) | 26.30589288 | 0.015625 |
| 82 | 9 | 2 | (-9,8) | (-23,-18) | 29.52964612 | 0 | (-23,-18) | 29.52964612 | 0.03125 |
| 83 | 9 | 3 | (0,-5) | (-9,-23) | 20.1246118 | 0 | (-2,-33) | 28.0713377 | 0 |
| 84 | 9 | 4 | (9,1) | (-9,6) | 18.68154169 | 0.015625 | (10,-14) | 15.03329638 | 0.03125 |
| 85 | 9 | 5 | (2,-4) | (-9,-23) | 21.9544984 | 0 | (-2,-33) | 29.27456234 | 0 |
| 86 | 9 | 6 | (5,8) | (-9,6) | 14.14213562 | 0.015625 | (-23,-18) | 38.20994635 | 0.03125 |
| 87 | 9 | 7 | (-7,-6) | (-9,-23) | 17.11724277 | 0 | (-2,-33) | 27.45906044 | 0 |
| 88 | 9 | 8 | (-6,7) | (-9,-23) | 30.14962686 | 0.03125 | (-23,-18) | 30.23243292 | 0.046875 |
| 89 | 9 | 9 | (-4,0) | (-9,-23) | 23.53720459 | 0 | (-2,-33) | 33.06055051 | 0 |
| 90 | 9 | 10 | (5,9) | (-9,6) | 14.31782106 | 0.015625 | (-23,-18) | 38.89730068 | 0.03125 |
| 91 | 10 | 1 | (2,-7) | (-9,-23) | 19.41648784 | 0.03125 | (14,-37) | 32.31098884 | 0.03125 |
| 92 | 10 | 2 | (-9,8) | (-23,-18) | 29.52964612 | 0 | (-38,-12) | 35.22782991 | 0 |
| 93 | 10 | 3 | (0,-5) | (-9,-23) | 20.1246118 | 0 | (14,-37) | 34.92849839 | 0.03125 |
| 94 | 10 | 4 | (9,1) | (-9,6) | 18.68154169 | 0.015625 | (-3,-5) | 13.41640786 | 0 |
| 95 | 10 | 5 | (2,-4) | (-9,-23) | 21.9544984 | 0 | (-3,-5) | 5.099019514 | 0.03125 |
| 96 | 10 | 6 | (5,8) | (-9,6) | 14.14213562 | 0.015625 | (-3,-5) | 15.26433752 | 0.015625 |
| 97 | 10 | 7 | (-7,-6) | (-9,-23) | 17.11724277 | 0 | (14,-37) | 37.44329045 | 0.03125 |
| 98 | 10 | 8 | (-6,7) | (-9,-23) | 30.14962686 | 0.015625 | (-38,-12) | 37.21558813 | 0 |
| 99 | 10 | 9 | (-4,0) | (-9,-23) | 23.53720459 | 0 | (-3,-5) | 5.099019514 | 0.015625 |
| 100 | 10 | 10 | (5,9) | (-9,6) | 14.31782106 | 0.015625 | (-3,-5) | 16.1245155 | 0.03125 |

Fig. 14. Comparing PST and PQT in terms of time and distance for 100 input points

| | | | | Polar Split Tree (200) | | | Polar Quad Tree (200) | | |
|---|---|---|---|---|---|---|---|---|---|
| Row# | DataSet# | QuerySet# | QueryPoint | 1NN | Distance | Time(ns) | 1NN | Distance | Time(ns) |
| 1 | 1 | 1 | (2,-7) | (0,-11) | 4.472135955 | 0.015625 | (6,-43) | 36.22154055 | 0.03125 |
| 2 | 1 | 2 | (-9,8) | (0,-11) | 21.02379604 | 0 | (-29,-10) | 26.90724809 | 0 |
| 3 | 1 | 3 | (0,-5) | (0,-11) | 6 | 0.015625 | (6,-43) | 38.47076812 | 0.046875 |
| 4 | 1 | 4 | (9,1) | (-4,6) | 13.92838828 | 0 | (-29,-10) | 39.56008089 | 0 |
| 5 | 1 | 5 | (2,-4) | (0,-11) | 7.280109889 | 0.015625 | (6,-43) | 39.20459157 | 0.046875 |
| 6 | 1 | 6 | (5,8) | (0,-11) | 19.6468827 | 0 | (-29,-10) | 38.47076812 | 0 |
| 7 | 1 | 7 | (-7,-6) | (0,-11) | 8.602325267 | 0.015625 | (6,-43) | 39.2173431 | 0.03125 |
| 8 | 1 | 8 | (-6,7) | (0,-11) | 18.97366596 | 0.015625 | (-29,-10) | 28.60069929 | 0 |
| 9 | 1 | 9 | (-4,0) | (0,-11) | 11.70469991 | 0 | (-29,-10) | 26.92582404 | 0 |
| 10 | 1 | 10 | (5,9) | (0,-11) | 20.61552813 | 0 | (-29,-10) | 38.94868419 | 0 |
| 11 | 2 | 1 | (2,-7) | (-5,-15) | 10.63014581 | 0 | (9,-40) | 33.73425559 | 0 |
| 12 | 2 | 2 | (-9,8) | (-5,-15) | 23.34523506 | 0.015625 | (-74,-27) | 73.8241153 | 0.03125 |
| 13 | 2 | 3 | (0,-5) | (-5,-15) | 11.18033989 | 0 | (9,-40) | 36.138622 | 0.046875 |
| 14 | 2 | 4 | (9,1) | (-5,-15) | 21.26029163 | 0 | (-74,-27) | 87.59566199 | 0 |
