Electronic Polar Alignment Scope

Rick Kellogg 2012-7-30

Summary

A traditional polar scope can be used to align an equatorial mount in a few minutes. However, the alignment achieved is typically not that accurate due to misalignment of the polar scope axis with the mount axis, or the polar scope reticle not being centered. I typically can only get to within 0.1 degree of the celestial pole with my Losmandy polar scope.

Extending the ideal of using a polar scope for aligning to an electronic scope – I use an SBIG STV with a 75 mm lens as an electronic finder. This combination provides a 3.7 by 2.7 degree field of view (FOV).

Setting the DEC to 90 degrees, Polaris and UMI λ are in the field of view – regardless of the RA setting. By taking 2 pictures with different RA (differing by 4-6 hours or 60-90 degrees),

1) the pixel location of the mount’s RA axis can be determined, and
2) the pixel location of the Celestial pole can be determined (knowing the current epoch coordinates of Polaris and UMI Lambda).

Then the offset from the mount’s RA axis to the Celestial pole is added to the location of Polaris (or UMI Lambda) to act as a target to move the scope via azimuth or altitude controls.

I have automated the process using the SBIG utility that comes with the STV (STV Remote), and a MATLAB script that I wrote to make the computations.

The typical polar alignment process can be easily completed in 15 minutes.

Aligning using a Polar scope

There are multiple types of polar scopes used for alignment. One form uses a bubble level with date and hour scales to set the proper orientation of the polar scope. Then the polar scope has a reticle with the location of Polaris on it.

You set the date/time scales to the current date/time – which rotates the scope’s bubble level relative to the reticle. You then turn the scope to level the bubble. Once this is done, the mount’s altitude and azimuth adjustments are used to align Polaris to a mark on the reticle corresponding to where Polaris is for the current year. I have not used a polar scope of this type and can not comment on how well it works.
A second type of polar scope (used by Losmandy – and the type I have) uses has a wider field of view – and has reticle marks for Polaris and UMI δ (and a 3rd star) covering a 30 year period. The reticle for this scope is shown in Figure 1.

![Figure 1. Losmandy type Polar scope reticle. To Polar align you: 1) Rotate the scope to roughly orient the Big Dipper and Cassiopeia (which are not visible in the scope). 2) Adjust Mount Azimuth and Altitude (while fine tuning the scope rotation) to align the 3 stars shown.](image)

In addition to the star marks, the reticle has orientation aids – the Big Dipper and Cassiopeia. While these 2 constellations are not visible in the scope, procedure is to rotate the scope such that the reticle orientation matches the position of the 2 constellations in the sky. Then the altitude/azimuth controls of the mount are used (with slight corrections to the rotation of the scope) to match the 3 stars in the scope to the reticle locations. Using this type of polar scope, a precise polar alignment should be possible. There are two issues that usually prevent a precise alignment. Issue 1 – the polar scope axis does not coincide with the telescope mount axis. On
Losmandy mounts, there is slop in the tube that holds the polar scope, so that the scope can wobble as it is rotated. I have used electrical tape around tube to take of the space with some success. Issue 2 - the center of the reticle does not match the center of rotation of the polar scope. Most of these reticles are not very well centered, and while there are adjustment screws – the web is full of stories of people breaking the reticle while trying to adjust out the error.

As a result – using a polar scope (at least of the Losmandy type) will likely not result in a very good polar alignment. My experience is I usually only can get to within about 0.1 degree of the celestial pole. While that is good enough for a 4 minute exposure, it is not for a 40 minute one. This has led me to extending the process to use an electronic finder and some software to emulate the function of the polar scope.

**STV**

The STV is a small CCD imager from SBIG intended for guiding and planetary imaging. It was introduced in 2000 and is no longer in production, but has many useful features. One is as an electronic finder. In the efinder mode – a picture is constantly being taken, and a crosshair is placed on the screen. A “set crosshair” function allows the user to change the pixel location of the crosshair. This is useful in setting a target location for a star (e.g. Polaris).

