ACTIVE OPTICAL SYSTEMS

AN041

Theoretical Centroid Accuracy

Author: Dr. Justin Mansell
Revision: 8/5/14

1 Introduction

The fundamental numerical image processing technique used in most Shack-Hartmann wavefront sensors to extract wavefront slope information from the image is centroiding or a first moment calculation. This application note explores the fundamental theoretical accuracy of this approach varying noise, pixels per sub-aperture, and bits of digitization. The goal is to establish an average conservative estimate of centroid accuracy over a variety of realistic conditions.

2 Study 1: Basic Centroid Accuracy

The script in the appendix performs centroiding on a single sub-aperture with a Gaussian spot. The $1/e^2$ Gaussian radius was set to one quarter of the sub-aperture diameter for each case. The spot was scanned to 100 positions over a one pixel offset. The absolute value of varying amplitude Gauss-random noise was added to each intensity profile with an amplitude factor equal to the reciprocal of a SNR factor. The intensity was normalized after noise addition then digitized to varying number of bits per pixel. A 10% threshold was used for each centroid calculation. The RMS of the difference between the input spot position and the centroid estimation of spot position was calculated over all the cases of offset and noise to determine the estimation error for a given condition (number of pixels across a sub-aperture, noise amplitude, and bits of digitation).

2.1 Study 1: Results & Conclusions

Figure 1 shows the results of centroid accuracy for different input cases. In the lab, it is fairly easy to achieve >4 pixels across a sub-aperture, SNR>20, and at least 8-bits of digitization. These cases achieve a centroid accuracy of $<1/100^{th}$ of a pixel. For a high speed field wavefront sensor, $1/4^{th}$ of a pixel accuracy is more realistic.