| 15 | 2 | 5 | (2,-4) | (-5,-15) | 13.03840481 | 0.03125 | (9,-40) | 36.67424164 | 0.046875 |
| 16 | 2 | 6 | (5,8) | (-17,15) | 23.08679276 | 0.015625 | (-74,-27) | 86.40601831 | 0 |
| 17 | 2 | 7 | (-7,-6) | (-5,-15) | 9.219544457 | 0.015625 | (9,-40) | 37.57658846 | 0.03125 |
| 18 | 2 | 8 | (-6,7) | (-5,-15) | 22.02271555 | 0.015625 | (-74,-27) | 76.02631123 | 0 |
| 19 | 2 | 9 | (-4,0) | (-5,-15) | 15.03329638 | 0 | (-74,-27) | 75.02666193 | 0 |
| 20 | 2 | 10 | (5,9) | (-17,15) | 22.8035085 | 0 | (-74,-27) | 86.81589716 | 0.015625 |
| 21 | 3 | 1 | (2,-7) | (-14,-27) | 25.61249695 | 0.046875 | (9,-40) | 33.73425559 | 0 |
| 22 | 3 | 2 | (-9,8) | (-14,-27) | 35.35533906 | 0 | (-74,-27) | 73.8241153 | 0.015625 |
| 23 | 3 | 3 | (0,-5) | (-14,-27) | 26.07680962 | 0 | (9,-40) | 36.138622 | 0 |
| 24 | 3 | 4 | (9,1) | (9,-40) | 41 | 0.03125 | (-74,-27) | 87.59566199 | 0.03125 |
| 25 | 3 | 5 | (2,-4) | (-14,-27) | 28.01785145 | 0 | (9,-40) | 36.67424164 | 0 |
| 26 | 3 | 6 | (5,8) | (-22,32) | 36.12478374 | 0.03125 | (-74,-27) | 86.40601831 | 0 |
| 27 | 3 | 7 | (-7,-6) | (-14,-27) | 22.13594362 | 0 | (9,-40) | 37.57658846 | 0 |
| 28 | 3 | 8 | (-6,7) | (-14,-27) | 34.92849839 | 0 | (-74,-27) | 76.02631123 | 0 |
| 29 | 3 | 9 | (-4,0) | (-14,-27) | 28.7923601 | 0 | (-74,-27) | 75.02666193 | 0.03125 |
| 30 | 3 | 10 | (5,9) | (-22,32) | 35.4682957 | 0.03125 | (-74,-27) | 86.81589716 | 0 |
| 31 | 4 | 1 | (2,-7) | (-14,-27) | 25.61249695 | 0 | (13,-28) | 23.70653918 | 0 |
| 32 | 4 | 2 | (-9,8) | (-14,-27) | 35.35533906 | 0 | (-38,5) | 29.15475947 | 0 |
| 33 | 4 | 3 | (0,-5) | (-14,-27) | 26.07680962 | 0.015625 | (13,-28) | 26.41968963 | 0 |
| 34 | 4 | 4 | (9,1) | (9,-40) | 41 | 0.03125 | (-38,5) | 47.16990566 | 0 |
| 35 | 4 | 5 | (2,-4) | (-14,-27) | 28.01785145 | 0.046875 | (13,-28) | 26.40075756 | 0.03125 |
| 36 | 4 | 6 | (5,8) | (-22,32) | 36.12478374 | 0.015625 | (-38,5) | 43.10452412 | 0 |
| 37 | 4 | 7 | (-7,-6) | (-14,-27) | 22.13594362 | 0 | (-2,-32) | 26.47640459 | 0.03125 |
| 38 | 4 | 8 | (-6,7) | (-14,-27) | 34.92849839 | 0 | (-38,5) | 32.06243908 | 0 |
| 39 | 4 | 9 | (-4,0) | (-14,-27) | 28.7923601 | 0 | (-2,-32) | 32.06243908 | 0.015625 |
| 40 | 4 | 10 | (5,9) | (-22,32) | 35.4682957 | 0.03125 | (-38,5) | 43.18564576 | 0 |
| 41 | 5 | 1 | (2,-7) | (-2,-32) | 25.3179778 | 0.015625 | (16,-14) | 15.65247584 | 0.03125 |
| 42 | 5 | 2 | (-9,8) | (-9,1) | 7 | 0 | (-49,-11) | 44.28317965 | 0 |
| 43 | 5 | 3 | (0,-5) | (-2,-32) | 27.07397274 | 0.03125 | (16,-14) | 18.35755975 | 0.03125 |
| 44 | 5 | 4 | (9,1) | (-2,-32) | 34.78505426 | 0.03125 | (-49,-11) | 59.22837158 | 0 |
| 45 | 5 | 5 | (2,-4) | (-2,-32) | 28.28427125 | 0.015625 | (16,-14) | 17.20465053 | 0.03125 |
| 46 | 5 | 6 | (5,8) | (-2,-32) | 40.60788101 | 0.015625 | (-49,-11) | 57.24508713 | 0.03125 |
| 47 | 5 | 7 | (-7,-6) | (-2,-32) | 26.47640459 | 0.015625 | (16,-14) | 24.35159132 | 0.03125 |
| 48 | 5 | 8 | (-6,7) | (-2,-32) | 39.20459157 | 0.015625 | (-49,-11) | 46.61544808 | 0 |
| 49 | 5 | 9 | (-4,0) | (-2,-32) | 32.06243908 | 0.015625 | (-49,-11) | 46.32493929 | 0.015625 |
| 50 | 5 | 10 | (5,9) | (-2,-32) | 41.59326869 | 0.015625 | (-49,-11) | 57.5847202 | 0.03125 |