The STV has a 4.85 mm by 3.55 mm imager with 7.4 micron pixels. The STV imaging head – with the 75mm/F1.8 Fujinon Factory Automation lens that I use as an efinder is shown in Figure 2.
Procedure for using Electronic Finder for polar alignment

The efinder is used to take 2 pictures of Polaris and UMI Lambda – at the same DEC angle (90 degrees), but at different RA hours (differing by 4-6 hours, or 60-90 degrees). The pixel location of Polaris and UMI Lambda are found in both pictures.
Figure 3. Picture 1  DEC = 90 degrees

Figure 4. Picture 2  DEC = 90 degrees - RA rotated by 67 degrees
By knowing the current celestial coordinates of 2 stars near the celestial pole, the pixel location of the celestial pole can be determined from a single image containing the 2 stars. The purpose of the second image is to determine the pixel location of a point that lies on the mount’s RA axis. Then the mount’s pointing error is just the difference between the mount’s RA axis and the celestial pole.

At the fixed DEC of 90 degrees, as the mount is rotated in RA, each star will sweep a circle around the same point. This point is along the mount’s RA axis.

We observe 2 points on each star’s circle (from the 2 pictures), draw a line drawn between the 2 points. Draw a 2nd line perpendicular to the 1st line at the center of the 1st line. This 2nd line will go through the mount’s RA axis. By doing this with 2 stars (Polaris and UMI Lambda), the pixel location of the mount’s RA axis is determined.

Figure 5. Find Polaris and UMI Lambda star locations for Pic 1 and Pic 2. Draw line 1P from Polaris pic 1 to Polaris pic 2. Then draw a line 2P, perpendicular to line 1P, that passes through the midpoint of line 1P. Now draw a line 1L from UMI λ pic 1 to UMI λ pic 2. Then draw a line 2L, perpendicular to line 1L, that passes through the midpoint of line 1L. The mount axis is located at the intersection of lines 2P and 2L.
In the 2nd picture, the pixel location of the Celestial pole is determined from the pixel locations of Polaris and UMI Lambda, and the current epoch (NOT J2000) RA and DEC coordinates of Polaris and UMI Lambda.

The RA and DEC coordinates of the 2 stars are converted to rectangular (X,Y) coordinates (I typically use a scale of degrees). The vector (A) from Polaris to UMI \( \lambda \) is formed by subtracting the (X,Y) coordinates of Polaris from UMI \( \lambda \). A second vector (B) is formed from Polaris to the celestial pole by subtracting the (X,Y) coordinates of Polaris from the Celestial pole. The vector (C) is formed by dividing vector (B) by vector (A).

(Note: these calculations are made using the current epoch coordinates of the 2 stars – and have nothing to do with the pictures.)

To find the celestial pole location in the 2nd picture, the vector (D) is formed by subtracting the pixel locations of Polaris from UMI \( \lambda \). The vector (D) is multiplied by the vector (C) and added to the pixel location of Polaris. The result is the pixel location of the celestial pole in the 2nd picture.

The mount’s axis pointing error is just the difference between pixel location of the Celestial pole, and the pixel location of the mount’s RA axis. The error is added to the locations of the 2 reference stars (Polaris and UMI Lambda) - to act as target locations - where these stars need to be for the mount to be polar aligned.

(Note: the mount should have been left on – and still in the same position it was for the 2nd picture).

Now the efinder is used to change the crosshair location to the target location for Polaris, and the mount’s azimuth and altitude adjustments are used to move Polaris to the target location.
Figure 6. The Celestial pole location in the 2\textsuperscript{nd} pictures is determined using this procedure. 1) The RA/DEC coordinates are converted to (X,Y) coordinates. 2) The vector (A) is formed by subtracting the coordinates of Polaris from UMI \( \lambda \). 3) The vector (B) is formed by subtracting the coordinates of Polaris from the celestial pole. 4) The vector (C) is calculated as \((A/B)\). 5) The pixel location of the celestial pole in the 2\textsuperscript{nd} picture is found by: forming the vector (D) by subtracting the pixel location of UMI \( \lambda \) from Polaris, then multiplying (D) by (C), and adding the pixel location of Polaris.

With a high quality mount – (with no wobble in the RA axis, and no movement in altitude or azimuth with time), a single correction is all that is needed. A second test can be done to verify correct alignment.

Typical time to do the alignment is under 15 minutes.
Test Examples

My mount was first polar aligned using my Losmandy Polar scope. This scope has noticeable offset of the reticle. If you point the center of the scope axis to Polaris – and rotate the scope – you see about 1/10 degree movement of the center.

