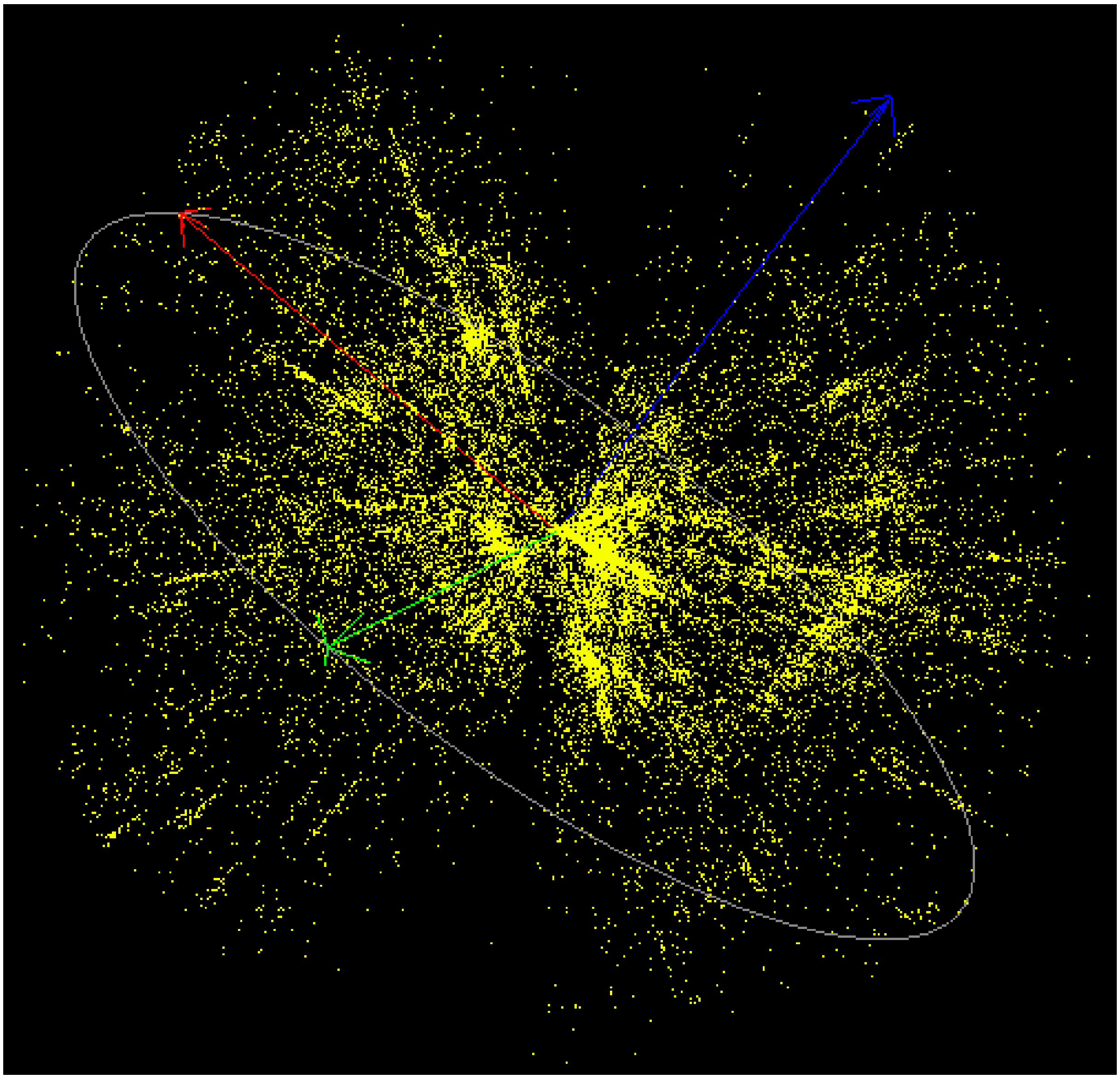
3D Redshift Space Visualization of 35699 Galaxies from the ADS CfA Catalogue

Lars Lindberg Christensen
Michael J.D. Linden
Bo Milvang-Jensen*

June 29th, 1994



*All from the Niels Bohr Institute for Astronomy, Physics og Geophysics, Astronomical Observatory, Copenhagen.

Abstract

This paper is the documentation for our video "3D Redshift Space Visualization of 35699 Galaxies from the ADS CfA Catalogue" made as the examination task for the course "3D Dynamics & Visualization" given by Åke Nordlund and Klaus Galsgaard in the spring term of 1994.

Our intension was to use the tools presented to us during the course (mainly SGI Explorer and IDL) to make a three-dimensional animation of a large data-set. We chose the CfA Redshift Catalogue retrieved from NASA Astrophysics Data System (ADS) in order to examine the evidence for large scale structure as indicated by (amongst others) Geller and Huchra (1989) in a way never presented before.

Please note that we have included printouts of the IDL procerdures used as well as snapshots of the Explorer-maps involved in the animation process. Since this is not requested for this report it is mainly for our own sake to make it easier to reproduce similar data at a later time.

Contents

1 Introduction

In the last few years deep redshift surveys of the Universe has supplied us with a possibility of mapping the distribution of galaxies with respect to thier spatial location and not only their projected position onto the celestial sphere. By mapping this distribution a large scale structure – if excisting – should be evident.

In different publications it has been inferred that evidence of this kind exists. Geller and Huchra (1989) present results of this kind for the northern hemisphere and recently da Costa *et al.* (1994) have presented similar results for the southern part of the sky.

A common characteristic for these publications is that they present their distributions mainly in the form of two dimensional plots of slices taken from the entire sample. These slices have been used to identify such famous large scale features as the Great Wall (for the northern survey) and a similar structure (the Southern Wall) in the southern hemisphere, but one could be tempted to ask how structures like these would look if one could see them from the outside, "fly" around them and in other ways examine them in three dimensional space - interactively.

The way towards this goal is of course related to the use of computers and their ability to handle large sets of data and furthermore display these sets in a real time animation. In this way it is possible to get a far better view of the structures and in general of the entire distribution.

The scientific implications of a computerbased visualization in this sense could be of considerable importance. Simulations of the evolution of a CDM[1] dominated universe give results that – to a large extent – look like the real universe (White, 1988), but maybe detailed 3D studies could impose further constraints on the models and thus on our understandig of the evolution of the universe as a whole.

1.1 A brief overview of the Catalogue

The CfA Redshift Catalogue is a compilation of a large number of sources containing coordinates and radial velocities for galaxies. It incorporates much of the latest velocity data from the Whipple Observatory and other sources as well as velocities from earlier compilations such as the Third Reference Catalogue by de Vaucouleurs, de Vaucouleurs and Corwin, the Index of Galaxy Spectra by Gisler and Friel and the Catalogue of Radial Velocities by Palumbo, Tanzella-Nitti and Vettolani (in this work no indication of source has been made).

The data presented in the catalogue have primarily been assembled for the purpose of studying the large scale structure of the universe and, as such, are nearly complete in redshift (velocity) information. The source data are obtained by means of both wide-angle and pencil-beam surveys and are thus very different in size and coverage.

2 Selection and input of data

2.1 The format of the database

The data were retrieved from the NASA ADS (Astrophysics Data System) which is a large compilation of different datafiles related to astronomy and astrophysics. After logging in to the NASA ADS one can get a listing of the catalogues available in the system. It is also possible to get a description of the catalogue which describes the catalogue in general and gives information about the specific format of the data (i.e. which columns contain what information). We used data from two catalogues: The CfA Redshift Catalogue and the CfA Redshift Catalogue with $V > 99999$. These catalogues contain slightly different information about the galaxies, with the former being far more detailed than the latter. This, however,

[1] **C**old **D**ark **M**atter

is of no significance, since the only information we needed was the coordinates of the objects and thier redshifts. In the CfA Redshift Catalogue the redshift is listed as a velocity and needed a small conversion to redshift.

2.2 Reading in the data

The data were read by means of the SQL (Structured Query Language) composition editor under the Queries menu in ADS. Here one enters an SQL text specifying which columns of the data in the catalogue one wants to retrieve and a path + filename for the output results. We asked for the columns containing right ascension and declination in decimal degrees, redshift (velocity for the CfA catalogue with V < 99999) and the source for the data specified by a number refering to a list of survey sources and thus ended up with a large text file containing coordinates (α, δ in J2000), redshift or velocity and source number for a total of 35699 galaxies.

2.3 Preprocessing the data

2.3.1 Splitting up the master-file

The first step in the preprocessing of the data was the rather tideous work of splitting up the "master-file" into one file for each of the sources. This was accomplished by a lot of manual work and some awk scripts (cf. appendices N and O).

The reason for this splitting of the master-file was to enable us – at a later time – to remove data from specific surveys i.e. pencil-beam surveys. Also the redshift data given as velocity were – by means of awk – recalculated to redshift.

2.3.2 Converting to Explorer-readable format

To be able to read in the data in SGI Explorer they had to be converted to a suitable format. This was done by using an IDL procedure called `ascii_to_bin.pro` that first converted the spherical coordinates (α, δ and z) to Cartesian coordiantes in a straightforward transformation and then produced a binary file containing the data (extension `.dat`) and a header file (extension `.idl`) readable by the Explorer module IDL2Lat. The `ascii_to_bin.pro` procedure is listed in appendix A.

There were prepared Explorer readable files for the entire set of data as well as for relevant subsamples (i.e. without pencil-beams, data with $z < 0.04$, 15° slices in right ascension etc.).

3 Preparing and doing the video

In this section we will shortly describe the methods and techniques used in preparing and actaually producing the video.

3.1 Setting up Explorer

One of two maintools for this production was – as already mentioned – the SGI Explorer program running on a Silicon Graphics Indigo^2 workstation with a 100 MHz RISC 4000 CPU, RISC 4010 FPU, 1Mb cache, 96 Mb RAM and an Extreme graphics board.