| | | | | Polar Split Tree (200) | | | Polar Quad Tree (200) | | |
|---|---|---|---|---|---|---|---|---|---|
| Row# | DataSet# | QuerySet# | QueryPoint | 1NN | Distance | Time(ns) | 1NN | Distance | Time(ns) |
| 51 | 6 | 1 | (2,-7) | (16,-14) | 15.65247584 | 0.015625 | (16,-14) | 15.65247584 | 0.03125 |
| 52 | 6 | 2 | (-9,8) | (16,-14) | 33.30165161 | 0.015625 | (-49,-11) | 44.28317965 | 0.03125 |
| 53 | 6 | 3 | (0,-5) | (16,-14) | 18.35755975 | 0 | (16,-14) | 18.35755975 | 0.03125 |
| 54 | 6 | 4 | (9,1) | (16,-14) | 16.55294536 | 0.015625 | (-49,-11) | 59.22837158 | 0.03125 |
| 55 | 6 | 5 | (2,-4) | (16,-14) | 17.20465053 | 0.015625 | (16,-14) | 17.20465053 | 0.03125 |
| 56 | 6 | 6 | (5,8) | (-12,15) | 18.38477631 | 0 | (-49,-11) | 57.24508713 | 0 |
| 57 | 6 | 7 | (-7,-6) | (16,-14) | 24.35159132 | 0.015625 | (16,-14) | 24.35159132 | 0.03125 |
| 58 | 6 | 8 | (-6,7) | (16,-14) | 30.41381265 | 0 | (-49,-11) | 46.61544808 | 0 |
| 59 | 6 | 9 | (-4,0) | (16,-14) | 24.41311123 | 0.015625 | (-49,-11) | 46.32493929 | 0.03125 |
| 60 | 6 | 10 | (5,9) | (-12,15) | 18.02775638 | 0.015625 | (-49,-11) | 57.5847202 | 0 |
| 61 | 7 | 1 | (2,-7) | (16,-14) | 15.65247584 | 0.015625 | (30,-45) | 47.20169488 | 0.015625 |
| 62 | 7 | 2 | (-9,8) | (16,-14) | 33.30165161 | 0.03125 | (-51,-2) | 43.17406629 | 0 |
| 63 | 7 | 3 | (0,-5) | (16,-14) | 18.35755975 | 0.015625 | (30,-45) | 50 | 0.03125 |
| 64 | 7 | 4 | (9,1) | (16,-14) | 16.55294536 | 0.03125 | (30,-45) | 50.56678752 | 0 |
| 65 | 7 | 5 | (2,-4) | (16,-14) | 17.20465053 | 0.015625 | (30,-45) | 49.64876635 | 0.03125 |
| 66 | 7 | 6 | (5,8) | (-12,15) | 18.38477631 | 0.015625 | (-51,-2) | 56.88585061 | 0 |
| 67 | 7 | 7 | (-7,-6) | (16,-14) | 24.35159132 | 0 | (30,-45) | 53.75872022 | 0.03125 |
| 68 | 7 | 8 | (-6,7) | (16,-14) | 30.41381265 | 0.015625 | (-51,-2) | 45.89117562 | 0 |
| 69 | 7 | 9 | (-4,0) | (16,-14) | 24.41311123 | 0.015625 | (30,-45) | 56.40035461 | 0.046875 |
| 70 | 7 | 10 | (5,9) | (-12,15) | 18.02775638 | 0 | (-51,-2) | 57.07013229 | 0 |
| 71 | 8 | 1 | (2,-7) | (-16,-33) | 31.6227766 | 0.015625 | (1,-15) | 8.062257748 | 0.03125 |
| 72 | 8 | 2 | (-9,8) | (-16,-33) | 41.59326869 | 0 | (-92,-18) | 86.97700846 | 0 |
| 73 | 8 | 3 | (0,-5) | (-16,-33) | 32.24903099 | 0 | (1,-15) | 10.04987562 | 0.03125 |
| 74 | 8 | 4 | (9,1) | (-16,-33) | 42.20189569 | 0.03125 | (-42,-29) | 59.16924877 | 0 |
| 75 | 8 | 5 | (2,-4) | (-16,-33) | 34.13209633 | 0.03125 | (1,-15) | 11.04536102 | 0.046875 |
| 76 | 8 | 6 | (5,8) | (-42,5) | 47.09564736 | 0.015625 | (-92,-18) | 100.4241007 | 0 |
| 77 | 8 | 7 | (-7,-6) | (-16,-33) | 28.46049894 | 0 | (1,-15) | 12.04159458 | 0.03125 |
| 78 | 8 | 8 | (-6,7) | (-16,-33) | 41.23105626 | 0.015625 | (-92,-18) | 89.56003573 | 0 |
| 79 | 8 | 9 | (-4,0) | (-16,-33) | 35.11409973 | 0 | (-92,-18) | 89.82204629 | 0 |
| 80 | 8 | 10 | (5,9) | (-42,5) | 47.16990566 | 0.015625 | (-92,-18) | 100.6876358 | 0 |
| 81 | 9 | 1 | (2,-7) | (1,-15) | 8.062257748 | 0.03125 | (1,-15) | 8.062257748 | 0.046875 |
| 82 | 9 | 2 | (-9,8) | (1,-15) | 25.07987241 | 0 | (-92,-18) | 86.97700846 | 0 |
| 83 | 9 | 3 | (0,-5) | (1,-15) | 10.04987562 | 0 | (1,-15) | 10.04987562 | 0.03125 |
| 84 | 9 | 4 | (9,1) | (1,11) | 12.80624847 | 0.046875 | (-42,-29) | 59.16924877 | 0.03125 |
| 85 | 9 | 5 | (2,-4) | (1,-15) | 11.04536102 | 0 | (1,-15) | 11.04536102 | 0.03125 |
| 86 | 9 | 6 | (5,8) | (1,-15) | 23.34523506 | 0 | (-92,-18) | 100.4241007 | 0.03125 |
| 87 | 9 | 7 | (-7,-6) | (1,-15) | 12.04159458 | 0.015625 | (1,-15) | 12.04159458 | 0 |
| 88 | 9 | 8 | (-6,7) | (1,-15) | 23.08679276 | 0.015625 | (-92,-18) | 89.56003573 | 0.015625 |
| 89 | 9 | 9 | (-4,0) | (1,-15) | 15.8113883 | 0 | (-92,-18) | 89.82204629 | 0 |
| 90 | 9 | 10 | (5,9) | (1,-15) | 24.33105012 | 0.015625 | (-92,-18) | 100.6876358 | 0.03125 |
| 91 | 10 | 1 | (2,-7) | (1,-15) | 8.062257748 | 0 | (9,-16) | 11.40175425 | 0.03125 |
| 92 | 10 | 2 | (-9,8) | (1,-15) | 25.07987241 | 0.03125 | (-41,-49) | 65.36818798 | 0.03125 |
| 93 | 10 | 3 | (0,-5) | (1,-15) | 10.04987562 | 0.015625 | (9,-16) | 14.2126704 | 0 |
| 94 | 10 | 4 | (9,1) | (1,11) | 12.80624847 | 0.046875 | (9,-16) | 17 | 0.03125 |
| 95 | 10 | 5 | (2,-4) | (1,-15) | 11.04536102 | 0 | (9,-16) | 13.89244399 | 0.015625 |
| 96 | 10 | 6 | (5,8) | (1,-15) | 23.34523506 | 0.015625 | (-41,-49) | 73.24616031 | 0.03125 |
| 97 | 10 | 7 | (-7,-6) | (1,-15) | 12.04159458 | 0.015625 | (9,-16) | 18.86796226 | 0.015625 |
| 98 | 10 | 8 | (-6,7) | (1,-15) | 23.08679276 | 0 | (-41,-49) | 66.03786792 | 0.015625 |
| 99 | 10 | 9 | (-4,0) | (1,-15) | 15.8113883 | 0.03125 | (9,-16) | 20.61552813 | 0 |
| 100 | 10 | 10 | (5,9) | (1,-15) | 24.33105012 | 0 | (-41,-49) | 74.02702209 | 0.03125 |

