HMS/SOS Tracking Code Enhancement

T. Navasardyan , P. Bosted , M Jones

Abstract

Examination of data taken in Hall C with high rates in the HMS and/or SOS spectrometers revealed that sometimes the code that picks the “best” among multiple track candidates pick a track that is not what a user looking at the one event display would have picked. To eliminate this problem a new code was written that uses different from old one track selection criteria.

1. Introduction.

The present article was done in the context of the experiment Е00-108 [1] in TJNAF. Next chapter will briefly describe apparatuses used for current work. Variables and determination of the trajectory of flying particle is discussed in chapter 3. Chapter 4 describes the problem, and in final chapter provides a solution and results that have been reached.

2 Apparatus

Current experiment was done in TJNAF Hall C using HMS [2] and SOS [3] spectrometers (Fig. 1).

![Figure 1: Schematic top view of Hall C spectrometer setup showing the location of the HMS and SOS relative to the target and incident beam.](image_url)

Detector packages are located in the detector hut of each spectrometer. Since both spectrometers have identical construction we will describe only one of them, for example HMS. Each spectrometer contains two drift chambers, two sets of x-y hodoscopes, a gas Cherenkov detector and a lead glass shower counter (Fig 2). In addition, for this experiment the HMS has aerogel Cherenkov detector [4].
In order to determine the trajectory of a detected particle at the target, one must determine its trajectory inside the detector hut using the information from the drift chambers. The first detectors encountered by a scattered particle in HMS are a pair of identical drift chambers. Each of the HMS drift chambers contains six planes of wires (Fig 3). The active area of each chamber is approximately 107x52 cm$^2$. An argon-ethane gas mixture is spreaded through the chambers at a rate between 400 and 800 cc/min. The sense wires are 25 $\mu$m gold-plated tungsten separated by 1cm within a given plane. The field wires are 150 $\mu$m gold-plated Cu-Be and are held at negative high voltages ~1800-2500V such that they generate equipotential surfaces that are approximately circular around each sense wire.

The tracking software also uses the position information gathered from scintillator hodoscopes of the HMS and the SOS to reconstruct the trajectory of a particle through
the hut. The HMS is equipped with four planes of scintillator hodoscopes. Each plane is composed of multiple scintillator elements called bars or paddles. Two of the planes have paddles arranged in spectrometer’s dispersive $x$ direction, and the other two in transverse $y$ horizontal direction. The planes are grouped into two $x$-$y$ pairs (S1X,S1Y and S2X,S2Y) and are situated as shown in Fig. 4.

The SOS drift chambers and scintillator hodoscopes are built on the same concept.

![Diagram of the HMS scintillator planes.](image)

Figure 4: View of the HMS scintillator planes.

The HMS has a lead-glass calorimeter, which was designed to measure deposited energy from the particle. The main goal of the lead-glass calorimeter is to separate electrons and hadrons. It is also used in tracking reconstruction and determining best track. The calorimeter consists of 52 blocks 10x10x70 cm$^3$ TF1 type lead glass. The blocks are arranged in four layers with 13 blocks per layer, giving a total thickness of the calorimeter along the direction of the particle motion of 16 radiation lengths. First two layers have PMT from each side, layers 3 and 4 have only one PMT read out. Calorimeter is rotated by 5$^0$ in dispersive plane to prevent particles from passing through occasional cracks in between blocks. The calorimeter designed to measure deposited energy from the particle.
3. Tracking

The trajectory of a particle in the detection system is measured by two drift chambers, each with six planes. Using matrix elements to describe the transport through the spectrometer, the vertex at the target is then reconstructed from the focal plane coordinates (position and direction of the trajectory at the target). The position of the particle as it passes through a focal plane is measured by using the position of the wire that is hit and adding to it the drift distance of the electron cloud from the ionized atoms. If wires from a drift chamber have been hit more than 30 times for one trigger then no tracks will be determined. The drift distance is determined from the drift time, which is recorded as drift chamber TDC values. Small corrections are applied for the time required for the signal to propagate along the wire and the differences in cable lengths between the chamber and the TDCs.