The program is a very intuitive piece of software where the data flows in pipes you connect to different visualization or calculation modules and ends up in a render module showing the data. In this render module it is possible (though dependent on the size and type of the data set) to manipulate the data (zoom, turn, fly through etc.) in real-time.

3.1.1 Building the relevant maps

The modules and pipes mentioned above are organized in so-called maps showing graphically what is connected to what and which modules are active at a given time.

In most cases (especially the simple ones!) it is rather straightforward to build these maps since one basically only needs

- A module to read in the data (IDL2Lat)
- One or more modules to give the data a "Physical shape" (Ball, IsosurfaceLat etc.)
- A Render module to show the data

Furthermore modules can be incorporated that bound the data with approriate lines (BoundBox or Line2Geo etc.) or one can apply modules that write images from the render module to a disk in a compatible graphics format (i.e. TIFF).

3.1.2 Live animations

Most of the video was produced making live animations in the render module while the videotape was running. Very simple maps were applied to do this including only the IDL2Lat module to read in the data, the Ball module to shape up the datapoints as points, a GenerateColormap module to set the color of the data points, to or three Line2Geo modules to create a system of reference (see section 4.1) and other bounding line effects and of course the render module to make it all visible. These sequenses needed some rehearsing of mouse-movement

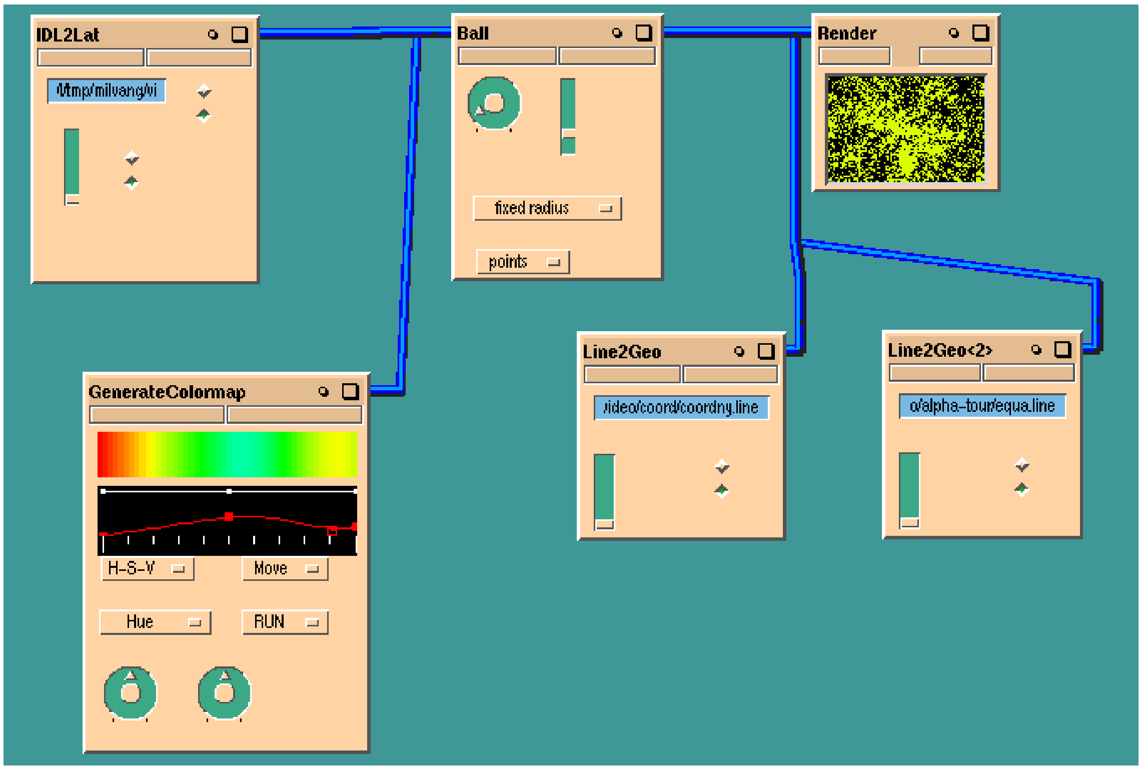


Figure 1: *One of the Explorer maps used.*

for the operator but in a larger perspective this was still less time-consuming than making "real" animations with IDL (see section 3.2). A snapshot of the map is shown in figure 1.

3.2 Animations with IDL

A more sophisticated technique was applied to produce the sixth section of the video – an animation of 15° slices revolving around the axis pointing towards the north celestial pole (see section 4.6 for further details).

The basic idea is to make a camera animation by setting out camera points in the render module by means of the AnimateCamera module which then is able to calculate and render a specified number of interpolated camerapoints between the points set by the user. In this manner an animation is produced.

By connecting the render output to the WriteImage module it is possible to write the render image to a disk in TIFF format. By synchronizing the input of a new image in the render module and the readout of the WriteImage module it is possible to write out the render image every time it has been redrawn. The module AnimFile gives the output images an appropriate path + name (number) and extension (`.tiff`).

The frames produced in this manner can now be read by IDL and stored into a socalled "pixmap". A pixmap is an invisible window or rather an invisible part of memory where the frames are lying side by side. Since the frames are stored this way they can be displayed very fast and smooth (approx. 20 frames pr. second!) and thus a very nice animation can be created. The obvious limit for this process is the available amount of free memory and since 200-300 frames are needed for doing only 10 sec. of animation this really sets a very low upper limit for the length of such animations.

We used the IDL procedures `loadanim.pro` for reading our frames into the pixmap and `runanim.pro` for running the animation. These procedures are listed in appendix H.

3.3 The computer to video interface

The actual videorecording was made with a SuperVHS Hitachi S890 NICAM Digital HiFi Stereo VCR and the output from the workstation was controlled by means of the VCP tool. This tool provides control of several parameters of the videooutput - horisontal and vertical offsets, antidithering and subcarrier frequency just to mention a few.

In practice a frame of fixed size is placed over the section of the screen one wants to record, the offsets and other relevant parameters are set and then the recording is controlled at the VCR "as usual".

3.4 On screen text, sound and narration

The on screen text separating the sections of the video was produced by means of IDL.

The sound and narration of the video were dubbed on the final video by means of an audio mixer plugged into the VCR audio input. A microphone for the narrator and a Compact Disc player were connected to the mixer and the audio dubbing was performed in real-time, without editing, while the video was running.

4 The video - section by section

In this section the six individual sections of the video will be shortly described and commented.

Furthermore this report is acompanied by six overhead viewgraphs showing one typical frame from each of the six sections. The frame is also shown under each section respectively.

4.1 Our system of reference

This first part of the video introduces the viewer to our "system of reference". It consists of a coordinate grid with the three axes marked by the colors red, green and blue where the

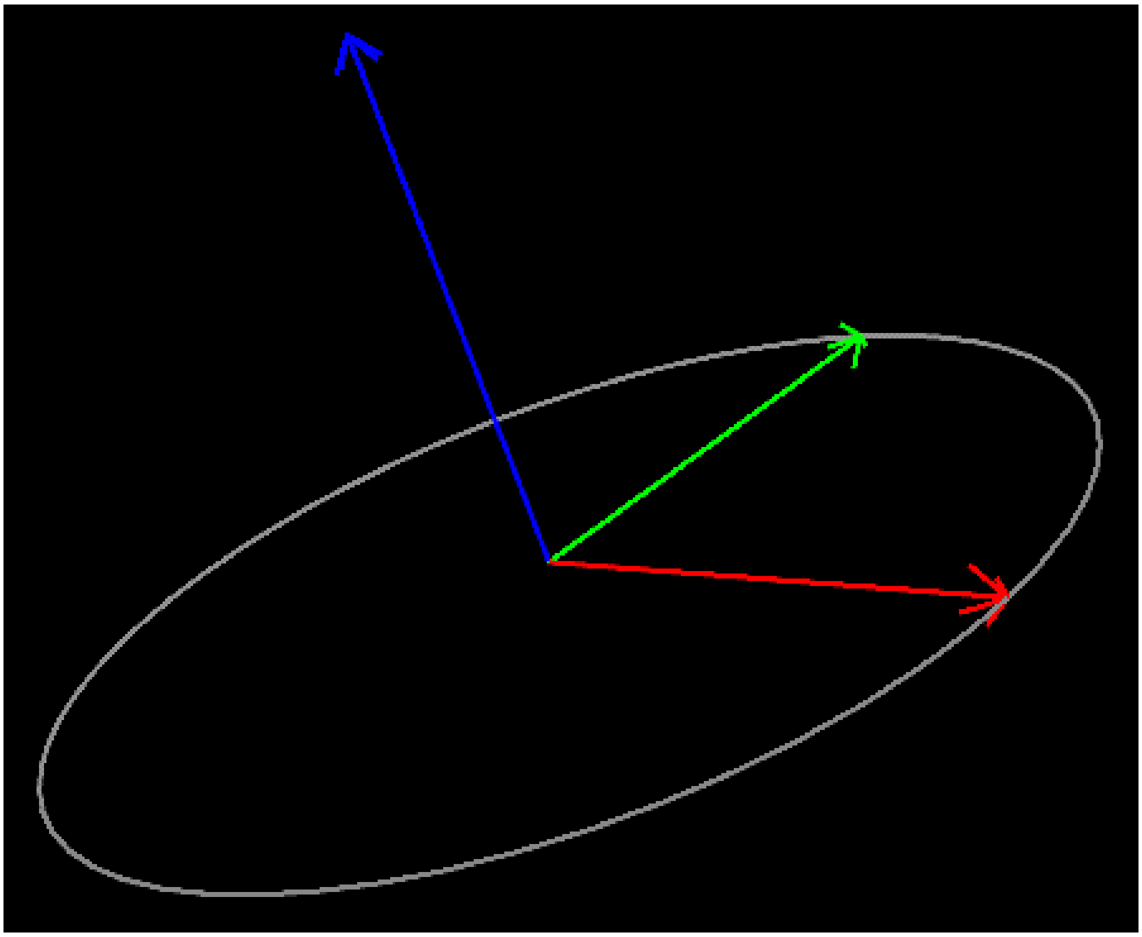


Figure 2: *Our system of reference*

blue axis points towards the northern celestial pole. A circle is also shown marking the sky's equator placed at a distance of $z = 0.04$. Also the length of the coordinate axis corresponds to this redshift.