The first step is to take pictures 1 and 2. Then transfer them to a PC via the STV remote program.

Then a matlab routine – find_polar – is run under MATLAB to determine the polar error. For the pics used in Figure 3 and Figure 4, the resulting plot is shown in Figure 7.

![Figure 7. Output of find_polar.m. This shows the mount is 0.16 degrees from the pole. To get the mount aligned - you would move the altitude and azimuth controls to move Polaris to the pixel location (192, 123).](image)

This plot shows a target location for Polaris of (192,123). With the STV in the eFinder mode, you select “set xhair” and set it to (192,123). Then use the mount’s Altitude/Azimuth controls to move Polaris to the crosshair.

After making the azimuth and altitude adjustments, a second set of 2 pictures were taken, and find_polar was rerun. The results are shown in Figure 8.
Figure 8. Output result after 1 adjustment. Total error from the pole shows 0.004 degree.

The pictures for Figure 8 were taken at 10:00PM. A separate set of pictures were taken at 11:00PM at the end of testing. These results are shown in Figure 9 and show virtually identical results.
Figure 9. Output of find_polar at the end of the night - showing virtually no change in the mount's position.

**Verification of polar alignment using the drift method**

In order to verify the mount was really aligned, an electronic drift test was done. The STV was set up to take a mosaic of 40 sub-pictures of 40x40 pixels each. The target was a star in the southern direction – near the celestial equator. The peak location for each star was determined, and the DEC (Y) pixel location vs time was plotted. Then a straight line was fit to the data.

Figure 10 shows the results of the drift test. The red dots are the Y (in the Declination direction) peak locations, the blue line is a best fit straight line.
Figure 10. Drift vs time.

The plot is in units of pixels. Each pixel is 7.4 microns and the FL of the lens is 75 mm. This results in 20 arc-sec per pixel. The drift is approximately 0.15 pixels in 2500 seconds. Or 3 arc-sec over a 40 min period. This shows the mount was well polar aligned (in azimuth at least).
Further development – adoption by popular CCD guiding programs

The method presented – using an STV, PC and Matlab to do an automated polar alignment – is not very practical for most people. Few people have an STV. Matlab is expensive (although there is a free alternative – OCTAVE – but only for Linux), and most people with an imager will want a longer focal length (FL) for guiding.

I present this method in the hope that the more popular CCD guiding programs (e.g. PHD, MAXIM DL, etc.) will adopt it, and add a quick polar aligning routine similar to what I’ve described here.

This method shows the target stars as Polaris and UMI $\lambda$, which requires about a 2.5 degree FOV – and with a 3.5mm by 4.8 mm imager - a 75 mm lens. Also these 2 stars are in my (free) star atlas – so I can look up current epoch coordinates. However, any 2 stars near the celestial pole can be used. There are several that would allow larger focal lengths. A focal length of 250 mm is feasible, and that provides adequate FL for autoguiding.

What I can imagine is a polar alignment mode in the popular CCD autoguiding programs that would:

1) Use the focal length of the scope and the imager chip size to automatically determine which 2 stars to use.
2) The user would be prompted to orient the mount for the 1st picture. Then push continue.
3) The user would be prompted to orient the mount for the 2nd picture. Then push continue.
4) The program would locate the 2 stars in the picture – make the calculations shown in this paper, and show target location for the 2 stars
5) The user would then make the azimuth/altitude adjustments.

If this was built into a CCD autoguiding package - I envision this total process to take less than 3 minutes.
APPENDIX A – MATLAB source code

Note: Most of the software shown in find_polar.m is associated with plotting – and printing the results.

find_polar.m

% find_polar.m    determine polar alignment

FL = 0.075;  % focal length of camera (m)
pixel = 14.8e-6;  % size of pixel (m)

if ~exist('BASE')
    BASE='1';
end

% read in data ;
PATH = 'c:\matlab\matstv_2012_07_29\';
FILE1=[PATH,BASE,'001.stv'];
FILE2=[PATH,BASE,'002.stv'];
[head1,imag1,im_size,Height,Width]=read_stv(FILE1);
[head2,imag2,im_size,Height,Width]=read_stv(FILE2);

