```
function results_trace_tree, pts, tree, points, stpt=stpt

treesize = size(*pts) & dim = treesize[2]
npts = (size(points))[1]

if (stpt EQ !null) then stpt = (where(*tree.level EQ 0))[0]
if ((n_elements(stpt) NE 1) and (n_elements(stpt) NE npts)) then stop

branches = replicate(-1L,npts,max(*tree.level)+2)
branches[*,0] = stpt

ct = 0
q = where(branches[*,0] NE -1L, nq)
while (nq NE 0) do begin
  ;; Current node
  a = branches[q,ct]

  ;; Level of current node and dimension to look at
  lev = (*tree.level)[a]
  dd = lev mod dim

  pp = points[q,dd]  ;; Points still to do
  tpp = (*pts)[a,dd] ;; Node values to compare with
  www = (pp LT tpp)

  ;; Determine whether to take hi or low branch
  newa = www*(*tree.lowchild)[a] + (1-www)*(*tree.hichild)[a]

  ;; Increment count
  ct++

  ;; add in the next node
  branches[q,ct] = newa
  q = where(branches[*,ct] NE -1L, nq)
endwhile

return, branches

end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

pro results_find_closest, pts, tree, points, stpt=stpt, rmin=rmin, pmin=pmin, $
  lll=lll

if (lll EQ !null) then lll = 0

treesize = size(*pts) & dim = treesize[2]
npts = (size(points))[1]
;;print, 'Tree Level = ', lll, npts

```

```
 52  if (stpt EQ !null) then begin
 53    stpt = (where(*tree.level EQ 0))[0]
 54  ;;  rmin = replicate(1d30, npts)
 55  ;;  pmin = lonarr(npts)
 56  endif
 57
 58  if ((n_elements(stpt) NE 1) and (n_elements(stpt) NE npts)) then stop
 59  if (n_elements(rmin) NE npts) then begin
 60    rmin = replicate(1d30, npts)
 61    pmin = lonarr(npts)
 62  endif
 63
 64  ;; First follow the tree to see where each point belongs
 65  branch = results_trace_tree(pts, tree, points, stpt=stpt)
 66
 67  bb = max(branch, dim=1)
 68  w = (where(bb EQ -1, nw))[0]
 69  if (nw NE 0) then branch = branch[*,0:w-1]
 70
 71  temp = (size(branch))
 72  brmax = (temp[0] EQ 1) ? 0 : temp[2]-1
 73
 74  ;; if (dd LT rmin) then it is possible that there is a closer point in that branch
 75  for i=brmax,0,-1 do begin
 76    nodes = branch[*,i]
 77  ;;  printf, 1, lll, i, nodes
 78    q = where(nodes NE -1, nq)
 79    if (nq NE 0) then begin
 80      pp = points[q,*]
 81      nn = nodes[q]
 82      ll = (*tree.level)[nn] mod dim
 83      dd = ((*pts)[nn,ll]-points[q,ll])^2
 84
 85      w = q[where(dd LT rmin[q], nw)]
 86      if (nw GT 0) then begin
 87        pp2 = points[w,*]
 88        nn2 = nodes[w]
 89        ll2 = (*tree.level)[nn2] mod dim
 90        node_pts = (*pts)[nn2,*]
 91        if (n_elements(pp2) NE n_elements(node_pts)) then stop
 92
 93        r0 = total((node_pts-pp2)^2,2)
 94        if (n_elements(r0) NE nw) then stop
 95        qr = where(r0 LT rmin[w], nr)
 96        if (nr NE 0) then begin
 97          rmin[w[qr]] = r0[qr]
 98          pmin[w[qr]] = nn2[qr]
 99        endif
100
101        ;; follow the branch not previously used
102        dir = (pp2[lindgen(nw),ll2] LT node_pts[lindgen(nw),ll2])
```

```
103         newst = dir*(*tree.hichild)[nn2] + (1-dir)*(*tree.lowchild)[nn2]
104         e = where(newst NE -1L, nee)
105         if (nee GT 0) then begin
106           pp3 = pp2[e,*]
107           newst = newst[e]
108           rtemp = rmin[w[e]]
109           ptemp = pmin[w[e]]
110           results_find_closest, pts, tree, pp3, stpt=newst, rmin=rtemp, pmin=ptemp, $
111             lll=lll+1
112           rmin[w[e]] = rtemp
113           pmin[w[e]] = ptemp
114         endif
115       endif
116     endif
117   endfor
118
119   end
120
121   ;;;;;;;;;;;;;;;;;;;;;;;;;;
122
123   pro results_kd_node, tree, pts, index, parent, ndim, level
124
125   n = n_elements(index)
126   case (n) of
127     1: begin
128        (*tree.level)[index] = level
129        (*tree.parent)[index] = parent
130        (*tree.lowchild)[index] = -1
131        (*tree.hichild)[index] = -1
132        indnode = index
133        end
134     2: begin
135        (*tree.level)[index[0]] = level
136        (*tree.parent)[index[0]] = parent
137        (*tree.lowchild)[index[0]] = index[1]
138        (*tree.hichild)[index[0]] = index[1]
139
140        (*tree.level)[index[1]] = level+1
141        (*tree.parent)[index[1]] = index[0]
142        (*tree.lowchild)[index[1]] = -1
143        (*tree.hichild)[index[1]] = -1
144        end
145     else: begin
146        dim = level mod ndim
147
148        p = (*pts)[dim,index]
149        s = sort(p)
150
151        ;; this node is at index[n2] = where(pts[*,dim] EQ median(pts[*,dim]))
152        n2 = n/2 & if ((n2 EQ 0) or (n2 EQ n-1)) then stop
153        indnode = index[s[n2]]
```

```
154       indlow = index[s[0:n2-1]] & indhi = index[s[n2+1:*]]
155
156       (*tree.level)[indnode] = level
157       (*tree.parent)[indnode] = parent
158       results_kd_node, tree, pts, indlow, indnode, ndim, level+1
159       results_kd_node, tree, pts, indhi, indnode, ndim, level+1
160
161       ii = indlow[(where((*tree.level)[indlow] EQ level+1, nq))[0]]
162       if (nq NE 1) then stop
163       if ((*tree.parent)[ii] NE indnode) then stop
164       (*tree.lowchild)[indnode] = ii
165
166       ii = indhi[(where((*tree.level)[indhi] EQ level+1, nq))[0]]
167       if (nq NE 1) then stop
168       if ((*tree.parent)[ii] NE indnode) then stop
169       (*tree.hichild)[indnode] = ii
170     endelse
171  endcase
172
173  end
174
175  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
176  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
177
178  function results_kd_tree, output
179
180  tstart = systime(1)
181
182  ;;pts = n x 3 array
183  pts = ptr_new(transpose([[*output.x], [*output.y], [*output.z]]))
184
185  sz = size(*pts)
186  ndim = sz[1]
187
188  tree = {level:ptr_new(lonarr(sz[2])), parent:ptr_new(lonarr(sz[2])), $
189    lowchild:ptr_new(lonarr(sz[2])), hichild:ptr_new(lonarr(sz[2]))}
190
191  index = lindgen(sz[2])
192  results_kd_node, tree, pts, index, -1, ndim, 0
193  pts = 0.
194
195  tend = systime(1)
196  print, 'results_kd_tree: ', tend-tstart
197
198  return, tree
199
200  end
```