Fig. 15. Comparing PST and PQT in terms of time and distance for 200 input points

| | | | | Polar Split Tree (500) | | | Polar Quad Tree (500) | | |
|---|---|---|---|---|---|---|---|---|---|
| Row# | DataSet# | QuerySet# | QueryPoint | 1NN | Distance | Time(ns) | 1NN | Distance | Time(ns) |
| 1 | 1 | 1 | (2,-7) | (-48,3) | 50.99019514 | 0.03125 | (-63,-12) | 65.19202405 | 0.03125 |
| 2 | 1 | 2 | (-9,8) | (-48,3) | 39.3192065 | 0.03125 | (-63,-12) | 57.5847202 | 0 |
| 3 | 1 | 3 | (0,-5) | (-48,3) | 48.66210024 | 0 | (-63,-12) | 63.38769597 | 0 |
| 4 | 1 | 4 | (9,1) | (-23,50) | 58.52349955 | 0.046875 | (-63,-12) | 73.1641989 | 0.0625 |
| 5 | 1 | 5 | (2,-4) | (-48,3) | 50.48762225 | 0.015625 | (-63,-12) | 65.49045732 | 0 |
| 6 | 1 | 6 | (5,8) | (-48,3) | 53.23532662 | 0.015625 | (-63,-12) | 70.88018059 | 0 |
| 7 | 1 | 7 | (-7,-6) | (-48,3) | 41.97618372 | 0 | (-63,-12) | 56.32051136 | 0 |
| 8 | 1 | 8 | (-6,7) | (-48,3) | 42.19004622 | 0 | (-63,-12) | 60.08327554 | 0 |
| 9 | 1 | 9 | (-4,0) | (-48,3) | 44.10215414 | 0 | (-63,-12) | 60.20797289 | 0 |
| 10 | 1 | 10 | (5,9) | (-48,3) | 53.33854141 | 0.015625 | (-63,-12) | 71.16881339 | 0 |
| 11 | 2 | 1 | (2,-7) | (-63,-12) | 65.19202405 | 0 | (-4,-62) | 55.32630477 | 0.03125 |
| 12 | 2 | 2 | (-9,8) | (-63,-12) | 57.5847202 | 0 | (-4,-62) | 70.17834424 | 0 |
| 13 | 2 | 3 | (0,-5) | (-63,-12) | 63.38769597 | 0.03125 | (-4,-62) | 57.14017851 | 0 |
| 14 | 2 | 4 | (9,1) | (-63,-12) | 73.1641989 | 0 | (-4,-62) | 64.32728814 | 0.03125 |
| 15 | 2 | 5 | (2,-4) | (-63,-12) | 65.49045732 | 0 | (-4,-62) | 58.30951895 | 0 |
| 16 | 2 | 6 | (5,8) | (-63,-12) | 70.88018059 | 0 | (-4,-62) | 70.57619995 | 0 |
| 17 | 2 | 7 | (-7,-6) | (-63,-12) | 56.32051136 | 0 | (-4,-62) | 56.08029957 | 0 |
| 18 | 2 | 8 | (-6,7) | (-63,-12) | 60.08327554 | 0 | (-4,-62) | 69.02897942 | 0 |
| 19 | 2 | 9 | (-4,0) | (-63,-12) | 60.20797289 | 0 | (-4,-62) | 62 | 0 |
| 20 | 2 | 10 | (5,9) | (-63,-12) | 71.16881339 | 0 | (-4,-62) | 71.56814934 | 0 |
| 21 | 3 | 1 | (2,-7) | (-48,-60) | 72.86288493 | 0.015625 | (-11,-25) | 22.20360331 | 0.03125 |
| 22 | 3 | 2 | (-9,8) | (-48,-60) | 78.39005039 | 0.015625 | (-11,-25) | 33.06055051 | 0 |
| 23 | 3 | 3 | (0,-5) | (-48,-60) | 73 | 0 | (-11,-25) | 22.82542442 | 0.015625 |
| 24 | 3 | 4 | (9,1) | (-48,-60) | 83.48652586 | 0.015625 | (-11,-25) | 32.80243893 | 0.046875 |
| 25 | 3 | 5 | (2,-4) | (-48,-60) | 75.07329752 | 0.015625 | (-11,-25) | 24.69817807 | 0.03125 |
| 26 | 3 | 6 | (5,8) | (-48,-60) | 86.21484791 | 0 | (-11,-25) | 36.67424164 | 0.03125 |
| 27 | 3 | 7 | (-7,-6) | (-48,-60) | 67.80117993 | 0.015625 | (-11,-25) | 19.41648784 | 0.03125 |
| 28 | 3 | 8 | (-6,7) | (-48,-60) | 79.07591289 | 0.015625 | (-11,-25) | 32.38826948 | 0 |
| 29 | 3 | 9 | (-4,0) | (-48,-60) | 74.40430095 | 0.015625 | (-11,-25) | 25.96150997 | 0.03125 |
| 30 | 3 | 10 | (5,9) | (-48,-60) | 87.00574694 | 0.015625 | (-11,-25) | 37.57658846 | 0 |
| 31 | 4 | 1 | (2,-7) | (-11,-25) | 22.20360331 | 0.015625 | (24,-5) | 22.09072203 | 0.03125 |
| 32 | 4 | 2 | (-9,8) | (-11,-25) | 33.06055051 | 0.015625 | (24,-5) | 35.4682957 | 0.03125 |
| 33 | 4 | 3 | (0,-5) | (-11,-25) | 22.82542442 | 0 | (24,-5) | 24 | 0.015625 |
| 34 | 4 | 4 | (9,1) | (-22,33) | 44.55333882 | 0 | (24,-5) | 16.15549442 | 0 |
| 35 | 4 | 5 | (2,-4) | (-11,-25) | 24.69817807 | 0 | (24,-5) | 22.02271555 | 0.03125 |
| 36 | 4 | 6 | (5,8) | (-22,33) | 36.79673899 | 0.015625 | (24,-5) | 23.02172887 | 0.046875 |
| 37 | 4 | 7 | (-7,-6) | (-11,-25) | 19.41648784 | 0.015625 | (24,-5) | 31.01612484 | 0.03125 |
| 38 | 4 | 8 | (-6,7) | (-11,-25) | 32.38826948 | 0.015625 | (24,-5) | 32.31098884 | 0.03125 |
| 39 | 4 | 9 | (-4,0) | (-11,-25) | 25.96150997 | 0.015625 | (24,-5) | 28.44292531 | 0.03125 |
| 40 | 4 | 10 | (5,9) | (-22,33) | 36.12478374 | 0 | (24,-5) | 23.60084744 | 0.03125 |
| 41 | 5 | 1 | (2,-7) | (24,-5) | 22.09072203 | 0 | (-16,-18) | 21.09502311 | 0 |
| 42 | 5 | 2 | (-9,8) | (-43,11) | 34.13209633 | 0.046875 | (-16,-18) | 26.92582404 | 0 |
| 43 | 5 | 3 | (0,-5) | (24,-5) | 24 | 0.015625 | (-16,-18) | 20.61552813 | 0.03125 |
| 44 | 5 | 4 | (9,1) | (24,-5) | 16.15549442 | 0.015625 | (-16,-18) | 31.40063694 | 0.03125 |
| 45 | 5 | 5 | (2,-4) | (24,-5) | 22.02271555 | 0.015625 | (-16,-18) | 22.8035085 | 0.015625 |
| 46 | 5 | 6 | (5,8) | (24,-5) | 23.02172887 | 0.015625 | (-16,-18) | 33.42154993 | 0.015625 |
| 47 | 5 | 7 | (-7,-6) | (24,-5) | 31.01612484 | 0.015625 | (-16,-18) | 15 | 0.03125 |
| 48 | 5 | 8 | (-6,7) | (24,-5) | 32.31098884 | 0.015625 | (-16,-18) | 26.92582404 | 0.03125 |
| 49 | 5 | 9 | (-4,0) | (24,-5) | 28.44292531 | 0.015625 | (-16,-18) | 21.63330765 | 0.015625 |
| 50 | 5 | 10 | (5,9) | (24,-5) | 23.60084744 | 0 | (-16,-18) | 34.20526275 | 0 |