% find  peak star location and value
nstars=3;  % find top 2 (3) peak star locations on both images
[peak1,x1,y1]=peak_star_loc(imag1,nstars);
[peak2,x2,y2]=peak_star_loc(imag2,nstars);

% now do math to find center of rotation between 2 images
% and pole location. Then offset between center of rot and pole loc.
% Then add offset to each star loc in 2nd image - as target to move
% mount in az, el to get stars at new loc.
xin = [x1 , x2]
yin = [y1 , y2]
imag3=zeros(320,320);
clf
image(imag3)
hold on
axis square
mpp=colormap;
mpp(1,:) = [.1, .1 .1 ];
colormap(mpp);

[ delta_x,delta_y,new_x,new_y, pole_x,pole_y, cent_x,cent_y, angle_pic2_re_pic1 ] = find_xy_offset( xin,yin );
image(imag2/2);
% make axis
ax=[ 0 320];
ay=[pole_y pole_y];
bx =[-pole_x pole_x];
by = [0 320];
plot(ax,ay,'k')
plot(bx,by,'k')

% black cross  for center of rotation
cx=[ cent_x-15 ;cent_x+15];
cy=[ cent_y ;cent_y];
dx =[- cent_x; cent_x];
dy =[-cent_y-15; cent_y+15];
plot(cx,cy,'k')
plot(dx,dy,'k')

% cross hair for where stars should go
```matlab
ex = [new_x(1) - 10; new_x(1) + 10];
ey = [new_y(1); new_y(1)];
fx = [new_x(1); new_x(1)];
fy = [new_y(1) - 10; new_y(1) + 10];
plot(ex, ey, 'k')
plot(fx, fy, 'k')

gx = [new_x(2) - 10; new_x(2) + 10];
gy = [new_y(2); new_y(2)];
hx = [new_x(2); new_x(2)];
hy = [new_y(2) - 10; new_y(2) + 10];
plot(gx, gy, 'k')
plot(hx, hy, 'k')

% cross hair for where stars are
kx = [x2(1) - 10; x2(1) + 10];
ky = [y2(1); y2(1)];
lx = [x2(1); x2(1)];
ly = [y2(1) - 10; y2(1) + 10];
plot(kx, ky, 'k')
plot(lx, ly, 'k')

% cross hair for where stars are
mx = [x2(2) - 10; x2(2) + 10];
my = [y2(2); y2(2)];
xn = [x2(2); x2(2)];
ny = [y2(2) - 10; y2(2) + 10];
plot(mx, my, 'k')
plot(nx, ny, 'k')

% one degree circle around pole
n1 = 2*pi*0.360/360;
pix_per_deg = 1/(pixel/FL*180/pi);
x1deg = pix_per_deg*cos(n1) + pole_x;
y1deg = pix_per_deg*sin(n1) + pole_y;
plot(x1deg, y1deg, 'k')
plot(0.1*pix_per_deg*cos(n1) + cent_x, 0.1*pix_per_deg*sin(n1) + cent_y, 'k')

new_starm2 = sprintf('Pole location (X,Y):
%8.1f,%8.1f', pole_x, pole_y);
ss = text(10, 220, new_starm2);
set(ss, 'Color', [1 1 1]);

new_starm3 = sprintf('Scope axis center (X,Y):
%8.1f,%8.1f', cent_x, cent_y);
ss = text(10, 207, new_starm3);
set(ss, 'Color', [1 1 1]);

new_star0 = sprintf('Amount to move (X,Y):');
new_star1 = sprintf('%8.1f,%8.1f', delta_x, delta_y);
ss0 = text(10, 235, new_star0);
ssm1 = text(220, 235, new_star1);
set(ss0, 'Color', [1 1 1]);
set(ssm1, 'Color', [1 1 1]);

new_star1 = sprintf('Move Star 1 (polaris) to (Normal):');
new_star2 = sprintf('Move Star 2 (UMI Lamda) to (Normal):');
ssl = text(10, 252, new_star1);
```
function [head, imag, im_size, Height, Width] = read_stv(FILE);
% [head, imag, im_size, Height, Width] = read_stv(FILE);
% e.g.
% PATH = 'c:\matlab_m\stv\';
% FILENAME='f002.stv';
% FILE=[PATH,FILENAME];
% reads stv data file
% head = 2048 ascii file header
% imag = imaged data - likely compressed
% im_size = size of image data;
% Height = image size
% Width = image size
% PATH = 'c:\matlab_m\stv\';
% FILENAME='f002.stv';
% FILE=[PATH,FILENAME]

fid = fopen(FILE);
[c,n]=fread(fid, 'uchar');

im=c(2049:end);

im_size=n-2048;
head=char(c(1:2048)');