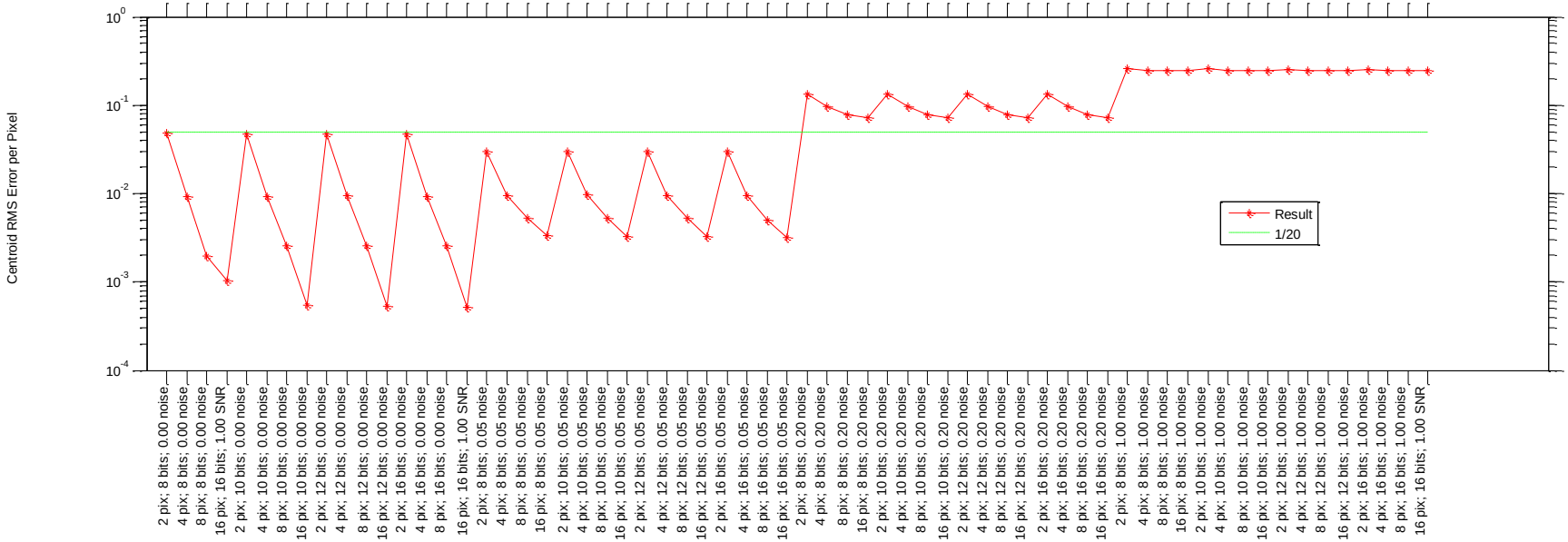
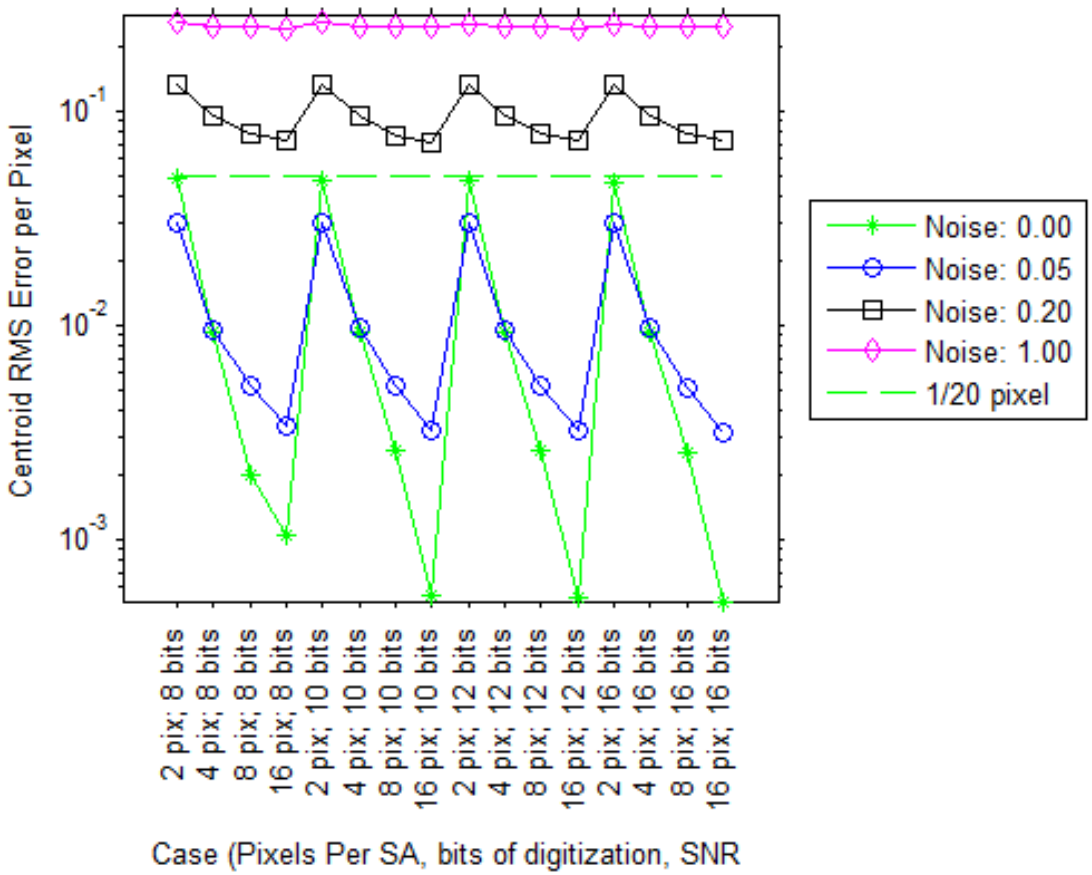