The reason for choosing this distance for dividing the data into "inner" and "outer" is that most of the surveys claim to be more or less complete out to this limit. This is obviously not the case for the samples from $z > 0.04$ out to the most distant objects at $z \approx 3.8$.

4.2 The full set of data - An overview

To give the viewer and perception af "where we are and where we are going" the second section gives an overview of all the data - zooming in from a great distance, turning the dataset around and moving back and forth inside it.

Here it is most certainly visible that the outermost sectios by no means are complete. Pencil-beam surveys are streching from the innermost parts of the data almost to the outer limit at $z \approx 3.8$. This becomes even more obvious as the data at the end of this sequence is shown without the pencil-beam surveys. A far more homogeneous sample emerges.

In this second section the "zone of avoidance" is clearly visible. It is the dark band devoid of galaxies that is due to the obscuration by our own galaxy - the Milky Way.

4.3 The innermost section, $z < 0.04$

In this - the third - section the most interesting subsample of the data is shown. It is the innermost 23000 galaxies within a redshift limit of 0.04 and as it was indicated above these

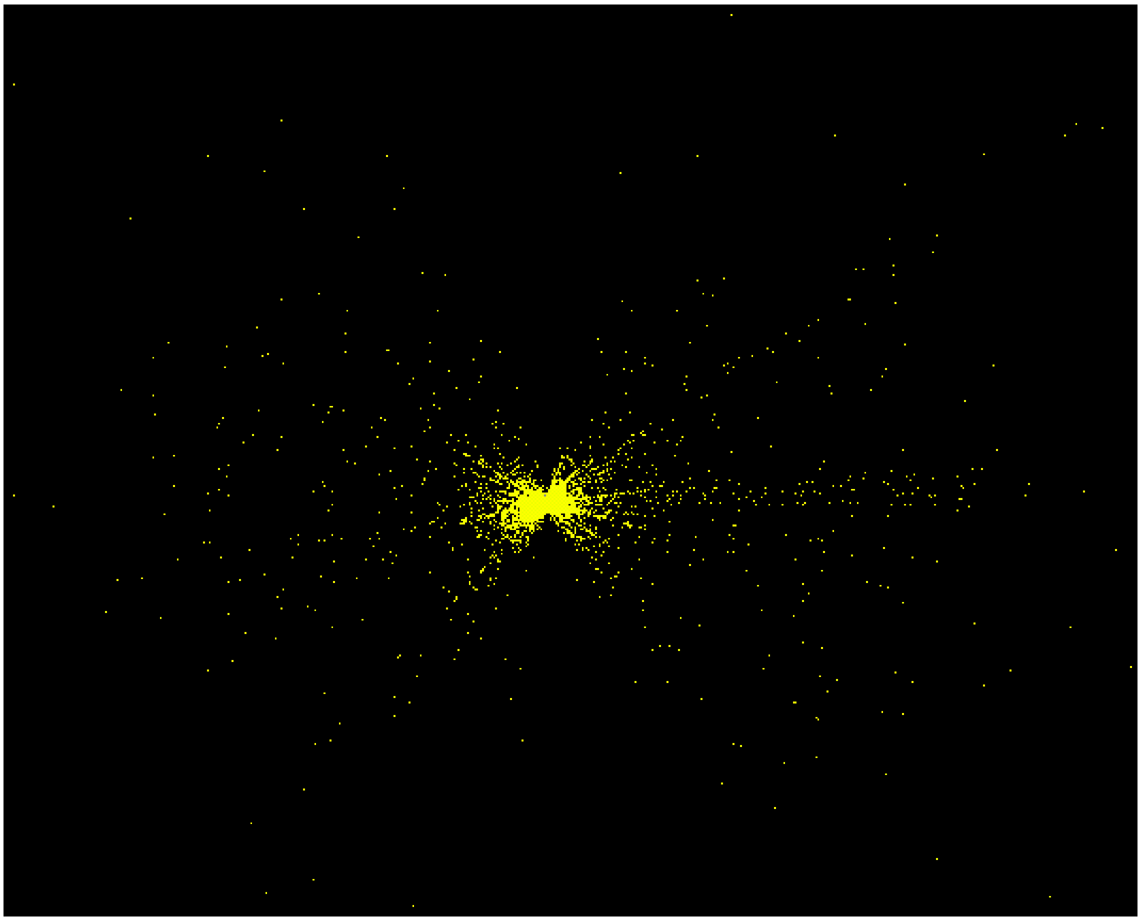


Figure 3: *The full set of data - An overview*

data are to a large extent complete except the "zone of avoidance" mentioned above and some unobserved areas in the southern part of the sky.

In these data a lot of structure is seen. The clusters of galaxies are seen as elongated structures due to their high velocity dispersion ($\sigma_r \approx 1000$ km/s) which is sometimes referred to as the "finger of God" effect.

4.4 A view from Origo

In this fourth section of the video we have placed the viewer in our normal observing point - on the Earth. In this view the projection of the structures onto the celestial sphere makes it if not impossible then at least very difficult to recover the distribution seen in the previous sections of the video. This demonstrates the need for 3D visualization.

4.5 The northern CfA slice, $8^h < \alpha < 17^h, 8.5° < \delta < 44.5°, z < 0.04$

In the fifth section of the video one of the well known CfA slices (Geller & Huchra, 1989) spanning from 8^h to 17^h in right ascension and from 8.5° to 44.5° in declination.

Here the famous "Homunculus" consisting mainly of the Coma cluster is seen and also "The Great Wall" and the Virgo cluster are evident.

An important observation however is that the impression of the structures is strongly dependent of the way you look upon the data. A slight turn can make an "obvious feature" disappear completely!

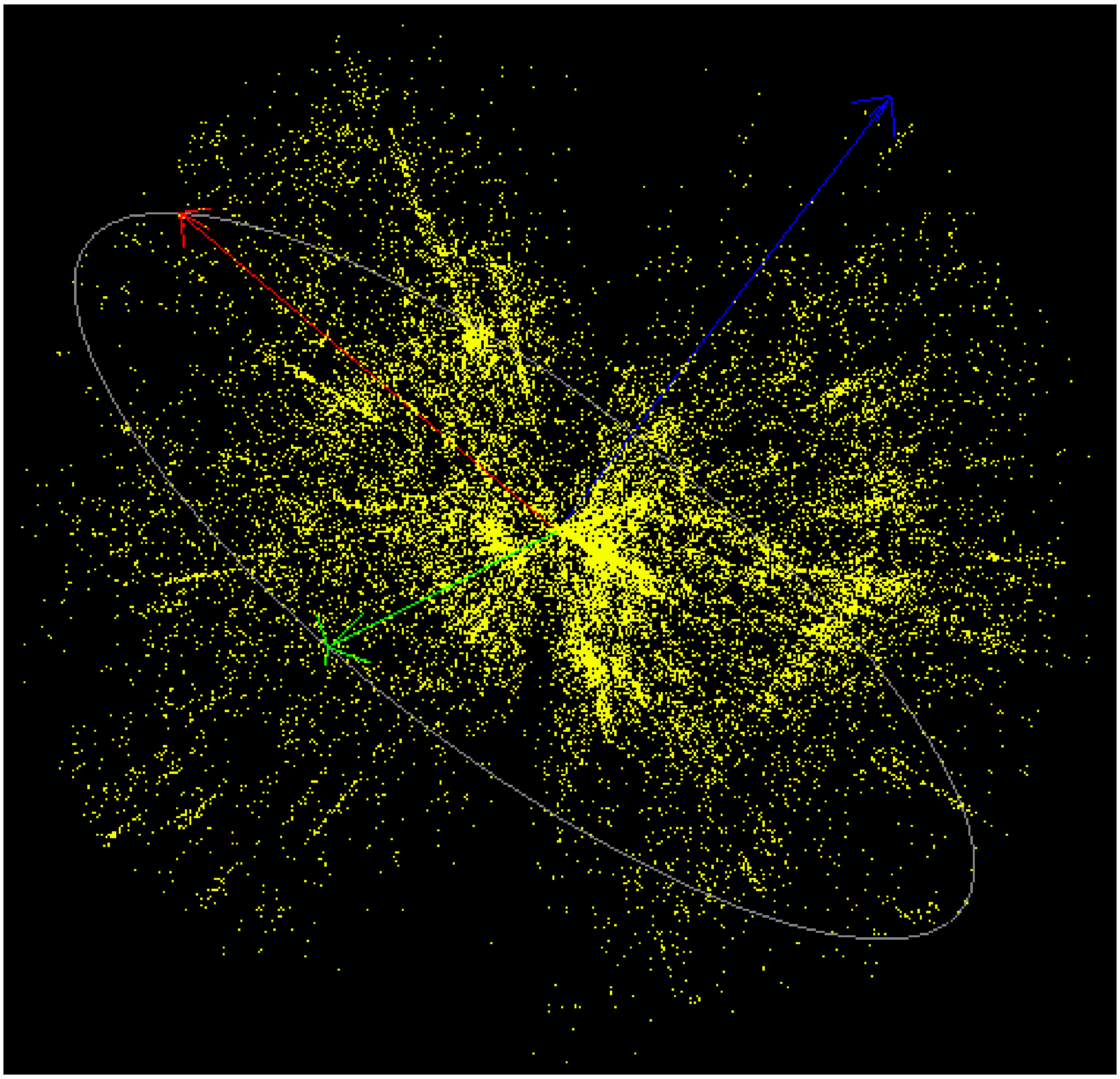


Figure 4: *The innermost section*, $z < 0.04$

4.6 $15°\alpha$-slices of the innermost data, $z < 0.04$

In the last section of the video an animation (as described in section 3.2) is used to show the spongy structure of the data.