In each chamber there are only two planes Y and Y' that can give information about the transverse coordinate (information about the dispersive coordinate is given by the X, X', V and U planes). That is why in order to have information about the transverse coordinate of the track it is necessary to have a fired Y (Y') plane in each chamber. So, at least 5 out of 6 planes should be fired in each drift chamber. The drift chamber hits (fired planes) are used to identify clusters of hits (space points) in each chamber. A space point is generated with information from hits in a pair of unlike planes (combo)\(^1\). For the HMS, the X-Y, X-Y', X'-Y, X'-Y', U-Y, U-Y', V-Y, V-Y', U-V planes are considered unlike, but the wires of the X-U, X'-U, X-V, X'-V planes are too close to being parallel to accurately determine a space point. The tracking program takes the first combo and calls it a space point. Then if the second combo is within a certain range (space point criterion) of this space point then these two combos are united to one space point (X_{av}, Y_{av} coordinates). If the second combo is out of the range of the first space point then the program creates

---

\(^1\) Where unlike means that they can determine the position in both the dispersive and the transverse direction.
new space point. The same procedure is repeated for all combos in each drift chamber. Ideally, if all combos are within the range, then the program will create only one space point per drift chamber. The space point criterion is the radius of a circle within which the combos must lie in order to be considered part of the same space point.

All planes of a chamber contributing to a space point are used to determine a track through that chamber, called a stub. For each wire in a space point, the particle could pass the wire on the left or the right. To solve this left-right ambiguity, stubs are fitted using the wire information ($2^6$ stubs at maximum), and the stub with the best $\chi^2$ is chosen. The $\chi^2$ for the stub (track) is calculated with the formula,

$$\chi^2 = \frac{DC_{wc} - DC_{tc}}{\sigma^2_{DC}}$$

where $DC_{wc}$ and $DC_{tc}$ are wire and track coordinates of the stub (track) and $\sigma_{DC}$ is the wire chamber resolution for each plane. The stubs from the two drift chambers that have positions and slopes such that they point to each other within certain criteria, are linked together to form a track. In case multiple tracks are possible each of these tracks is recorded along with the $\chi^2$ of the fit, and the track with the lowest $\chi^2$ is selected as the final track. The stub criteria define the horizontal ($Y, Y'$) and vertical ($X, X'$) distance and slope ranges in which two stubs must lie to be considered part of the same track².

4. Tracking enhancement.

Examination of data taken in Hall C with high rates in the HMS and/or SOS spectrometers revealed that sometimes the code that picks the best among multiple track candidates pick a track that is not what a user looking at the one event display would have picked (Fig. 6).

---

² Since in the Hall C reconstruction software it has been assumed that $Y' = 0$ when fitting a stub, the $Y'$ criterion is not effective.
Two main reasons for this are that: the old code did not consider if a hit in S2X or S2Y was in time with the track (so sometimes a random hit in an S2 paddle would select a track with far-fetched values of reconstructed physics quantities, or a bad value of beta); the code had a cut on shower energy that was usually set to zero, because if no tracks passed the shower energy cut, the code would not look at S2 as a criteria for picking the best track, but rather skip directly to picking the track with the best chi-squared. To remedy these deficiencies, a new code was developed with the following algorithm:

- Make an outer loop over a set of track pruning criteria;
- For each criteria, loop over all tracks that have not been eliminated and if one or more of these tracks passes the given criteria, then eliminate the tracks that don’t pass criteria.

The new code is called x_select_best_track_prune.f (where x is h for HMS and s for SOS), and it is called from x_select_best_track.f if the new variable xsel_using_prune is set to 1 in the file xtracking.param. In this case the new variables\(^3\) are used in the selection criteria. The \(x_p, y_p, y_{tar}, \delta\) cuts are maximum values for the absolute value of the corresponding reconstructed target quantities. The value of \(\beta\) is for the maximum deviation from the nominal \(\beta\) for the given momentum and particle type. The variable \(she\) is for the minimum shower energy in GeV (note: not E/P). The meaning of these and the other criteria is briefly explained in xtracking.param, where the user should input the desired values. If the variables are not defined, the FORTRAN code uses some reasonable defaults. There are two additional criteria that not controlled by input variables: these are that there should be at least one S2X paddle lined up in space and time with the track (i.e. used in the time of flight code), and a similar criteria on S2Y. The order we choose is on reconstructed \(x_p, y_p, y_{tar}, \delta, \beta, df, chibeta, npmt, fptime\) time relative to the nominal time \(fp\_time\), a hit in SY2, and a hit in SX2. If still more than one track remains after all the above pruning, then the one with the best track chi-squared is picked. As an example fig. 7 shows the effect of the pruning compared to the using_scin option for the high rate run 52481.