| | | | | Polar Split Tree (500) | | | Polar Quad Tree (500) | | |
|---|---|---|---|---|---|---|---|---|---|
| Row# | DataSet# | QuerySet# | QueryPoint | 1NN | Distance | Time(ns) | 1NN | Distance | Time(ns) |
| 51 | 6 | 1 | (2,-7) | (1,-8) | 1.414213562 | 0.015625 | (21,-70) | 65.80273551 | 0 |
| 52 | 6 | 2 | (-9,8) | (1,-8) | 18.86796226 | 0.015625 | (-58,-49) | 75.16648189 | 0.03125 |
| 53 | 6 | 3 | (0,-5) | (1,-8) | 3.16227766 | 0.015625 | (-58,-49) | 72.80109889 | 0.03125 |
| 54 | 6 | 4 | (9,1) | (1,-8) | 12.04159458 | 0.015625 | (-58,-49) | 83.60023923 | 0.03125 |
| 55 | 6 | 5 | (2,-4) | (1,-8) | 4.123105626 | 0.015625 | (-58,-49) | 75 | 0 |
| 56 | 6 | 6 | (5,8) | (1,-8) | 16.4924225 | 0.03125 | (-58,-49) | 84.95881355 | 0.03125 |
| 57 | 6 | 7 | (-7,-6) | (1,-8) | 8.246211251 | 0 | (21,-70) | 69.85699679 | 0.03125 |
| 58 | 6 | 8 | (-6,7) | (1,-8) | 16.55294536 | 0.015625 | (-58,-49) | 76.4198927 | 0.03125 |
| 59 | 6 | 9 | (-4,0) | (1,-8) | 9.433981132 | 0.015625 | (-58,-49) | 72.9177619 | 0.015625 |
| 60 | 6 | 10 | (5,9) | (1,-8) | 17.4642492 | 0.03125 | (-58,-49) | 85.63293759 | 0 |
| 61 | 7 | 1 | (2,-7) | (17,2) | 17.49285568 | 0.03125 | (-102,-15) | 104.3072385 | 0 |
| 62 | 7 | 2 | (-9,8) | (17,2) | 26.68332813 | 0 | (-102,-15) | 95.8018789 | 0 |
| 63 | 7 | 3 | (0,-5) | (17,2) | 18.38477631 | 0 | (-102,-15) | 102.4890238 | 0.03125 |
| 64 | 7 | 4 | (9,1) | (17,2) | 8.062257748 | 0.046875 | (-102,-15) | 112.1472247 | 0.015625 |
| 65 | 7 | 5 | (2,-4) | (17,2) | 16.15549442 | 0.015625 | (-102,-15) | 104.5801128 | 0 |
| 66 | 7 | 6 | (5,8) | (17,2) | 13.41640786 | 0.015625 | (-102,-15) | 109.4440496 | 0 |
| 67 | 7 | 7 | (-7,-6) | (17,2) | 25.29822128 | 0.03125 | (-102,-15) | 95.4253635 | 0.046875 |
| 68 | 7 | 8 | (-6,7) | (17,2) | 23.53720459 | 0.015625 | (-102,-15) | 98.48857802 | 0.03125 |
| 69 | 7 | 9 | (-4,0) | (17,2) | 21.09502311 | 0 | (-102,-15) | 99.14131329 | 0.03125 |
| 70 | 7 | 10 | (5,9) | (17,2) | 13.89244399 | 0.015625 | (-102,-15) | 109.658561 | 0 |
| 71 | 8 | 1 | (2,-7) | (-24,-39) | 41.23105626 | 0.015625 | (-19,-38) | 37.44329045 | 0 |
| 72 | 8 | 2 | (-9,8) | (-24,-39) | 49.33558553 | 0.015625 | (-19,-38) | 47.07440918 | 0 |
| 73 | 8 | 3 | (0,-5) | (-24,-39) | 41.61730409 | 0.015625 | (-19,-38) | 38.07886553 | 0 |
| 74 | 8 | 4 | (9,1) | (-24,-39) | 51.85556865 | 0.015625 | (-19,-38) | 48.01041554 | 0.03125 |
| 75 | 8 | 5 | (2,-4) | (-24,-39) | 43.60045871 | 0.015625 | (-19,-38) | 39.96248241 | 0.03125 |
| 76 | 8 | 6 | (5,8) | (-24,-39) | 55.22680509 | 0.03125 | (-19,-38) | 51.88448708 | 0 |
| 77 | 8 | 7 | (-7,-6) | (-24,-39) | 37.12142239 | 0.015625 | (-19,-38) | 34.17601498 | 0 |
| 78 | 8 | 8 | (-6,7) | (-24,-39) | 49.39635614 | 0.015625 | (-19,-38) | 46.84015371 | 0 |
| 79 | 8 | 9 | (-4,0) | (-24,-39) | 43.829214 | 0.015625 | (-19,-38) | 40.85339643 | 0.015625 |
| 80 | 8 | 10 | (5,9) | (-24,-39) | 56.08029957 | 0.015625 | (-19,-38) | 52.77309921 | 0.015625 |
| 81 | 9 | 1 | (2,-7) | (5,-26) | 19.23538406 | 0.015625 | (8,-52) | 45.39823785 | 0 |
| 82 | 9 | 2 | (-9,8) | (5,-26) | 36.76955262 | 0.015625 | (-67,-15) | 62.39390996 | 0 |
| 83 | 9 | 3 | (0,-5) | (5,-26) | 21.58703314 | 0.015625 | (-67,-15) | 67.74215822 | 0 |
| 84 | 9 | 4 | (9,1) | (-32,35) | 53.26349594 | 0.015625 | (-67,-15) | 77.66595136 | 0.03125 |
| 85 | 9 | 5 | (2,-4) | (5,-26) | 22.20360331 | 0.015625 | (-67,-15) | 69.87131028 | 0.03125 |
| 86 | 9 | 6 | (5,8) | (-32,35) | 45.80392996 | 0.015625 | (-67,-15) | 75.58438992 | 0.046875 |
| 87 | 9 | 7 | (-7,-6) | (5,-26) | 23.32380758 | 0.015625 | (8,-52) | 48.38388161 | 0 |
| 88 | 9 | 8 | (-6,7) | (5,-26) | 34.78505426 | 0.015625 | (-67,-15) | 64.84597135 | 0 |
| 89 | 9 | 9 | (-4,0) | (5,-26) | 27.51363298 | 0 | (-67,-15) | 64.76109943 | 0 |
| 90 | 9 | 10 | (5,9) | (-32,35) | 45.22167622 | 0.015625 | (-67,-15) | 75.89466384 | 0.03125 |
| 91 | 10 | 1 | (2,-7) | (25,-22) | 27.45906044 | 0 | (-121,-43) | 128.1600562 | 0 |
| 92 | 10 | 2 | (-9,8) | (25,-22) | 45.3431362 | 0.046875 | (-121,-43) | 123.0650235 | 0 |
| 93 | 10 | 3 | (0,-5) | (25,-22) | 30.23243292 | 0 | (-121,-43) | 126.8266534 | 0 |
| 94 | 10 | 4 | (9,1) | (25,-22) | 28.01785145 | 0 | (-121,-43) | 137.2443077 | 0 |
| 95 | 10 | 5 | (2,-4) | (25,-22) | 29.20616373 | 0 | (-121,-43) | 129.034879 | 0 |
| 96 | 10 | 6 | (5,8) | (25,-22) | 36.05551275 | 0 | (-121,-43) | 135.9301291 | 0.03125 |
| 97 | 10 | 7 | (-7,-6) | (25,-22) | 35.77708764 | 0.03125 | (-121,-43) | 119.8540779 | 0.03125 |
| 98 | 10 | 8 | (-6,7) | (25,-22) | 42.44997055 | 0 | (-121,-43) | 125.399362 | 0 |
| 99 | 10 | 9 | (-4,0) | (25,-22) | 36.40054945 | 0.015625 | (-121,-43) | 124.6515142 | 0 |
| 100 | 10 | 10 | (5,9) | (25,-22) | 36.89173349 | 0 | (-121,-43) | 136.3084737 | 0 |

Fig. 16. Comparing PST and PQT in terms of time and distance for 500 input points