The animation is made by creating subsamples of the complete dataset consisting of slices $15°$ in right ascension each with an α_{start} increased by $2°$ with respect to the previous.

In the animation the viewer is placed in a co-moving system moving a bit up and down while the slices are being shown one by one. Because of the speed of the animation one sees a smooth transition of the structures making especially the voids very evident as almost circular areas completely devoid of galaxies.

5 Further developments

In order to make further progress down the "3D visualization" road a number of interesting topics - which the scheme of this task unfortunately did not allow us to persue - could be investigated.

The visual impression of the structures is - as mentioned above - strongly dependent on the way one looks upon the data, but also on the completeness of the samples. These two effects can be hard if not impossible to distinguish and it is therefore of vital importance that

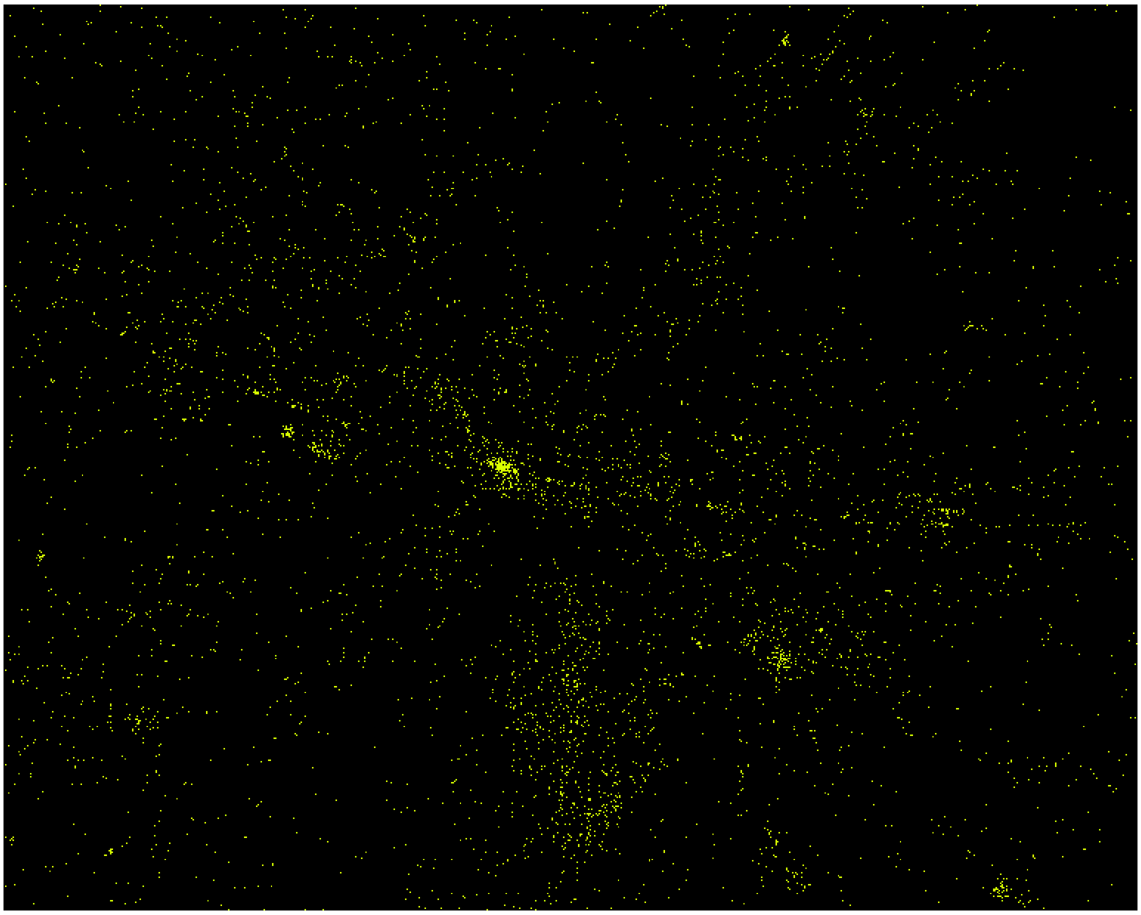


Figure 5: *A view from Origo*

the completeness of the samples used in a visualization is investigated and documented.

Before this is done any quantification of the distribution could easily give in the best case inaccurate and in the worst case misleading results.

Several ways of realizing such a quantification could be considered. One could perform galaxy counts within small subvolumes of the complete sample and this way map the density distribution of galaxies. This density could in turn be visualized by means of isosurfaces or strings tracing the density.

Analysis of this kind may be an hithero unknown source for finding constraints on cosmological models and are therefore an obvious goal for further developments.

6 Concluding remarks

During the preperation and production of our video we feel that we have seen a new and rather revolutionary way to investigate redshift survey data mapping the large-scale structure of our universe. We hope that this feeling also emerges when looking at the video.

We strongly believe that 3D visualization may prove to be a powerful tool when examining these structures in order to put further constraints on cosmological models as mentioned in the section 'Further developments'.

Acknowledgements

We would like to thank our supervisors, Åke Nordlund and Klaus Galsgaard, for introducing us to 3D visualization and for their help with hard- and software problems.

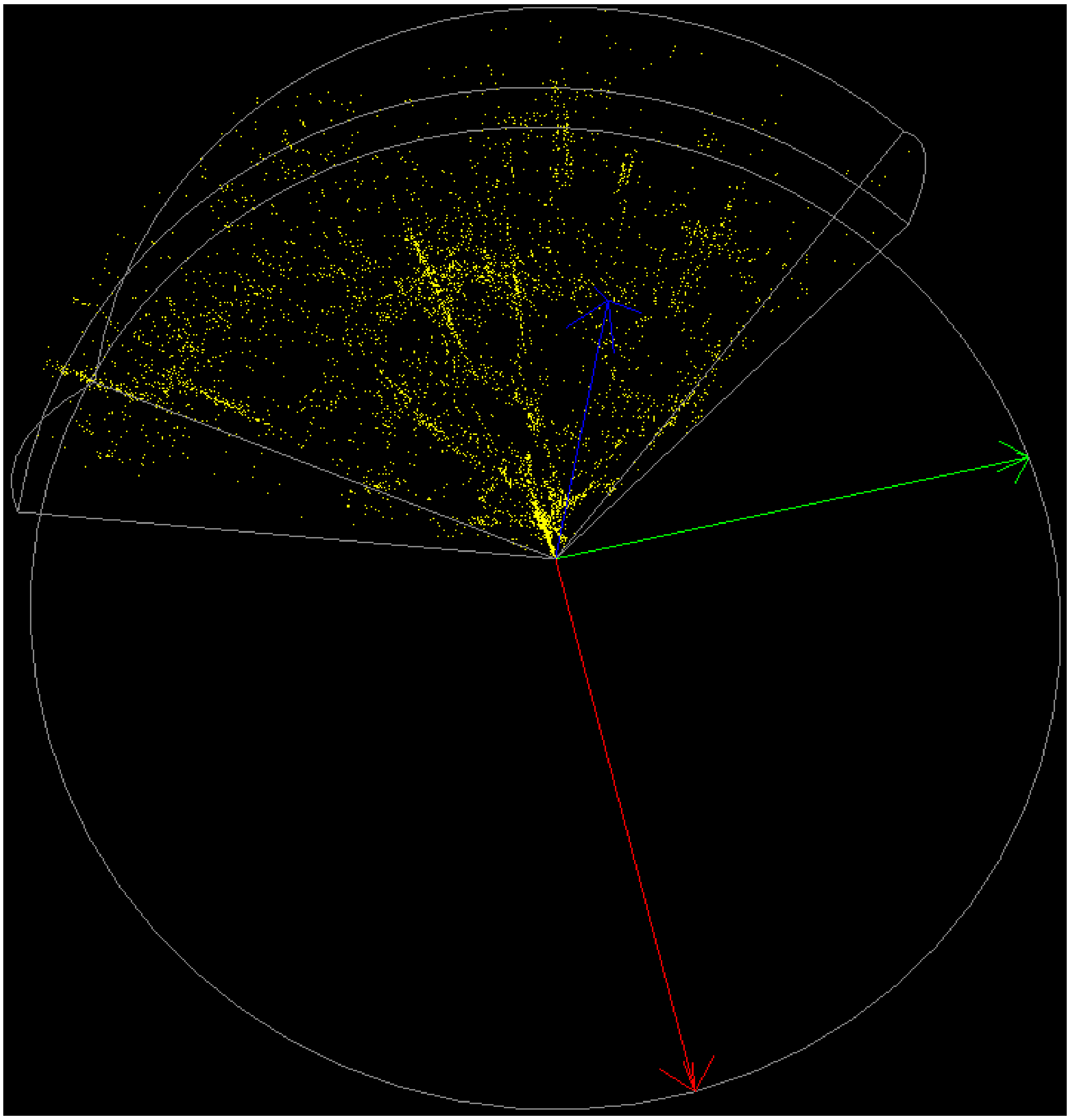


Figure 6: *The northern CfA slice,* $8^h < \alpha < 17^h, 8.5° < \delta < 44.5°, z < 0.04$

References

da Costa, L., Geller, M.J., Pellegrini, P.S., Latham, D.W., Fairall, A.P., Marzke, R.O., Willmer, C.N.A., Huchra, J.P., Calderon, J.H., Ramella, M., Kurtz, M.J., 1994, Astrophys. J. Lett., 424, L1-L4

Geller, M.J., Huchra, J.P., 1989, Science, 246, 857 (CfA2)

Nordlund, Å., Galsgaard K., "3D Dynamics & Visualization", 1994, Lecture Notes, NBIfAFG, Astronomical Observatory.