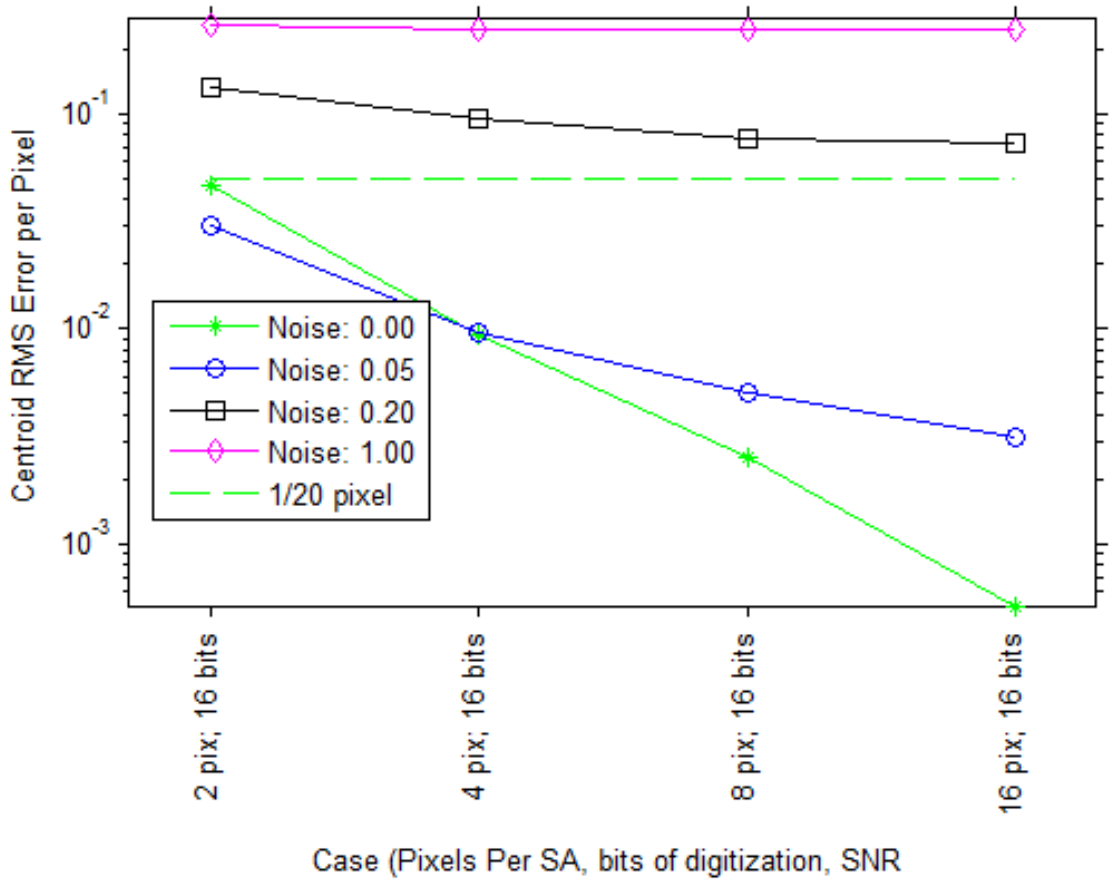