% find image size in header
k = findstr('Height', head);
Height=str2num(head(k+9:k+9+4));

k = findstr('Width', head);
Width=str2num(head(k+8:k+8+4));

% decode compressed image
% SBIG format 1st 2 bytes are line size in bytes (LSByte, MSByte), each line is then:
% next 2 bytes are 1st value, then after that if next is not 128,
% then each additional byte is delta from previous sample(8 bit signed). if = 128, then
% next 2 byte are next sample (LSB, MSB);

N=Height*Width;
imag=zeros(Height,Width);

k=0;

for h=1:Height
k=k+1;
line_size=im(k)+256*im(k+1);
k=k+2;
imag(h,1)=im(k)+256*im(k+1);
k=k+1;
for w=2:Width
k=k+1; % next value
next = im(k);
if(next)<128 % h, w, k
 pause
 k=k+1;
 imag(h,w)=im(k)+256*im(k+1);
k=k+1;
else
 if(next > 128)
 next = next = 256;
 end
 imag(h,w)=imag(h,w-1)+ next;
 end
end
end
fclose(fid);
return;
function [peak,x,y]=peak_star_loc(imag1,nstars);
% [peak,x,y]=peak_star_loc(imag1,nstars);
% eg
% nstars = 3;
% [peak,x,y]=peak_star_loc(imag1,nstars);
% find nstars peak locations in imag
wx=4;
wy=4;
imag=imag1;
for n=1:nstars
[max_value,max_y]=max(imag);
[max_total,max_x]=max(max_value);
yy=max_y(max_x);
xx=max_x;

%y(n)=yy;
%x(n)=xx;
if xx< wx
xx=wx+1;
end
if yy < wy;
yy=wy+1;
end

back = median(imag(:));
peaks=sum( imag(yy-wy:yy+wy,xx-wx:xx+wx) - back );
peak(n)=sum(peaks);
% find centroid
xpeaks=sum( imag(yy-wy:yy+wy,xx-wx:xx+wx) - back );
xww=xx-wx:xx+wx;
x_centroid = sum(xww.*xpeaks)/sum(xpeaks);
x(n)=x_centroid;
ypeaks=sum( imag(yy-wy:yy+wy,xx-wx:xx+wx)' - back );
yww=yy-wy:yy+wy;
y_centroid = sum(yww.*ypeaks)/sum(ypeaks);
y(n)=y_centroid;
% [n xx x_centroid yy y y_centroid]
% replace peak to find next peak
imag(yy-wy:yy+wy,xx-wx:xx+wx) = back ; % replace peak with back
end
end
function [ delta_x,delta_y,new_x,new_y, pole_x,pole_y, cent_x,cent_y,angle_pic2_re_pic1 ] = find_xy_offset( xin,yin );

% Polar alignment GEM:
% Take 2 images near the pole with a rotation between them. Find the location
% of 2 stars (UMI alpha, lambda). From this information, the
% center of rotation is computed, and the x, y offset to the pole
% determined. Then move the mount (via az,el only) to move the stars
% on the 2nd image to the new location.
% Then the pole will be moved to the center of rotation.

% delta_x, delta_y = the offset from the center of rotation to the pole
% image 2
% nex_x, new_y = new positions to put stars in 2nd image to move pole
% to center of rotation.