White, S.D.M., 1988, in "The Minnesota Lectures on Clusters of Galaxies and Large-Scale Structure", Ed. J.M.Dickey, Astronomical Society of the Pacific Conference Series Vol. 5

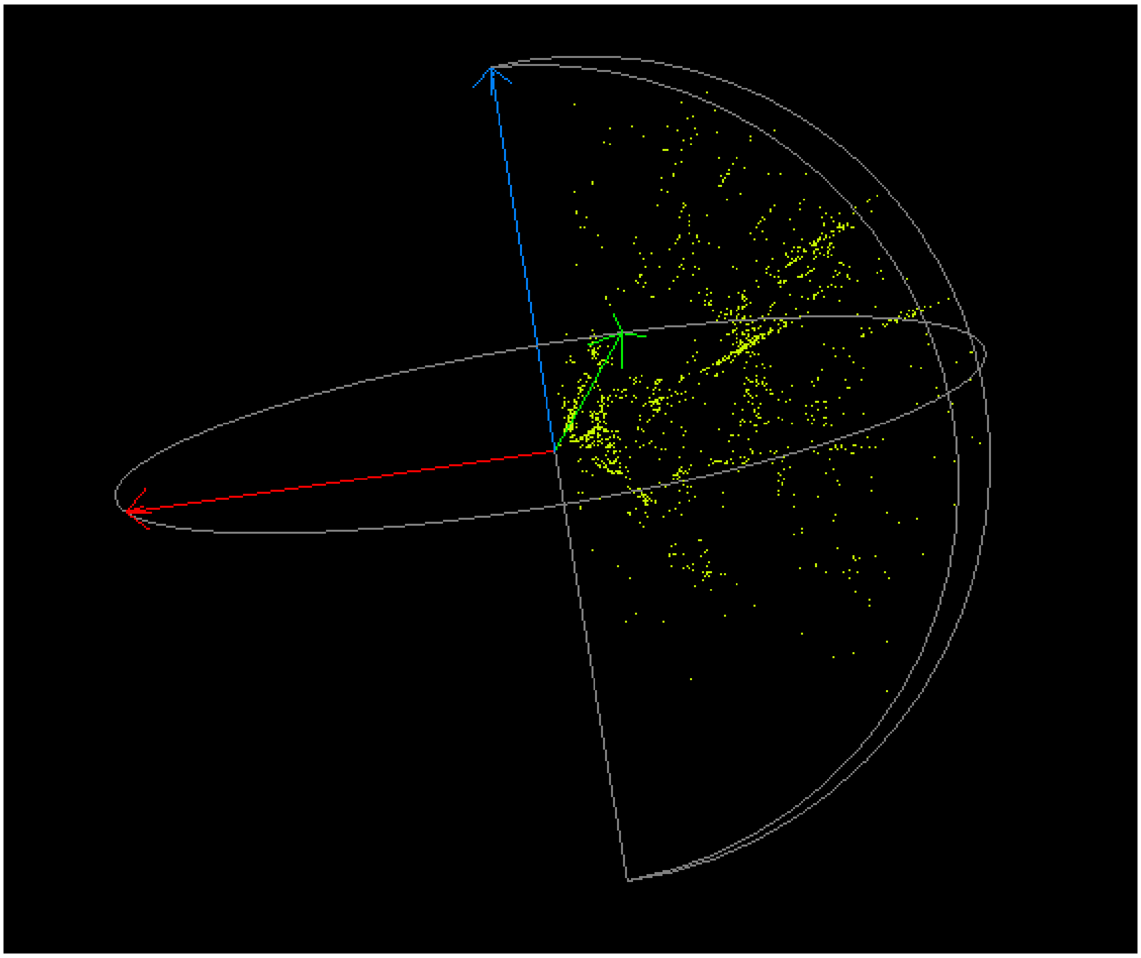


Figure 7: 15°*α-slices of the innermost data*, $z < 0.04$

A The IDL procedure ascii_to_bin

```
pro ascii_to_bin, file_in, f_outpre, ndata,$
                  z_low, z_high, a_low, a_high, d_low, d_high

;   Bo Milvang-Jensen, May 20, 1994.

    if n_params() eq (0) then begin
      print
      print, 'Procedure ascii_to_bin'
      print
      print, 'Usage:'
      print, 'ascii_to_bin, file_in, f_outpre, ndata,$
      print, '              z_low, z_high, a_low, a_high, d_low, d_high
      print
      print, 'Description:'
      print, 'This procedure converts an ascii data file, with alpha, delta, and'
      print, 'redshift, to a format suitable for Explorer'
      print
      print, 'file_in : name of data ascii input file with alpha, delta, redshift'
      print, 'f_outpre: prefix of output files -'
      print, '          binary output file will get suffix .dat'
      print, '          header output file will get suffix .idl'
      print, 'ndata   : number of datapoint in file_in'
      print, 'z_low   : minimum redshift value'
```

```
  print, 'z_high  : maximum redshift value'
  print, 'a_low   : minimum alpha    value'
  print, 'a_high  : maximum alpha    value'
  print, 'd_low   : mimimum delta    value'
  print, 'd_high  : maximum delta    value'
  print
  return
end

n_data_before = ndata

calcdata, file_in, ndata, a, maxv,$
  z_low, z_high, a_low, a_high, d_low, d_high, new_n

diam = 1.0 ; diameter of balls

dat_name = f_outpre + '.dat'    ; binary file (.dat)
openw,  lun, dat_name, /GET_LUN
writeu, lun,replicate(diam,ndata)
for i=0,2 do writeu, lun, a(*,i)
close, lun
free_lun, lun

idl_name = f_outpre + '.idl'   ; ascii file (.idl)
openw,  lun, idl_name, /GET_LUN
printf, lun, '# AVS field file'
printf, lun, '#'
printf, lun, '# Created by IDL procedure: ascii_to_bin'
spawn, 'date', date_result
printf, lun, '#'
printf, lun, '# Created date: ', date_result
printf, lun, '#'
spawn, 'pwd', pwd_result
printf, lun, '# IDL pwd was: ', pwd_result
printf, lun, '#'
printf, lun, '# Input ascii data file: ', file_in
printf, lun, '# Number of data point herein: ', n_data_before
printf, lun, '#'
printf, lun, '# z     low  cut:', z_low
printf, lun, '# z     high cut:', z_high
printf, lun, '# alpha low  cut:', a_low
printf, lun, '# alpha high cut:', a_high
printf, lun, '# delta low  cut:', d_low
printf, lun, '# delta high cut:', d_high
printf, lun, '#'
printf, lun, '# Number of data points left after cuts: ', new_n
printf, lun, '# Maximum absolute values (x,y,z): ', maxv
printf, lun, '#'
printf, lun, 'ndim=1'
printf, lun, 'dim1=', new_n
printf, lun, 'nspace=3'
printf, lun, 'veclen=1'
printf, lun, 'data=float'
printf, lun, 'field=irregular'
printf, lun, 'file="', dat_name, '"'
close,  lun
free_lun, lun
```

```
end
```

B The IDL procedure calcdata

```
pro calcdata, file_in, ndata, a, maxv, $
    z_low, z_high, a_low, a_high, d_low, d_high, new_n

;   Bo Milvang-Jensen, May 21, 1994.

    if n_params() eq (0) then begin
      print
      print, 'Procedure calcdata'
      print
      print, 'Usage:'
      print, 'calcdata, file_in, ndata, a, maxv, $'
      print, '          z_low, z_high, a_low, a_high, d_low, d_high, new_n'
      print
      print, 'Description:'
      print, 'This procedure converts an ascii data file, with alpha, delta, and'
      print, 'redshift, to cartesian coordinates, stored in the array a'
      print
      print, 'file_in : name of data ascii input file with alpha, delta, redshift'
      print, 'file_in: name of data ascii file with alpha, delta, redshift'
      print, 'ndata  : number of datapoint in file_in'
      print, 'a      : array which returns catesian coordinates'
      print, 'mav    : array which return maximum absolute values in x, y and z'
      print, 'z_low  : minimum redshift value'
      print, 'z_high : maximum redshift value'
      print, 'a_low  : minimum alpha    value'
      print, 'a_high : maximum alpha    value'
      print, 'd_low  : mimimum delta    value'
      print, 'd_high : maximum delta    value'
      print, 'new_n  : number of data points left'
      print
      return
    end

    diam = 1.     ; ball diameter

    a=fltarr(3,ndata)

    openr,    lun, file_in, /GET_LUN
    readf,    lun, a
    close,    lun
    free_lun, lun

    a=shift(a,1,0) ; before: alpha,    delta, redshift
   ; after : redshift, alpha, delta

    b=reform(a,ndata,3) ; transpose
    b(*,0) = a(0,*)
    b(*,1) = a(1,*)
    b(*,2) = a(2,*)
    a=b

    c = where(a(*,0) gt z_low, ndata2)
    if ndata2 ne 0 then begin
      d = a(c,*)
      a = d
      print, 'z     low  cut: ','before = ', ndata, '  after = ', ndata2
    endif
```

```
    ndata = ndata2

    c = where(a(*,0) lt z_high, ndata2)
    if ndata2 ne 0 then begin
      d = a(c,*)
      a = d
      print, 'z     high cut: ','before = ', ndata, '  after = ', ndata2
    endif
    ndata = ndata2

    c = where(a(*,1) gt a_low, ndata2)
    if ndata2 ne 0 then begin
      d = a(c,*)
      a = d
      print, 'alpha low  cut: ','before = ', ndata, '  after = ', ndata2
    endif
    ndata = ndata2

    c = where(a(*,1) lt a_high, ndata2)
    if ndata2 ne 0 then begin
      d = a(c,*)
      a = d
      print, 'alpha high cut: ','before = ', ndata, '  after = ', ndata2
    endif
    ndata = ndata2

    c = where(a(*,2) gt d_low, ndata2)
    if ndata2 ne 0 then begin
      d = a(c,*)
      a = d
      print, 'delta low  cut: ','before = ', ndata, '  after = ', ndata2
    endif
    ndata = ndata2

    c = where(a(*,2) lt d_high, ndata2)
    if ndata2 ne 0 then begin
      d = a(c,*)
      a = d
      print, 'delta high cut: ','before = ', ndata, '  after = ', ndata2
    endif
    ndata = ndata2

      ; NO scaling of redshift!!!
    a(*,1) = (a(*,1)/180)*!pi ; conversion to radians
    a(*,2) = (a(*,2)/180)*!pi ; conversion to radians

    a=new_pol_to_cart(a) ; conversion to cartesian coordinates

    maxv=fltarr(3)
    maxv(0)=max(abs(a(*,0)),min=minx)
    maxv(1)=max(abs(a(*,1)),min=miny)
    maxv(2)=max(abs(a(*,2)),min=minz)
    print
    print, 'Maximum absolute values (maxv) are: ' ,maxv

    new_n = ndata2
    print, 'Number of data points left (new_n) = ', new_n

end
```