Figure 1 - Results of Centroid Accuracy for Various Input Cases

2.2 Other Plots: Study 1





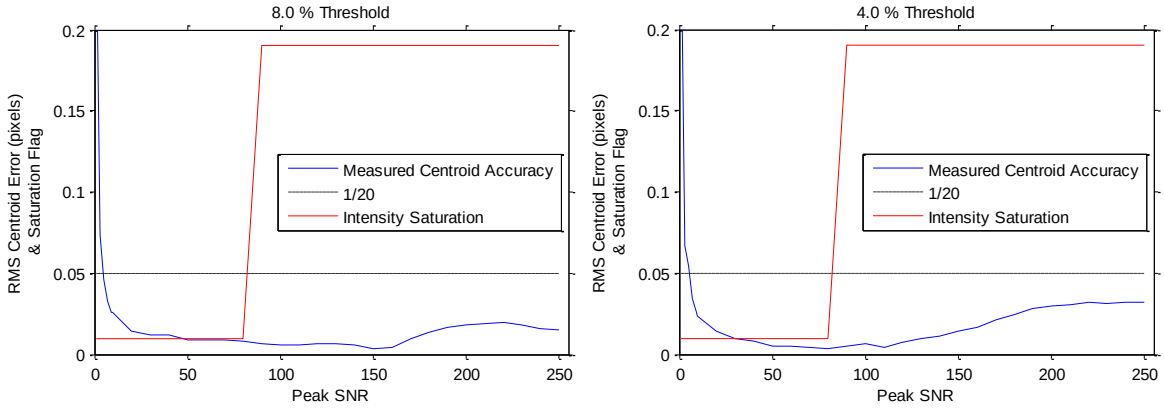
3 Study 2: Centroid Accuracy vs. SNR

In a follow-up study, the centroid accuracy was calculated by computing the RMS of the difference between the ideal centroid locations (input) from the measured centroids in the presence of a uniformly distributed noise. The noise was fixed at 0-3 counts. The maximum signal was 8 bit (255 counts). The sinc distribution was scaled by multiplying the maximum noise (3 counts) by the SNR, which was varied throughout the study. The intensity was allowed to saturate at 255 counts to see the effect of intensity saturation. The intensity was discreteized to 8 bits. It is important to note that for this study, SNR is calculated on peak value (peak signal to peak noise) not on RMS. The intensity profile was pixelated into a 16x16 sample grid.

The image was initially thresholded at 3 counts by subtracting 3 counts from the image and then zeroing any pixel less than zero. For centroiding, an additional relative threshold was computed by multiplying the peak intensity in the sub-aperture by a percentage (typically between 4 and 10%), subtracting this value from the image, and then zeroing any intensity less than zero. If any frame at a given SNR in the study had one pixel at 255 counts, the saturation flag was set.

3.1 Study 2: Results & Conclusions

The following charts show the results of the study for 8% and 4% relative threshold. We see that good centroid accuracy (1/20th of a pixel) was achieved by a SNR of ~5 (with a maximum of 3 counts of noise and a signal maximum of 15 counts) and maintained deep into the saturation region. This study shows how robust the thresholded centroid algorithm is to intensity variations including into saturation and even with only 8 digitization bits.