---

\(^3\) The definition of these and other variables can be found in the Appendix A.
Figure 7: Coincidence time spectrum (in nsec) for high rate run with old track selection (using_scin in HMS and using chi-squared in SOS, blue) and with new track pruning code (red).

About 2-3% more counts are found in the coincidence peak, and the $\delta$ spectrum ($\Delta p/p$ distribution within momentum acceptance) shows that there is a tendency for these recovered events to be at negative $\delta$ in the HMS (Fig 8).

Figure 8: Coincidence time spectrum for the HMS $\delta$ spectrum.

Most of recovered events previously had values of $\delta$ that were outside the nominal acceptance range. The code was also tested by looking at the double arm missing mass peak for all of the 47 degree runs of the Baryons experiment [5] (Fig. 9).
Figure 9: Double arm missing mass squared for the bulk of the Baryon experiment runs. The solid (dashed) curve is without (with) the new pruning code.

**Conclusion**

With the new pruning code, one gains 3.1% useful tracks for the analysis. The tracks with far-fetched parameters are eliminated. Suggested approach is included in Hall C Analysis Code and it is multipurpose and can be used in wide range of experiments.

**Appendix A**

**Definition of variables.**

\[ x_{tar} \ (y_{tar}) - \text{vertical (horizontal) position at the target plane} \]

\[ x_{ptar} \ (y_{ptar}) - \text{tangents of the in-plane (out-of-plane) scattering angles.} \]

\[ \delta = \frac{P_{\text{recon}} - P_0}{P_0} - \text{particle momentum relative to the central momentum of the spectrometer (} P_0 \text{)} \]

\[ x_{fp} \ (y_{fp}) - \text{focal plane vertical (horizontal) coordinates.} \]

\[ \beta = \frac{v}{c} - \text{velocity of the particle.} \]
Appendix B
Prune Subroutine for Selecting Best Track in HMS.

SUBROUTINE H_SELECT_BEST_TRACK_PRUNE(ABORT,err)
*--------------------------------------------------------
*-
* Purpose and Methods : Select the best track through the HMS
*-
* Required Input BANKS
*-
* Output BANKS
*-
* Output: ABORT           - success or failure
* : err             - reason for failure, if any
*-
* $Log: h_select_best_track_prune.f,v $
* Revision 1.1  2005/03/23 16:33:32  jones
* Add new code s_select_best_track_prune.f (P Bosted)
*-
* Revision 1.1  2005/03/08 bosted
* Initial revision
*-
*--------------------------------------------------------
IMPLICIT NONE
SAVE
*
character*50 here
parameter (here= 'H_SELECT_BEST_TRACK_PRUNE')
*
logical ABORT
character(*) err
*
INCLUDE 'hms_data_structures.cmn'
INCLUDE 'gen_routines.dec'
INCLUDE 'gen_constants.par'
INCLUDE 'gen_units.par'
INCLUDE 'hms_physics_sing.cmn'
INCLUDE 'hms_calorimeter.cmn'
INCLUDE 'hms_scin_parms.cmn'
INCLUDE 'hms_scin_tof.cmn'
INCLUDE 'hms_tracking.cmn'

integer*4 goodtrack,track,ngood,reject(1000),trk
logical first,keep(1000)
real*4 chi2perdeg,chi2min,betap,p
c
integer*4 i,j
data first /.true./

*--------------------------------------------------------
ABORT= .FALSE.
err= ' '
* Need to test to chose the best track
HSNUM_FPTRACK = 0
HSNUM_TARTRACK = 0