C The IDL procedure new_pol_to_cart

```
function new_pol_to_cart,rtp
;
; transform  polare coordinats to cartesian coordinats.
;
; done by Klaus Galsgaard
;
; CORRECTED VERSION!!! (Bo, May 19, 1994)
; rtp(*,0): redshift
; rtp(*,1): alpha
; rtp(*,2): delta

on_error,2

;   Bo Milvang-Jensen, May 20, 1994.

    if n_params() eq (0) then begin
      print
      print, 'Function new_pol_to_cart'
      print
      print, 'Usage:'
      print, 'new_pol_to_cart, rtp'
      print
      print, 'Description:'
      print, 'This function converts from (redshift, alpha, delta) to'
      print, '(x, y, z).'
      print
      print, 'rtp: array, with'
      print, '     rtp(*,0) = redshit'
      print, '     rtp(*,1) = alpha'
      print, '     rtp(*,2) = delta'
      print
      rc = -1 ; ?!
      return, rc
    end
s=size(rtp)
rc=fltarr(s(1),3,/nozero)

rc(*,0)=rtp(*,0)*sin(!pi/2.-rtp(*,2))*cos(rtp(*,1))
rc(*,1)=rtp(*,0)*sin(!pi/2.-rtp(*,2))*sin(rtp(*,1))
rc(*,2)=rtp(*,0)*cos(!pi/2.-rtp(*,2))

return,rc
end
```

D The IDL procedure lines

```
PRO lines, file_in, file_out

; by
; Lars C

; modified by
; Bo, May 21, 1994

    if n_params() eq (0) then begin
      print
      print, 'Procedure lines'
      print
      print, 'Usage:'
      print, 'lines, file_in, file_out'
      print
      print, 'Description:'
      print, 'This procedure converts an data file to a binary file in a format'
      print, 'suitable for the Explorer module Line2Geo'
      print
      print, 'file_in : name of data ascii input file - example:'
      print, '(NB: you should probably not include comments)'
      print
      print, '2       # number of points to be connected with a line'
      print, '0.0     # x-coordinate of the first point'
      print, '0.0     # y-coordinate of the first point'
      print, '0.0     # z-coordinate of the first point'
      print, '1.0     # x-coordinate of the second point'
      print, '0.0     # y-coordinate of the second point'
      print, '0.0     # z-coordinate of the second point'
      print, '0.99    # R - color of first object'
      print, '0.0     # G - color of first object'
      print, '0.00    # B - color of first object'
      print, '2       # number of points to be connected with a line'
      print, '1.0     # x-coordinate of the first point'
      print, '1.0     # y-coordinate of the first point'
      print, '1.0     # z-coordinate of the first point'
      print, '2.0     # x-coordinate of the second point'
      print, '2.0     # y-coordinate of the second point'
      print, '2.0     # z-coordinate of the second point'
      print, '0.99    # R - color of second object'
      print, '0.0     # G - color of second object'
      print, '0.00    # B - color of second object'
      print, '-1      # no more data marker'
      print
      print, 'file_out: name of binary output file'
      print
      return
    end

   n = 0L
   data = 0.0
   openr, lun1, file_in,  /get_lun ; ascii input file
   openw, lun2, file_out, /get_lun ; binary output file

   while (n ne -1) do begin
     readf,  lun1, n
     writeu, lun2, n
```

```
    for i=0, (n+1)*3-1 do begin
      readf,  lun1, data
      writeu, lun2, data
    endfor
  endwhile

  close, lun1
  close, lun2
  free_lun, lun1
  free_lun, lun2

end
```

E The IDL procedure equa

```
pro equa, file_out, z_high

; Bo Milvang-Jensen, May 21, 1994.

    if n_params() eq (0) then begin
      print
      print, 'Procedure equa'
      print
      print, 'Usage:'
      print, 'equa, file_out, z_high'
      print
      print, 'Description:'
      print, 'This procedure makes a binary file suitable for the Explorer'
      print, 'module Line2Geo showing the equator'
      print
      print, 'file_out: output file name'
      print, 'z_high  : maximum redshift (i.e. radius)'
      print
      return
    end

    a_low  =   0.0
    a_high = 360.0
    d_low  =   0.0
    d_high =   0.0

    arcl = 6.0

    nde = fix((a_high-a_low)/arcl)

    de = fltarr(2*(2+nde),3) ; arcs with delta = constant

    for i=0, nde+1 do begin  ; delta = d_low
      de(i,0) = z_high
      de(i,1) = a_low + (1.0*i/(nde+1)) * (a_high-a_low)
      de(i,2) = d_low
    endfor

    for i=0, nde+1 do begin  ; delta = d_high
      de(i+nde+2,0) = z_high
      de(i+nde+2,1) = a_low + (1.0*i/(nde+1)) * (a_high-a_low)
      de(i+nde+2,2) = d_high
    endfor

    de(*,1) = (de(*,1)/180)*!pi ; conversion to radians
    de(*,2) = (de(*,2)/180)*!pi ; conversion to radians

    de = new_pol_to_cart(de) ; convert to cartesian coordiantes


    openw, lun, file_out, /get_lun

    for i=0, 1 do begin
      writeu, lun, 1L*(nde+2)
      for j=0, nde+1 do begin
        writeu, lun, de(i*(nde+2)+j,*)
      endfor
```

```
      writeu, lun, 0.50     ; R
      writeu, lun, 0.50     ; G
      writeu, lun, 0.50     ; B
    endfor

    writeu, lun, -1L
    close, lun
    free_lun, lun

end
```

F The IDL procedure alphatour

```
pro alphatour, file_in, f_outpre, ndata, nslices, width

    ; May 24, 1994, Bo.
    ; June 29, 1994, Bo.

    if n_params() eq (0) then begin
      print
      print, 'Procedure alphatour'
      print
      print, 'Usage:'
      print, 'pro alphatour, file_in, f_outpre, ndata, nslices, width'
      print
      print, 'Description'
      print, 'This procedure makes the files for an "alpha-tour"!'
      print, 'NB: This version has a slight problem with making the last slice!'
      print
      print, 'file_in : name of data ascii input file with alpha, delta, redshift'
      print, "f_outpre: output name prefix (e.g. '/l/tmp/milvang/video/alpha-tour/'"
      print, 'ndata   : number of datapoint in file_in'
      print, 'nslices : number of slices (e.g. 250) - must be < 1000'
      print, 'width   : width of slices in degrees (e.g. 15.0)'
      return
    end

    print

    for i=0L, nslices-1 do begin
      print, 'i = ', i
      nulls = ''
      if ((i ge  0) and (i lt  10)) then nulls = '00'
      if ((i ge 10) and (i lt 100)) then nulls = '0'
      file_out = f_outpre+nulls+strtrim(string(i),1)+'.line'
      f2       = f_outpre+nulls+strtrim(string(i),1)
      slicelines,  file_out, $
   0.0, 0.04, $
   i*360./nslices, width+i*360./nslices, $
   -90.0, 90.0
      ascii_to_bin, file_in, f2, ndata, $
   0.0, 0.04, $
   i*360./nslices, width+i*360./nslices, $
   -90.0, 90.0
    endfor

end
```