% xin, yin = positions of the 3 stars (UMI alpha and lambda,
% TYC 4627-259-1 = 2nd star is a check) by 2 pictures (2x3)
% angle_pic2_re_pic1 = angle or rotation (deg) between the 2 pics

% 1) UMI alpha - polaris 2012-05-30 Visual magnitude: 2.0
% Apparent RA (epoch of date): 02h 45m 50.66s
% Apparent Dec (epoch of date): +89° 18' 53.6"
%
% 2) UMI lambda Visual magnitude: 6.34
% Apparent RA (epoch of date): 17h 02m 30.09s
% Apparent Dec (epoch of date): +89° 01' 21.8"
%
% 3) Tycho catalog number: TYC 4627-259-1 Visual magnitude: 6.47
% Apparent RA (epoch of date): 01h 40m 52.67s
% Apparent Dec (epoch of date): +89° 4' 31.7"
%
% stars are reference stars near pole. We send in 3 stars but only need 2.
% 3rd star is to make sure we dont get
% wrong one, as 2nd and 3rd stars are near the same mag.

stars = [ 02+ 45/60+ 50.66/3600, 89+ 18/60+ 53.6/3600, 2.00 ;
17+ 02/60+ 30.09/3600, 89+ 01/60+ 21.8/3600, 6.34 ;
01+ 40/60+ 52.67/3600, 89+ 04/60+ 31.7/3600, 6.47 ;
...]

[nstars, szc] = size(stars) ; % number of stars
x=xin; y=-yin; % y is from image locations - where increasing y is in -y direction.

% convert reference stars to x,y (scaled in degrees)

dist_from_pole = 90-stars(:,2); % in degrees
ang = 2*pi*(-(stars(:,1))/24); % convert RA to radians 0
x_stars = dist_from_pole.*cos(ang);
y_stars = dist_from_pole.*sin(ang); % because y starts at top left in picture coord
mag_stars = stars(:,3);

% use complex math - for vector x +jy
star_v = x_stars +j*y_stars;

% create vector from star 2 to star 1
sa=star_v(1); sb=star_v(2); sc=star_v(3);
sba=sb-sa;
scor= -sa/sba;
% check star positions on the images - error checks
%max mag will be polaris - check to make sure 2nd star is further away
%than 1st. also check that 1st 2 stars are same dist in 1st and 2nd images

imaged_stars=x +j*y;  % 2 images - colm, by 3 stars)
imaged_sba=imaged_stars(:,2)-imaged_stars(:,1);
imaged_sca=imaged_stars(:,3)-imaged_stars(:,1);
imaged_sab2=imaged_stars(2,1);
imaged_sb2=imaged_stars(2,2);
imaged_sab1=imaged_stars(1,1);

% make sure sab is bigger than sac
%if   (abs(imaged_sba(1)) <  abs(imaged_sca(1)))
%  error('3rd star picked in error - larger than 2nd star - 1st image')
%end
%if    (abs(imaged_sba(2)) <  abs(imaged_sca(2)))
%  error('3rd star picked in error - larger than 2nd star - 2nd image')
%end

% make sure sab is same on boh pictures
if( abs(abs(imaged_sba(1)) / abs(imaged_sba(2)) -1) > 0.1 )
  error('distance between 1st and 2nd star differs by more than 10%')
end

% passes error checks now on image 2, find pole location

imaged_pole2 = imaged_sab2 + imaged_sba(2)*scor;  % this is location of pole on image2
imaged_pole1 = imaged_sab1 + imaged_sba(1)*scor;

% find center of rotation using 2 images
pole_x= real(imaged_pole2);
pole_y= - imag(imaged_pole2);

cax= ( x(1,1)+x(2,1) )/2;
cay= ( y(1,1)+y(2,1) )/2;
cbx= ( x(1,2)+x(2,2) )/2;
cby= ( y(1,2)+y(2,2) )/2;
mra= ( x(1,1)-x(2,1) ) / ( y(2,1)-y(1,1) );
mrb= ( x(1,2)-x(2,2) ) / ( y(2,2)-y(1,2) );
bra = cay - mra*cax;
brb = cby - mrb*cbx;
x_center=(brb - bra)/(mra - mrb);
y_center = mra*(x_center) +bra;

cent_x=x_center;
cent_y= y_center;

center=x_center + j*y_center;

loc_error = center - imaged_pole2;

new_image_a2=imaged_sab2 + loc_error;  % target to move stars to
new_image_b2=imaged_sb2 + loc_error;

new_x = [real(new_image_a2), real(new_image_b2)];
new_y = [imag(new_image_a2), imag(new_image_b2)];

delta_x = real(loc_error);
delta_y = -imag(loc_error);
angle_pic2_re_pic1=180/pi*{angle(imaged_sba(2)/imaged_sba(1))};
end