if (first) then
  write(*,*) ' HMS track selection using pruning method'
  first = .false.
! Make sure limits are reasonable
  hprune_xp    = max(0.08, hprune_xp)
  hprune_yp    = max(0.04, hprune_yp)
  hprune_ytar  = max(4.0,  hprune_ytar)
  hprune_delta = max(13.0, hprune_delta)
  hprune_beta  = max(0.1,  hprune_beta)
  hprune_df    = max(1,  hprune_df)
  hprune_chibeta= max(2.,  hprune_chibeta)
  hprune_fptime= max(5.,  hprune_fptime)
  hprune_npmt  = max(6.,  hprune_npmt)
  hprune_she   = max(0.1,  hprune_she)
  write(*,'(1x,''using following HMS limits''/
    1x,''abs(xptar)<'',f6.3/
    1x,''abs(yptar)<'',f6.3/
    1x,''abs(ytar)<'',f6.3/
    1x,''abs(delta)<'',f6.3/
    1x,''shower energy >'',f6.3, '' GeV''/
    1x,''abs(beta-betap)<'',f6.3/
    1x,''ndegfreedom trk>='',i2/
    1x,''beta chisq>'',f6.1/
    1x,''num PMT hits >=",i3/
    1x,"(ftime-hstart_time_center)<=",f6.1')
  hprune_xp,hprune_yp,hprune_ytar,hprune_delta,hprune_she,
  hprune_beta,hprune_df,hprune_chibeta,hprune_npmt,hprune_fptime
endif
c
if( HNTRACKS_FP.GT. 0) then
  chi2min= 1e10
  goodtrack = 0

! Initialize all tracks to be good
  do track = 1, HNTRACKS_FP
keep(track) = .true.
reject(track)=0
enddo

! Prune on xptar
ngood=0
do track = 1, HNTRACKS_FP
  if( abs(hxp_tar(track)) .lt. hprune_xp .and. keep(track)) then
    ngood = ngood + 1
  endif
enddo
if(ngood.gt.0) then
  do track = 1, HNTRACKS_FP
    if( abs(hxp_tar(track)) .ge. hprune_xp) then
      keep(track) = .false.
      reject(track) = reject(track) + 1
    endif
  enddo
endif

! Prune on yptar
ngood=0
do track = 1, HNTRACKS_FP
  if( abs(hyp_tar(track)) .lt. hprune_yp .and. keep(track)) then
    ngood = ngood + 1
  endif
enddo
if(ngood.gt.0) then
  do track = 1, HNTRACKS_FP
    if( abs(hyp_tar(track)) .ge. hprune_yp) then
      keep(track) = .false.
      reject(track) = reject(track) + 2
    endif
  enddo
endif

! Prune on ytar
ngood=0
do track = 1, HNTRACKS_FP
  if( abs(hy_tar(track)) .lt. hprune_ytar .and. keep(track)) then
    ngood = ngood + 1
  endif
enddo
if(ngood.gt.0) then
  do track = 1, HNTRACKS_FP
    if( abs(hy_tar(track)) .ge. hprune_ytar) then
      reject(track) = reject(track) + 3
    endif
  enddo
endif
keep(track) = .false.
reject(track) = reject(track) + 10
endif
enddo
endif

! Prune on delta
ngood=0
do track = 1, HNTRACKS_FP
if( abs(hdelta_tar(track)) .lt. hprune_delta .and. keep(track)) then
ngood = ngood + 1
endif
enddo
if(ngood.gt.0) then
do track = 1, HNTRACKS_FP
if(abs(hdelta_tar(track)) .ge. hprune_delta) then
keep(track) = .false.
reject(track) = reject(track) + 20
endif
enddo
endif

! Prune on shower energy in GeV
ngood=0
do track = 1, HNTRACKS_FP
if(htrack_et(track) .ge. hprune_she .and. keep(track)) then
ngood = ngood + 1
endif
enddo
if(ngood.gt.0) then
do track = 1, HNTRACKS_FP
if(htrack_et(track) .lt. hprune_she) then
keep(track) = .false.
reject(track) = reject(track) + 40
endif
enddo
endif

! Prune on beta
ngood=0
do track = 1, HNTRACKS_FP
p = hp_tar(track)
betap = p/sqrt(p*p+hpartmass*hpartmass)
if( abs(hbeta(track)-betap) .lt. hprune_beta .and. keep(track)) then
endif
ngood = ngood + 1
endif
enddo
if(ngood.gt.0) then
  do track = 1, HNTRACKS_FP
    p = hp_tar(track)
    betap = p/sqrt(p*p+hpartmass*hpartmass)
    if(abs(hbeta(track)-betap) .ge. hprune_beta) then
      keep(track) = .false.
      reject(track) = reject(track) + 100
    endif
  enddo
endif

! Prune on deg. freedom for track chisq
ngood=0
do track = 1, HNTRACKS_FP
  if(HNFREE_FP(track) .ge. hprune_df .and. keep(track)) then
    ngood = ngood + 1
  endif
enddo
if(ngood.gt.0) then
  do track = 1, HNTRACKS_FP
    if(HNFREE_FP(track) .lt. hprune_df) then
      keep(track) = .false.
      reject(track) = reject(track) + 200
    endif
  enddo
endif