| | | | | Polar Split Tree (1000) | | | Polar Quad Tree (1000) | | |
|---|---|---|---|---|---|---|---|---|---|
| Row# | DataSet# | QuerySet# | QueryPoint | 1NN | Distance | Time(ns) | 1NN | Distance | Time(ns) |
| 1 | 1 | 1 | (2,-7) | (10,-28) | 22.47220505 | 0.046875 | (18,-74) | 68.8839604 | 0.046875 |
| 2 | 1 | 2 | (-9,8) | (10,-28) | 40.70626487 | 0 | (-147,4) | 138.0579588 | 0 |
| 3 | 1 | 3 | (0,-5) | (10,-28) | 25.07987241 | 0.015625 | (18,-74) | 71.30918594 | 0.03125 |
| 4 | 1 | 4 | (9,1) | (10,-28) | 29.01723626 | 0 | (-147,4) | 156.0288435 | 0 |
| 5 | 1 | 5 | (2,-4) | (10,-28) | 25.29822128 | 0 | (18,-74) | 71.80529228 | 0.03125 |
| 6 | 1 | 6 | (5,8) | (10,-28) | 36.34556369 | 0 | (-147,4) | 152.0526225 | 0.03125 |
| 7 | 1 | 7 | (-7,-6) | (10,-28) | 27.80287755 | 0.03125 | (18,-74) | 72.44998275 | 0.03125 |
| 8 | 1 | 8 | (-6,7) | (10,-28) | 38.48376281 | 0.015625 | (-147,4) | 141.0319113 | 0 |
| 9 | 1 | 9 | (-4,0) | (10,-28) | 31.30495168 | 0 | (-147,4) | 143.0559331 | 0.015625 |
| 10 | 1 | 10 | (5,9) | (10,-28) | 37.33630941 | 0 | (-147,4) | 152.0822146 | 0 |
| 11 | 2 | 1 | (2,-7) | (18,-23) | 22.627417 | 0.046875 | (21,-129) | 123.4706443 | 0.015625 |
| 12 | 2 | 2 | (-9,8) | (18,-23) | 41.10960958 | 0.015625 | (-213,-78) | 221.3865398 | 0 |
| 13 | 2 | 3 | (0,-5) | (18,-23) | 25.45584412 | 0 | (21,-129) | 125.7656551 | 0.03125 |
| 14 | 2 | 4 | (9,1) | (18,-23) | 25.63201124 | 0.015625 | (21,-129) | 130.5526714 | 0 |
| 15 | 2 | 5 | (2,-4) | (18,-23) | 24.8394847 | 0.015625 | (21,-129) | 126.4357544 | 0 |
| 16 | 2 | 6 | (5,8) | (18,-23) | 33.61547263 | 0.015625 | (-213,-78) | 234.3501654 | 0.015625 |
| 17 | 2 | 7 | (-7,-6) | (18,-23) | 30.23243292 | 0.03125 | (21,-129) | 126.1467399 | 0 |
| 18 | 2 | 8 | (-6,7) | (18,-23) | 38.41874542 | 0.015625 | (-213,-78) | 223.7722056 | 0.03125 |
| 19 | 2 | 9 | (-4,0) | (18,-23) | 31.82766093 | 0.03125 | (21,-129) | 131.4001522 | 0 |
| 20 | 2 | 10 | (5,9) | (18,-23) | 34.53983208 | 0.015625 | (-213,-78) | 234.7189809 | 0 |
| 21 | 3 | 1 | (2,-7) | (-17,-57) | 53.48831648 | 0.015625 | (-100,-70) | 119.8874472 | 0 |
| 22 | 3 | 2 | (-9,8) | (-64,-3) | 56.08921465 | 0 | (-100,-70) | 119.8540779 | 0.03125 |
| 23 | 3 | 3 | (0,-5) | (-17,-57) | 54.70831747 | 0 | (-100,-70) | 119.2686044 | 0 |
| 24 | 3 | 4 | (9,1) | (-17,-57) | 63.56099433 | 0 | (-100,-70) | 130.0845879 | 0 |
| 25 | 3 | 5 | (2,-4) | (-17,-57) | 56.30275304 | 0.015625 | (-100,-70) | 121.4907404 | 0 |
| 26 | 3 | 6 | (5,8) | (-17,-57) | 68.62215386 | 0 | (-100,-70) | 130.8013761 | 0 |
| 27 | 3 | 7 | (-7,-6) | (-17,-57) | 51.97114584 | 0 | (-100,-70) | 112.8937554 | 0 |
| 28 | 3 | 8 | (-6,7) | (-64,-3) | 58.85575588 | 0.015625 | (-100,-70) | 121.5113163 | 0 |
| 29 | 3 | 9 | (-4,0) | (-17,-57) | 58.46366393 | 0.015625 | (-100,-70) | 118.8107739 | 0 |
| 30 | 3 | 10 | (5,9) | (-17,-57) | 69.57010852 | 0.015625 | (-100,-70) | 131.4001522 | 0 |
| 31 | 4 | 1 | (2,-7) | (11,-33) | 27.51363298 | 0 | (31,-43) | 46.22769733 | 0 |
| 32 | 4 | 2 | (-9,8) | (11,-33) | 45.61797891 | 0.015625 | (-137,-35) | 135.0296264 | 0 |
| 33 | 4 | 3 | (0,-5) | (11,-33) | 30.08321791 | 0.03125 | (-137,-35) | 140.2462121 | 0.03125 |
| 34 | 4 | 4 | (9,1) | (11,-33) | 34.05877273 | 0 | (-137,-35) | 150.3728699 | 0 |
| 35 | 4 | 5 | (2,-4) | (11,-33) | 30.3644529 | 0.015625 | (-137,-35) | 142.4148869 | 0 |
| 36 | 4 | 6 | (5,8) | (11,-33) | 41.43669871 | 0 | (-137,-35) | 148.3677863 | 0 |
| 37 | 4 | 7 | (-7,-6) | (11,-33) | 32.44996148 | 0.015625 | (31,-43) | 53.03772242 | 0 |
| 38 | 4 | 8 | (-6,7) | (11,-33) | 43.46262762 | 0 | (-137,-35) | 137.5681649 | 0 |
| 39 | 4 | 9 | (-4,0) | (11,-33) | 36.24913792 | 0.015625 | (-137,-35) | 137.5281789 | 0 |
| 40 | 4 | 10 | (5,9) | (11,-33) | 42.42640687 | 0.015625 | (-137,-35) | 148.6606875 | 0 |
| 41 | 5 | 1 | (2,-7) | (31,-43) | 46.22769733 | 0.046875 | (-11,-6) | 13.03840481 | 0.015625 |
| 42 | 5 | 2 | (-9,8) | (31,-43) | 64.81512169 | 0.015625 | (-11,-6) | 14.14213562 | 0 |
| 43 | 5 | 3 | (0,-5) | (31,-43) | 49.04079934 | 0.03125 | (-11,-6) | 11.04536102 | 0.03125 |
| 44 | 5 | 4 | (9,1) | (31,-43) | 49.1934955 | 0.0625 | (-11,-6) | 21.1896201 | 0.015625 |
| 45 | 5 | 5 | (2,-4) | (31,-43) | 48.60041152 | 0.03125 | (-11,-6) | 13.15294644 | 0.015625 |
| 46 | 5 | 6 | (5,8) | (31,-43) | 57.24508713 | 0.03125 | (-11,-6) | 21.26029163 | 0.015625 |
| 47 | 5 | 7 | (-7,-6) | (31,-43) | 53.03772242 | 0.046875 | (-11,-6) | 4 | 0 |
| 48 | 5 | 8 | (-6,7) | (31,-43) | 62.20128616 | 0.015625 | (-11,-6) | 13.92838828 | 0.03125 |
| 49 | 5 | 9 | (-4,0) | (31,-43) | 55.4436651 | 0.015625 | (-11,-6) | 9.219544457 | 0.03125 |
| 50 | 5 | 10 | (5,9) | (31,-43) | 58.13776741 | 0.03125 | (-11,-6) | 21.9317122 | 0.03125 |