G The IDL procedure slicelines

```
pro slicelines, file_out, z_low, z_high, a_low, a_high, d_low, d_high

; Bo Milvang-Jensen, May 21, 1994.

    if n_params() eq (0) then begin
      print
      print, 'Procedure slicelines'
      print
      print, 'Usage:'
      print, 'slicelines, file_out, z_low, z_high, a_low, a_high, d_low, d_high'
      print
      print, 'Description:'
      print, 'This procedure makes a binary file suitable for the Explorer'
      print, 'module Line2Geo'
      print
      print, 'file_out: output file name'
      print, 'z_low   : minimum redshift value'
      print, 'z_high  : maximum redshift value'
      print, 'a_low   : minimum alpha    value'
      print, 'a_high  : maximum alpha    value'
      print, 'd_low   : mimimum delta    value'
      print, 'd_high  : maximum delta    value'
      print
      return
    end

    a = fltarr(8,3)

    a(0,0) = z_low  ; redshift - point 1
    a(0,1) = a_low  ; alpha
    a(0,2) = d_low  ; delta
    a(1,0) = z_high ;            point 2
    a(1,1) = a_low
    a(1,2) = d_low

    a(2,0) = z_low  ;            point 3
    a(2,1) = a_high
    a(2,2) = d_low
    a(3,0) = z_high ;            point 4
    a(3,1) = a_high
    a(3,2) = d_low

    a(4,0) = z_low  ;            point 5
    a(4,1) = a_low
    a(4,2) = d_high
    a(5,0) = z_high ;            point 6
    a(5,1) = a_low
    a(5,2) = d_high

    a(6,0) = z_low  ;            point 7
    a(6,1) = a_high
    a(6,2) = d_high
    a(7,0) = z_high ;            point 8
    a(7,1) = a_high
    a(7,2) = d_high

    a(*,1) = (a(*,1)/180)*!pi ; conversion to radians
```

```
a(*,2) = (a(*,2)/180)*!pi ; conversion to radians

a = new_pol_to_cart(a) ; convert to cartesian coordiantes


arcl = 6.0

nal = fix((d_high-d_low)/arcl)

al = fltarr(2*(2+nal),3) ; arcs with alpha = constant

for i=0, nal+1 do begin  ; alpha = a_low
  al(i,0) = z_high
  al(i,1) = a_low
  al(i,2) = d_low + (1.0*i/(nal+1)) * (d_high-d_low)
endfor

for i=0, nal+1 do begin  ; alpha = a_high
  al(i+nal+2,0) = z_high
  al(i+nal+2,1) = a_high
  al(i+nal+2,2) = d_low + (1.0*i/(nal+1)) * (d_high-d_low)
endfor

al(*,1) = (al(*,1)/180)*!pi ; conversion to radians
al(*,2) = (al(*,2)/180)*!pi ; conversion to radians

al = new_pol_to_cart(al) ; convert to cartesian coordiantes


nde = fix((a_high-a_low)/arcl)

de = fltarr(2*(2+nde),3) ; arcs with delta = constant

for i=0, nde+1 do begin  ; delta = d_low
  de(i,0) = z_high
  de(i,1) = a_low + (1.0*i/(nde+1)) * (a_high-a_low)
  de(i,2) = d_low
endfor

for i=0, nde+1 do begin  ; delta = d_high
  de(i+nde+2,0) = z_high
  de(i+nde+2,1) = a_low + (1.0*i/(nde+1)) * (a_high-a_low)
  de(i+nde+2,2) = d_high
endfor

de(*,1) = (de(*,1)/180)*!pi ; conversion to radians
de(*,2) = (de(*,2)/180)*!pi ; conversion to radians

de = new_pol_to_cart(de) ; convert to cartesian coordiantes


openw, lun, file_out, /get_lun
for i=0, 3 do begin
  writeu, lun, 2L
  writeu, lun, a(2*i+0,*) ; points 1, 3, 5 and 7
  writeu, lun, a(2*i+1,*) ; points 2, 4, 6 and 8
  writeu, lun, 0.50     ; R
  writeu, lun, 0.50     ; G
  writeu, lun, 0.50     ; B
```

```
    endfor

    for i=0, 1 do begin
      writeu, lun, 1L*(nal+2)
      for j=0, nal+1 do begin
        writeu, lun, al(i*(nal+2)+j,*)
      endfor
      writeu, lun, 0.50     ; R
      writeu, lun, 0.50     ; G
      writeu, lun, 0.50     ; B
    endfor

    for i=0, 1 do begin
      writeu, lun, 1L*(nde+2)
      for j=0, nde+1 do begin
        writeu, lun, de(i*(nde+2)+j,*)
      endfor
      writeu, lun, 0.50     ; R
      writeu, lun, 0.50     ; G
      writeu, lun, 0.50     ; B
    endfor

    writeu, lun, -1L
    close, lun
    free_lun, lun

end
```

H The IDL procedures loadanim and runanim

```
PRO runanim,nend,delay,backward=backward,nstart=nstart

  if n_params() eq 0 then begin
    print,'runanim,nend,delay,backward=backward,nmin=nmin'
    print,'nend    : number of the last frame to show'
    print,'delay   : wait between frames'
    print,'backward: show frames backward'
    print,'nstart  : number of start frame (default=0)'
    return
  endif

  COMMON canim, s, nx, ny, dx, dy
  if keyword_set(nstart) eq 0 then nstart = 0

  wset,0
  if keyword_set(backward) eq 0 then begin
     step = 1
     is = 0
     ie = nx-1
     js = 0
     je = ny-1
     test = 0
  endif else begin
     step = -1
     ie = 0
     is = nx-1
     je = 0
     js = ny-1
     test = nx*ny
  endelse

  k=1
  for i=is,ie,step do begin
    for j=js,je,step do begin
      if(test le nend and test ge nstart) then begin
device,copy=[i*s(2),j*s(3),s(2),s(3),dx,dy,1]
        wait,delay
        if k ge nend then goto, finis
        k=k+1
      endif
      test = test + step
    end
  end
finis:
END

PRO loadanim,nstart,nfile,file,ext,compress=compress,temp=temp,$
     no_zero=no_zero, nu=nu, m=m

; stone age/Aake init version
; 25.05-94/Bo    included 8 bit conversion
; 26.05-94/kg  included file and ext to the input
; 26.05-94/kg,Bo included nstart, /compress, /no_zero
; 26.05-94/Bo    included temp=temp
; 26.05-94/kg    included backward frame stepping in runanim
; ??.06-94/kg    included nu=nu
```

```
COMMON canim, s, nx, ny, dx, dy

if n_params() eq 0 then begin
   print,'loadanim,nstart,nfile,file,ext,/compress,temp=temp,/no_zero,/nu'
   print,'nstart    : number of the first data file
   print,'nfile     : last file to read'
   print,'file      : main file name'
   print,"ext       : extension on file name, e.g. '.tiff'"
   print,'/compress : used when the data files are compressed'
   print,"temp      : temporary filename when uncompressing, e.g. '/tmp/blaaa'"
   print,'/no_zero  : used when no leading zeros in numbers'
   print,'/nu       : when set numbers appear on the animation'
   return
endif
COMMON c_fade,r,g,b
set_plot,'X'
device,pseudo=8
!order=1

ny=7l
nx=long((nfile+1-nstart)/ny)+1l
print,nfile-nstart+1,nx,ny

k=nstart
if keyword_set(no_zero) ne 0 then begin
   file_name=strcompress(file+str(k)+ext)
endif else begin
    if (k lt 10) then file_name =strcompress(file+'00'+str(k)+ext)
    if (k ge 10 and k lt 100) then file_name =strcompress(file+'0'+str(k)+ext)
    if (k ge 100 and k lt 1000) then file_name =strcompress(file+str(k)+ext)
endelse
if keyword_set(compress) ne 0 then begin
   if n_elements(temp) eq 0 then temp='tmp' ; default is 'tmp'
   cmd='zcat '+file_name+'.Z > '+ temp
   print,cmd
   spawn,cmd
   file_name=temp
endif

im=tiff_read(file_name)

s=size(im)
im=reform(color_quan(im, 1, R, G, B, COLORS=256, GET_TRANSLATION=ctrans, /MAP_ALL))

window,1,/pix,xsize=nx*s(2),ysize=ny*s(3)

m=0
xsize=775>s(2)
ysize=580>s(3)
dx=((xsize-s(2)+2*m)>0)/2
dy=((ysize-s(3)+2*m)>0)/2
window,0,xsize=xsize,ysize=ysize
tvlct, R, G, B

wset,1 & erase
for i=0,nx-1l do begin
  for j=0,ny-1l do begin
    if keyword_set(no_zero) ne 0 then begin
       file_name=strcompress(file+str(k)+ext)
```

```
      endif else begin
          if (k lt 10) then file_name =strcompress(file+'00'+str(k)+ext)
          if (k ge 10 and k lt 100) then file_name =strcompress(file+'0'+str(k)+ext)
          if (k ge 100 and k lt 1000) then file_name =strcompress(file+str(k)+ext)
      endelse
      if keyword_set(compress) ne 0 then begin
         cmd='zcat '+file_name+'.Z > '+ temp
         print,cmd
         spawn,cmd
 file_name=temp
      endif
      im=tiff_read(file_name)
      im=reform(color_quan(im, 1, R, G, B, COLORS=256, TRANSLATION=ctrans))
;      im = rotate(rotate(im,2),5)
      wset,0
      if (k ne nstart) then xyouts,.1,.1,/nor,str(k-1),col=0
      tv,im(m:s(2)-m-1,m:s(3)-m-1),dx,dy
      xyouts,.1,.1,/nor,str(k),col=10
      wset,1
      tv,im(m:s(2)-m-1,m:s(3)-m-1),i*s(2),j*s(3)
      if keyword_set(nu) ne 0 then xyouts,10+i*s(2),10+j*s(3),/dev,str(k)
      if k ge nfile then goto, display
      k=k+1
    end
  end

display:

  if keyword_set(compress) ne 0 then begin
    cmd='\rm '+temp
    print,cmd
    spawn,cmd
  endif

  wset,0
  erase
  wset,1
  runanim,nfile-nstart+1,.1

  print, 'there are '+str(nfile-nstart+1)+' frames in the sequence'

END
```

I The IDL procedure plotdata

```
pro plotdata, a, maxv, ax, az

;   Bo Milvang-Jensen, May 20, 1994.

    if n_params() eq (0) then begin
      print
      print, 'Procedure plotdata'
      print
      print, 'Usage:'
      print, 'plotdata, a, maxv, ax, az
      print
      print, 'Description:'
      print, 'This procedure plots the data in the array a.'
      print, 'Run the procedure calcdata first.'
      print
      print, 'file_in : name of data ascii input file with alpha, delta, redshift'
      print, 'a       : array which returns catesian coordinates'
      print, 'mav     : array which return maximum absolute values in x, y and z'
      print, 'ax      : ax angle (?)'
      print, 'az      : az angle (?)'
      print
      return
    end

    kgbox, -[maxv(0),maxv(1),maxv(2)],[maxv(0),maxv(1),maxv(2)],ax,az
    plots, a(*,0), a(*,1), a(*,2), psym=3, /t3d

end
```