! Prune on num pmt hits
ngood=0
do track = 1, HNTRACKS_FP
  if(hnum_pmt_hit(track) .ge. hprune_npmt.and. keep(track)) then
    ngood = ngood + 1
  endif
enddo
if(ngood.gt.0) then
  do track = 1, HNTRACKS_FP
    if(hnum_pmt_hit(track) .lt. hprune_npmt) then
      keep(track) = .false.
      reject(track) = reject(track) + 100000
    endif
  enddo
endif
! Prune on beta chisqr
    ngood = 0
    do track = 1, HNTRACKS_FP
        if(hbeta_chisq(track) .lt. hprune_chibeta .and. hbeta_chisq(track) .gt. 0.01 .and. keep(track)) then
            ngood = ngood + 1
        endif
    enddo
    if(ngood.gt.0) then
        do track = 1, HNTRACKS_FP
            if(abs(htime_at_fp(track)-hstart_time_center).lt.hprune_fptime .and. keep(track)) then
                ngood = ngood + 1
            endif
        enddo
        if(ngood.gt.0) then
            do track = 1, HNTRACKS_FP
                if(hgood_plane_time(track,4).and. keep(track)) then
                    ngood = ngood + 1
                endif
            enddo
        endif
        if(ngood.gt.0) then
            do track = 1, HNTRACKS_FP
                if(hbeta_chisq(track) .ge. hprune_chibeta .or. hbeta_chisq(track) .le. 0.01) then
                    keep(track) = .false.
                    reject(track) = reject(track) + 1000
                endif
            enddo
        endif
    endif

! Prune on fptime
    ngood = 0
    do track = 1, HNTRACKS_FP
        if(abs(htime_at_fp(track)-hstart_time_center).lt.hprune_fptime .and. keep(track)) then
            ngood = ngood + 1
        endif
    enddo
    if(ngood.gt.0) then
        do track = 1, HNTRACKS_FP
            if(abs(htime_at_fp(track)-hstart_time_center).ge. hprune_fptime) then
                keep(track) = .false.
                reject(track) = reject(track) + 2000
            endif
        enddo
    endif

! Prune on Y2 being hit
    ngood = 0
    do track = 1, HNTRACKS_FP
        if(hgood_plane_time(track,4).and. keep(track)) then
            ngood = ngood + 1
        endif
    enddo
    if(ngood.gt.0) then
        do track = 1, HNTRACKS_FP
            if(.not.hgood_plane_time(track,4)) then
                keep(track) = .false.
                reject(track) = reject(track) + 10000
            endif
        enddo
    endif
endif
enddo
endif

! Prune on X2 being hit
ngood=0
do track = 1, HNTRACKS_FP
   if(hgood_plane_time(track,3).and. keep(track)) ngood = ngood + 1
endo
if(ngood.gt.0) then
   do track = 1, HNTRACKS_FP
      if(.not.hgood_plane_time(track,3)) then
         keep(track) = .false.
         reject(track) = reject(track) + 20000
      endif
   enddo
endif

! Pick track with best chisq if more than one track passed prune tests
goodtrack = 1
do track = 1, HNTRACKS_FP
   chi2perdeg = HCHI2_FP(track)/max(1.,FLOAT(HNFREE_FP(track)))
   if(chi2perdeg .lt. chi2min .and. keep(track)) then
      goodtrack = track
      chi2min = chi2perdeg
   endif
endo
HSNUM_TARTRACK = goodtrack
HSNUM_FPTRACK  = goodtrack
endif

! for debugging
if( HNTRACKS_FP.GT. 100) then
   write(*,'(/)')
do trk = 1, HNTRACKS_FP
      write(*,'(3i3,4L2,7f6.1,L2,i9)') trk,HNFREE_FP(trk),
         hnum_pmt_hit(trk),
         hgood_plane_time(trk,1), hgood_plane_time(trk,3),
         hgood_plane_time(trk,2), hgood_plane_time(trk,4),
         htime_at_fp(trk), hbeta(trk), hbeta_chisq(trk),
         hdelta_tar(trk), hy_tar(trk), hxp_tar(trk), hyp_tar(trk),
         keep(trk), reject(trk)
endo
write(*,'(1x,''good trk='',2i4)') goodtrack
endif
return
end
Bibliography.