| | | | | Polar Split Tree (1000) | | | Polar Quad Tree (1000) | | |
|---|---|---|---|---|---|---|---|---|---|
| Row# | DataSet# | QuerySet# | QueryPoint | 1NN | Distance | Time(ns) | 1NN | Distance | Time(ns) |
| 51 | 6 | 1 | (2,-7) | (-11,-6) | 13.03840481 | 0.015625 | (38,-140) | 137.7860661 | 0.03125 |
| 52 | 6 | 2 | (-9,8) | (-11,-6) | 14.14213562 | 0 | (-192,-53) | 192.8989373 | 0.015625 |
| 53 | 6 | 3 | (0,-5) | (-11,-6) | 11.04536102 | 0 | (38,-140) | 140.2462121 | 0.03125 |
| 54 | 6 | 4 | (9,1) | (-11,-6) | 21.1896201 | 0 | (-192,-53) | 208.1273649 | 0.03125 |
| 55 | 6 | 5 | (2,-4) | (-11,-6) | 13.15294644 | 0.015625 | (38,-140) | 140.6840432 | 0.015625 |
| 56 | 6 | 6 | (5,8) | (-11,-6) | 21.26029163 | 0 | (-192,-53) | 206.2280291 | 0.015625 |
| 57 | 6 | 7 | (-7,-6) | (-11,-6) | 4 | 0 | (38,-140) | 141.3541651 | 0.015625 |
| 58 | 6 | 8 | (-6,7) | (-11,-6) | 13.92838828 | 0 | (-192,-53) | 195.4379697 | 0 |
| 59 | 6 | 9 | (-4,0) | (-11,-6) | 9.219544457 | 0 | (-192,-53) | 195.3279294 | 0 |
| 60 | 6 | 10 | (5,9) | (-11,-6) | 21.9317122 | 0.015625 | (-192,-53) | 206.5260274 | 0.046875 |
| 61 | 7 | 1 | (2,-7) | (-46,-104) | 108.2266141 | 0.015625 | (13,-180) | 173.3493582 | 0.03125 |
| 62 | 7 | 2 | (-9,8) | (-46,-104) | 117.9533806 | 0 | (13,-180) | 189.2828571 | 0 |
| 63 | 7 | 3 | (0,-5) | (-46,-104) | 109.1650127 | 0.03125 | (13,-180) | 175.4821928 | 0 |
| 64 | 7 | 4 | (9,1) | (-56,93) | 112.6454615 | 0.03125 | (13,-180) | 181.0441935 | 0 |
| 65 | 7 | 5 | (2,-4) | (-46,-104) | 110.923397 | 0.03125 | (13,-180) | 176.343415 | 0 |
| 66 | 7 | 6 | (5,8) | (-56,93) | 104.6231332 | 0.015625 | (13,-180) | 188.1701358 | 0 |
| 67 | 7 | 7 | (-7,-6) | (-46,-104) | 105.4751155 | 0 | (13,-180) | 175.1456537 | 0.03125 |
| 68 | 7 | 8 | (-6,7) | (-56,93) | 99.47864092 | 0.015625 | (13,-180) | 187.9627623 | 0.03125 |
| 69 | 7 | 9 | (-4,0) | (-56,93) | 106.5504575 | 0 | (13,-180) | 180.8009956 | 0 |
| 70 | 7 | 10 | (5,9) | (-56,93) | 103.8123307 | 0.015625 | (13,-180) | 189.1692364 | 0 |
| 71 | 8 | 1 | (2,-7) | (-32,-70) | 71.58910532 | 0.03125 | (47,-82) | 87.46427842 | 0 |
| 72 | 8 | 2 | (-9,8) | (-86,5) | 77.0584194 | 0.015625 | (-109,-39) | 110.4943437 | 0 |
| 73 | 8 | 3 | (0,-5) | (-86,5) | 86.57944329 | 0 | (47,-82) | 90.21086409 | 0.03125 |
| 74 | 8 | 4 | (9,1) | (-32,-70) | 81.98780397 | 0 | (-109,-39) | 124.595345 | 0.03125 |
| 75 | 8 | 5 | (2,-4) | (-32,-70) | 74.24284477 | 0 | (47,-82) | 90.04998612 | 0.03125 |
| 76 | 8 | 6 | (5,8) | (-32,-70) | 86.33075929 | 0.015625 | (-109,-39) | 123.3085561 | 0 |
| 77 | 8 | 7 | (-7,-6) | (-86,5) | 79.76214641 | 0 | (47,-82) | 93.23089617 | 0.03125 |
| 78 | 8 | 8 | (-6,7) | (-86,5) | 80.02499609 | 0.03125 | (-109,-39) | 112.8051417 | 0.03125 |
| 79 | 8 | 9 | (-4,0) | (-86,5) | 82.15229759 | 0.015625 | (-109,-39) | 112.0089282 | 0 |
| 80 | 8 | 10 | (5,9) | (-32,-70) | 87.23531395 | 0 | (-109,-39) | 123.6931688 | 0 |
| 81 | 9 | 1 | (2,-7) | (-57,49) | 81.34494453 | 0.046875 | (34,-128) | 125.1598977 | 0 |
| 82 | 9 | 2 | (-9,8) | (-57,49) | 63.12685641 | 0 | (1,-4) | 15.62049935 | 0.015625 |
| 83 | 9 | 3 | (0,-5) | (-57,49) | 78.51751397 | 0 | (1,-4) | 1.414213562 | 0 |
| 84 | 9 | 4 | (9,1) | (-57,49) | 81.60882305 | 0.015625 | (1,-4) | 9.433981132 | 0.015625 |
| 85 | 9 | 5 | (2,-4) | (-57,49) | 79.30952024 | 0.015625 | (1,-4) | 1 | 0.015625 |
| 86 | 9 | 6 | (5,8) | (-57,49) | 74.33034374 | 0.015625 | (1,-4) | 12.64911064 | 0 |
| 87 | 9 | 7 | (-7,-6) | (-57,49) | 74.33034374 | 0.015625 | (1,-4) | 8.246211251 | 0.03125 |
| 88 | 9 | 8 | (-6,7) | (-57,49) | 66.06814664 | 0 | (1,-4) | 13.03840481 | 0.03125 |
| 89 | 9 | 9 | (-4,0) | (-57,49) | 72.18032973 | 0 | (1,-4) | 6.403124237 | 0.03125 |
| 90 | 9 | 10 | (5,9) | (-57,49) | 73.78346698 | 0.015625 | (1,-4) | 13.60147051 | 0 |
| 91 | 10 | 1 | (2,-7) | (-45,35) | 63.03173804 | 0 | (-152,-101) | 180.4217282 | 0.03125 |
| 92 | 10 | 2 | (-9,8) | (-45,35) | 45 | 0 | (-256,-8) | 247.5176761 | 0 |
| 93 | 10 | 3 | (0,-5) | (-45,35) | 60.20797289 | 0 | (-152,-101) | 179.7776404 | 0.03125 |
| 94 | 10 | 4 | (9,1) | (64,-25) | 60.8358447 | 0.015625 | (-152,-101) | 190.5911855 | 0 |
| 95 | 10 | 5 | (2,-4) | (-45,35) | 61.07372594 | 0 | (-152,-101) | 182.0027472 | 0 |
| 96 | 10 | 6 | (5,8) | (-45,35) | 56.82429058 | 0.015625 | (-152,-101) | 191.1282292 | 0.03125 |
| 97 | 10 | 7 | (-7,-6) | (-45,35) | 55.90169944 | 0 | (-152,-101) | 173.3493582 | 0.03125 |
| 98 | 10 | 8 | (-6,7) | (-45,35) | 48.01041554 | 0 | (-256,-8) | 250.4495957 | 0 |
| 99 | 10 | 9 | (-4,0) | (-45,35) | 53.90732789 | 0 | (-152,-101) | 179.1786818 | 0.03125 |
| 100 | 10 | 10 | (5,9) | (-45,35) | 56.35601121 | 0.046875 | (-152,-101) | 191.7002869 | 0.03125 |

Fig. 17. Comparing PST and PQT in terms of time and distance for 1000 input points

Since we traverse from top to down in the polar split tree, we have at most logn levels. Thus the time complexity of this code is O(logn). If we add the preprocessing steps, the total complexity is O(nlogn).

## 7. Evaluation of the Proposed Method

Using Matlab programming language 2015b, simulation of this algorithm has been done in a machine with Windows 10 Operating System with CPU Intel core i5 2.5 GHZ, RAM:6GB. The input contains random coordinates between -1000 and 1000 in a 2-dimensional plane. In this paper, searching time and the distance to the nearest neighbor for polar quad tree and polar split tree are compared. The number of random query points change from 100 to 1000. Four types of input with the number of points of 100, 200, 500 and 1000 are tested. This test has been done for 10 types of random input and each random input has been repeated 10 times and for each random input, one random query point is chosen and the nearest neighbor to the query point is calculated for both trees. The results of the distance and time comparison for both algorithms are presented in the following tables of figures 14, 15, 16 and 17.