J The IDL procedure kgbox

```
pro kgbox,rlo,rhi,ax,az,nosurf=nosurf
;
; make t3d transformation for a box with upper and lower boundary defined
; by rlo and rhi, turned the angles ax, az.
;
; done by Klaus Galsgaard
;
; name changes from box to kgbox (Bo, May 20, 1994)

on_error,2

if n_params() eq (0) then begin
print
print,'Procedure kgbox (was: box)'
print
print,'kgbox,rlo,rhi,ax,az,nosurf=nosurf'
print,'rlo,rhi : lower and upper boundary of box'
print,'ax,az : turning angles around x and z axis'
print,'nosurf : make a box with out changing t3d,'
                print,'         : 1 - without tick marks, ne (0,1) make new t3d setup.'
                print,'         : 2 - make a setup with out real t3d + box and tick marks'
return
end

if n_elements(ax) eq 0 then ax=30
if n_elements(az) eq 0 then az=30

x0=rlo(0) & x1=rhi(0)
y0=rlo(1) & y1=rhi(1)
z0=rlo(2) & z1=rhi(2)
z=fltarr(2,2)
z(0,*)=z0 & z(1,*)=z1

if keyword_set(nosurf) eq 0 then begin
erase
t3dset,rlo,rhi,ax=ax,az=az,scale=.57,per=5
     surface,z,[x0,x1],[y0,y1],/nodata,/t3d
endif else begin
  if nosurf eq 1 then begin
     plots,[x1,x0,x0,x0],[y0,y0,y1,y1],[z0,z0,z0,z1],/t3d,/data
  endif else if nosurf eq 2 then begin
     surface,z,[x0,x1],[y0,y1],ax=ax,az=az,/nodata,/save
  endif else begin
     t3dset,rlo,rhi,ax=ax,az=az,scale=.57,per=5
     return
  endelse
end
plots,[x0,x1,x1],[y1,y1,y0],[z0,z0,z0],/t3d,/data
plots,[x0,x1,x1],[y1,y1,y0],[z1,z1,z1],/t3d,/data
plots,[x1,x1],[y1,y1],[z0,z1],/t3d,/data
plots,[x1,x1],[y0,y0],[z0,z1],/t3d,/data
end
```

K The IDL procedure t3dset

```
pro t3dset,rlo,rhi,ax=ax,ay=ay,az=az,per=per,scale=scale, len=len
;
;  make a t3d setup for a given box of data
;
; done by Klaus Galsgaard
;
if n_params() eq 0 then begin
print,'t3dset,rlo,rhi,ax=ax,ay=ay,az=az,per=per'
print,'rlo,rhi : lower and upper boundary for box'
print,'ax,ay,az : the wanted view angle'
print,'per      : perspektiv distance'
print,'scale : scale of plot'
print,'len      : length that the plot is moved
return
endif

if keyword_set(ax)    eq 0 and n_elements(ax) eq 0 then ax=30
if keyword_set(ay)    eq 0 and n_elements(ay) eq 0 then ay=0
if keyword_set(az)    eq 0 and n_elements(az) eq 0 then az=30
if keyword_set(per)   eq 0 then per=30
if keyword_set(scale) eq 0 then scale=.57
if keyword_set(len)   eq 0 then len=.5

        !x.s=[-rlo(0),1.]/(rhi(0)-rlo(0))
        !y.s=[-rlo(1),1.]/(rhi(1)-rlo(1))
        !z.s=[-rlo(2),1.]/(rhi(2)-rlo(2))

        u=[1,1,1]

        t3d,/reset,tr=-len*u
t3d,rot=[0,ay,0]
        t3d,rot=[0,0,az]
        t3d,rot=[ax-90,0,0]
        t3d,sc=scale*u
t3d,per=per
        t3d,tr=len*u

end
```

L The IDL procedure galcount

```
pro galcount, file_in, f_outpre, ndata, n_box,$
              z_low, z_high, a_low, a_high, d_low, d_high

;   Bo Milvang-Jensen, May 20, 1994.

    if n_params() eq (0) then begin
      print
      print, 'Procedure galcount'
      print
      print, 'Description:'
      print, 'This procedure counts the number of galaxies (datapoints) in a box'
      print, 'which encompasses the data, and which is of size n_box**3'
      print, 'The output is suitable for an Explorer isosurface'
      print
      print, 'Usage:'
      print, 'galcount, file_in, f_outpre, ndata, n_box,$
      print, '          z_low, z_high, a_low, a_high, d_low, d_high
      print
      print, 'file_in : name of data ascii input file with alpha, delta, redshift'
      print, 'f_outpre: prefix of output files -'
      print, '          binary output file will get suffix .dat'
      print, '          header output file will get suffix .idl'
      print, 'ndata   : number of datapoint in file_in'
      print, 'n_box   : size of 3D box (e.g. 128)'
      print, 'z_low   : minimum redshift value'
      print, 'z_high  : maximum redshift value'
      print, 'a_low   : minimum alpha    value'
      print, 'a_high  : maximum alpha    value'
      print, 'd_low   : mimimum delta    value'
      print, 'd_high  : maximum delta    value'
      print
      return
    end

    n_data_before = ndata

    calcdata, file_in, ndata, a, maxv,$
      z_low, z_high, a_low, a_high, d_low, d_high, new_n

    box=fltarr(n_box,n_box,n_box)

    L = 2 * max(maxv)
    dL = L/n_box

    for t=0L, ndata-1 do begin
        x = a(t,0)
y = a(t,1)
z = a(t,2)
i = fix((x+L/2)/dL)
j = fix((y+L/2)/dL)
k = fix((z+L/2)/dL)
if (i eq n_box) then i=i-1
if (j eq n_box) then j=j-1
if (k eq n_box) then k=k-1
; print, t, x, y, z, i, j, k
box(i,j,k) = box(i,j,k)+1.
    endfor
```

```
    dat_name = f_outpre + '.dat'    ; binary output file (.dat)
    openw,  lun, dat_name, /GET_LUN
    writeu, lun, box
    close,  lun
    free_lun, lun

    idl_name = f_outpre + '.idl'    ; header output file (.idl)
    openw,  lun, idl_name, /GET_LUN
    printf, lun, '# IDL field description file'
    printf, lun, '#'
    printf, lun, '#'
    printf, lun, '# Created by IDL procedure: galcount'
    spawn, 'date', date_result
    printf, lun, '#'
    printf, lun, '# Created date: ', date_result
    printf, lun, '#'
    spawn, 'pwd', pwd_result
    printf, lun, '# IDL pwd was: ', pwd_result
    printf, lun, '#'
    printf, lun, '# Input ascii data file: ', file_in
    printf, lun, '# Number of data point herein: ', n_data_before
    printf, lun, '#'
    printf, lun, '# z     low  cut:', z_low
    printf, lun, '# z     high cut:', z_high
    printf, lun, '# alpha low  cut:', a_low
    printf, lun, '# alpha high cut:', a_high
    printf, lun, '# delta low  cut:', d_low
    printf, lun, '# delta high cut:', d_high
    printf, lun, '#'
    printf, lun, '# Number of data points left after cuts: ', new_n
    printf, lun, '# Maximum absolute values (x,y,z): ', maxv
    printf, lun, '#'
    printf, lun, '#'
    printf, lun, 'ndim=3'
    printf, lun, ''
    printf, lun, 'dim1=', n_box
    printf, lun, 'dim2=', n_box
    printf, lun, 'dim3=', n_box
    printf, lun, ''
    printf, lun, 'veclen=1'
    printf, lun, ''
    printf, lun, 'data=float'
    printf, lun, 'field=uniform'
    printf, lun, ''
    printf, lun, 'frames=1'
    printf, lun, 'file="', dat_name, '"'
    close,  lun
    free_lun, lun

end
```

M The IDL procedure sources

```
pro sources

; Bo, May 24, 1994.

    ; pattern for the files
    spawn, 'ls /l/tmp/lars_c/video/datafinal/sources/source?',  one
    spawn, "wc source? | awk '{print $1}' -", wc1
    spawn, 'ls /l/tmp/lars_c/video/datafinal/sources/source??', two
    spawn, "wc source?? | awk '{print $1}' -", wc2

    t1 = size(one)
    t2 = size(two)
    len1 = t1(1)
    len2 = t2(1)

    reads, wc1, lines, words, chars, name

    print, 'Converting ascii source files to explorer format'
    print

    for i=0, len1-1 do begin
      reads, wc1(i), ndataf
      ndata = fix(ndataf)
      print, 'file: ', one(i), '  ndata: ', ndata
      if (ndata gt 1) then ascii_to_bin, one(i), one(i), ndata, $
                    0.0, 10.0, 0.0, 360.0, -90.0, 90.0
    endfor

    for i=0, len2-1 do begin
      reads, wc2(i), ndataf
      ndata = fix(ndataf)
      print, 'file: ', two(i), '  ndata: ', ndata
      if (ndata gt 1) then ascii_to_bin, two(i), two(i), ndata, $
                    0.0, 10.0, 0.0, 360.0, -90.0, 90.0
    endfor

end
```

N The C-shell script sourceawk

```
#Script for use of awk with wildcards. LLC 13/5-94
#!/bin/csh
foreach f (source*)
   awk '{print $1,$2,($3/299792)}' $f > /l/tmp/lars_c/video/sources/$f
end
```

O The C-shell script sourcesort

```
#Script for use of sort with wildcards. LLC 13/5-94
#!/bin/csh
foreach f (source*)
   sort -n +2 /l/tmp/lars_c/video/sources/$f > /l/tmp/lars_c/video/sources/sorts/$f
end
```