The graph of comparing two PST and PQT algorithms in terms of time and distance is in shown in figures 18, 19 and 20.

| | Polar Split Tree | | Polar Quad Tree | |
|---|---|---|---|---|
| DataSet Size | Time(ns) | Distance | Time(ns) | Distance |
| 100 | 0.013438 | 17.2179489 | 0.017031 | 28.5644889 |
| 200 | 0.012969 | 23.7880872 | 0.01625 | 47.3654892 |
| 500 | 0.013125 | 38.9709372 | 0.014844 | 62.4753168 |
| 1000 | 0.013438 | 54.5585853 | 0.013125 | 122.750356 |
| 10000 | 0.026563 | 312.666854 | 0.045313 | 762.142584 |

Fig. 18. Searching Time and Distance to the Nearest Neighbor for PST and PQT

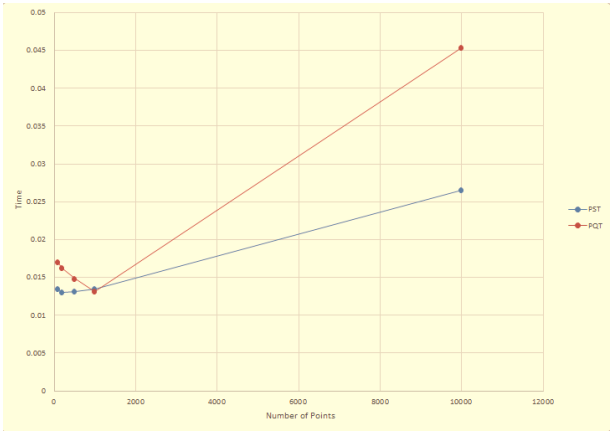


Fig. 19. Comparing PQT and PST in terms of searching Time NN1

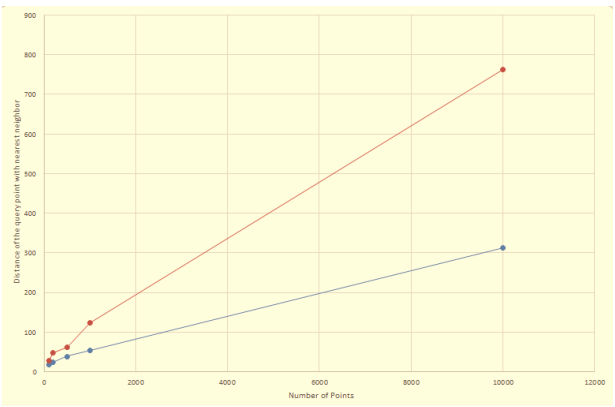


Fig. 20. Comparing PQT and PST in terms of Distance to NN1

## 8. Conclusion

In this paper, it is assumed that the input data have been distributed in a circle form, the smallest circle containing the input points is drawn and using the mentioned method, the polar split tree is constructed. As it was observed, the results of the nearest neighbor search in terms of time and distance in polar split tree have been better than polar quad tree. This matter indicates the optimality of the polar split tree in nearest neighbor search. This method has several applications including transmitting radio and telecommunication waves form host stations to receivers and searching the receivers. As in these areas we are dealing with circular shapes, it is better to use polar coordinate. B y entering one newly query data using NN1, polar split tree for the set of input points enables us to detect which cell of the tree belongs to this point.

**Acknowledgment**

We acknowledge the contribution of Marzieh Nazari for the sincere help that went through the initial writing of the paper. We also appreciate the referees for many useful comments and suggestions.

## References

[1] Robert Adams and Essex Christopher, Calculus: a complete course (Eighth ed.). Pearson Canada Inc., 2013.#
[2] Pablo Alonso Gonzalez, Optimization of antenna coverage in telecommunication systems, AGH University of Science and Technology, 2018.#
[3] Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars, Computational Geometry: Algorithms and Applications. Springer, pp. 309-312, 2008.#
[4] Cristina Costa, Francesco G.B. De Natale and Fabrizio Granelli, Quality Evaluation and Nonuniform Compression of Geometrically Distorted Images Using the Quadtree Distortion Map, EURASIP Journal on Applied Signal Processing, pp. 1899–1911, 2004.#
[5] Ameneh Eskandari, Zahra Nilforoushan and javad ranjbar, A Novel Method to Improve Query in Big Databases Using a Geometric tree Base Algorithm: International Journal of computer & Information Technologies, 5(1): 53-58, 2017.#
[6] Clara Grima and Alberto Marquez, Computational Geometry on Surfaces: Performing Computational Geometry on the Cylinder, the Sphere, the Torus, and the Cone, Kluwer Academic Publishers, 2001.#
[7] Anton Howard, Irl Bivens and Stephen Davis, Calculus (Seventh ed.). Anton Textbooks, Inc., 2002.#
[8] Moritz von Looz, Christian L. Staudt, Henning Meyerhenke and Roman Prutkin, Fast generation of dynamic complex networks with underlying hyperbolic geometry, Karlsruhe Reports in Informatics, 2014.#
[9] Nimrod Megiddo, Linear-time algorithms for Linear programing in R and related problems .: SIAM Journal on Computing, 12(4): 759-776,1983.#
[10] Giri Narasimhan and Michiel Smid, Geometric Spanner Networks.: Cambridge University Press, 2007.#
[11] Joseph O'Rourke, Computational Geometry in C, Cambridge University Press,1998.#
[12] Rina Panigrahy, An Improved Algorithm Finding Nearest Neighbor Using kd-trees, Latin American Symposium on Theoretical Informatics, pp. 387--398, 2008.#
[13] Sudhir Porwal, Quad tree-based level-of-details representation of digital Globe, Defence Science Journal, Vol. 63, No. 1, pp. 89-92, 2013.#
[14] Franco P. Preparata and Michael Ian Shamos, Computational Geometry - An Introduction, Springer, 1985.#
[15] Javad Ranjbar, Zahra Nilforoushan and Ameneh Eskandari, Improved Fingerprint Matching Speed in Large Databases Using Split Tree, 3rd National Conference on Distributed Computing and Big Data Processing, 2017.#
[16] J. R. Sack and J. Urrutia, Handbook of Computational Geometry, Elsevier, 1999.#
[17] Bahram Sadegh Bigham, Ali Mohades and Lidia Ortega, Dynamic Polar Diagram, Information Processing Letters, 109.2, pp. 142-146, 2008.#
[18] Hanan Samet, Foundations of Multidimensional and Metric Data Structures, Elsevier, 2006.#
[19] S. Saric , Z. Bozanic and R. Svalina: Automation in Developing Technical Documentation of Telecommunication Networks, Promet- Traffic- Traffico, Vol. 16, No. 5, pp. 257-262, 2004.#
[20] Ian Stewart and David Tall, Complex Analysis (the Hitchhiker's Guide to the Plane). Cambridge University Press, 1983.#
[21] A. M. Sukhov and D. Yu. Chemodanov, The Neighborhoods Method and Virtual Polar coordinates in Wireless Sensor Networks, Network and Communication Technologies, Vol. 2, No. 1, pp. 19-27, 2013.#
[22] Waldo Tobler and Zitan Chen, A Quad tree for Global Information Storage, Geographical Analysis, Volume18, Issue4, 1986.#
[23] https://www.coursera.org/learn/ml-clustering-and-retrieval/lecture/6eTzw/nn-search-with-kd-rees #

**Farzad Bayat** received his B.Sc. degree in Software Engineering from Buali Sina University of Hamedan,Hamedan,Iran in 2016. He received the M.Sc. degree in Knowledge Engineering from Kharazmi University, Tehran,Iran, in 2019. His area research interests include Software Development like Android App, Web Site and etc. He studied and worked in CRM (Customer Relationship Management) from Microsoft for two years.

**Zahra Nilforoushan** received her M.Sc. degree in Pure Mathematics (Algebraic Geometry) and Ph.D in Computer Science (Computational Geometry) from Amir kabir University of Technology, Tehran, Iran in 2003 and 2009 respectively. She served as a lecturer at Dept. of Computer Science, Faculty of Mathematical Sciences and Computer, Kharazmi University, Tehran, Iran from 2009 to 2011. Since 2012 she is with Dept. of Electrical and Computer Engineering, Faculty of Engineering at Kharazmi University as an Assistant Professor. Her research interests are Computational Geometry, Computer Graphics, Computer Vision, Analysis Algorithm and Robotics.