4 Appendix: Matlab Code

The code below contains some dependent functions that will not work on all systems, but should be easy to remove to make this script work on all Matlab-enabled systems.

```
% centroid accuracy testing
close all; clear all; clc; format compact;
ppt=0;
dbg = 0;
dbg2 = 0;
```

```
seed = -1234567;
Nreal = 20;
randn('seed',seed);

%noiseVec = 1./[1e10 20 10 5 2 1];
%bitsVec=6:2:16;
noiseVec = 1./[1e10 20 5 1];
NpixVec = [2 4 8 16];
bitsVec=[8 10 12 16];
dpix = 10e-6;
thr = 0.1;
dxy = dpix/10;

c1 = 0;
totalcount = length(noiseVec)*length(bitsVec)*length(NpixVec);
for noise = noiseVec;
    for bits=bitsVec;
        Imax = 2^bits;
        for Npix = NpixVec;
            c1=c1+1;
            disp(sprintf('%i / %i',c1,totalcount));
            w = dpix*Npix/4;

            nxy = round((Npix)*dpix/dxy);
            xv = (1:nxy)*dxy;
            [xx,yy]=meshgrid(xv,xv);
            dxcVec = (-dpix/2:dxy:dpix/2);
            dycVec = (-dpix/2:dxy:dpix/2);
            [dxx,dyy]=meshgrid(dxcVec,dycVec);
            xc0 = (max(xv)-min(xv))/2 + min(xv);
            yc0 = xc0;

            [xxsa,yysa]=meshgrid(1:Npix,1:Npix);
            Isa = zeros(Npix,Npix);
            c2=0;
            for dxc = dxcVec;
                for dyc = dycVec;
                    xc = xc0 + dxc;  yc = yc0 + dyc;

                    %generate image
                    I0 = exp(-1 * ( (xx-xc).^2 + (yy-yc).^2 ) / (w.^2));
                    %I = I + noise * randn(size(I));
                    for realization = 1:Nreal;
                        c2=c2+1;
                        I = I0 + abs(noise * randn(size(I0)));

                        %generate sub-ap pixelation
                        for ii=1:Npix;
                            iiv = (1:10)+(ii-1)*10;
                            for jj=1:Npix;
                                jjv = (1:10)+(jj-1)*10;
                                Isa(ii,jj) = sum(sum(I(iiv,jjv)));
                            end;
                        end;
                        Isa = Isa ./ max(max(Isa));
                        Isa = round(Isa*Imax)./Imax;
```

```
                        Ithr = Isa - thr;
                        Ithr = Ithr .* (Ithr>0);

                        %calc centroid and centroid error
                        xm(c2) = dpix * sum(sum(Ithr .* xxsa)) ./
sum(sum(Ithr));
                        ym(c2) = dpix * sum(sum(Ithr .* yysa)) ./
sum(sum(Ithr));
                        xmIdeal(c2) = xc0 + dxc + dpix/2 - dxy/2;
                        ymIdeal(c2) = yc0 + dyc + dpix/2 - dxy/2;
                        xmErr(c2) = xm(c2) - xmIdeal(c2);
                        ymErr(c2) = ym(c2) - ymIdeal(c2);
                    end;

                    if (dbg || (ppt && dxc==0 && dyc==0))
                        clf;
                        subplot(1,2,1); imagesc(I); axis image; caxis([0 1]);
colorbar;
                        subplot(1,2,2); imagesc(Ithr); axis image; caxis([0
1]); colorbar;
                        drawnow;
                        if (ppt)
                            ToPPT(sprintf('Bits=%i Npix=%i',bits,Npix));
                        end;
                    end;

                end;
            end;

            dxxv = dxx(:);
            dyyv = dyy(:);
            if (dbg2)
                clf;
                subplot(1,2,2); plot(xmIdeal,xmErr,'k*');
                subplot(1,2,1); plot(xmIdeal,xm,'k*');
                hold on;
                LinearFit(gcf,xmIdeal,xm,12,'r-');
                drawnow;
                pause;
            end;

            xerr(c1) = RMS(xmErr);
            yerr(c1) = RMS(ymErr);%RMS(ym(:) - dyyv) - yc0 - dpix/2);

            clf; semilogy(xerr./dpix,'r*-');
            hold on;
            semilogy(xerr.*0+1/20,'g--');
            description{c1} = sprintf('%i pix; %i bits; %.2f
SNR',Npix,bits,noise);
            title(description{c1});
            legend('Result','1/20','Location','Best');
            xlabel('Case (Pixels Per SA, bits of digitization, SNR');
            ylabel('Centroid RMS Error per Pixel');
            if (length(xerr)>2)
                xticklabel_rotate(1:length(xerr),90,description);
            end;
```

```
            drawnow;
        end;
    end;
end;

%% final plot
%nf; semilogy([NpixVec],xerr./dpix,'b*-');
nf; semilogy(xerr./dpix,'b*-');
hold on;
semilogy(xerr.*0+1/20,'g--');
legend('Result','1/20','Location','Best');
xlabel('Case (Pixels Per SA, bits of digitization, SNR');
ylabel('Centroid RMS Error per Pixel');
if (ppt)
    ToPPT('Summary');
end;
```

4.1 Script 2: Centroid Accuracy vs. SNR

```
%routine for centroid error vs SNR
Setup;
ppt=0;

N1D = 16;
wavelength = 795e-9;
f=6.7e-3;
dsub=150e-6;
nxy = 256;
thrrel = 0.04;
rx = f*wavelength/dsub
ry = f*wavelength/dsub;

dxy = dsub/nxy;
g = makeGrid(nxy,dxy);
xx=g.xx; yy=g.yy;

Nsamp = nxy/N1D;
dpix = dxy*Nsamp;

for ii=1:nxy;
    for jj=1:nxy;
        idxX(ii,jj) = double(int32(ii/Nsamp)+1);
        idxY(ii,jj) = double(int32(jj/Nsamp)+1);
    end;
end;
idx = idxX + (idxY-1)*max(idxX(:));
idxXc = idxX(:);
idxYc = idxY(:);
idxc = idx(:);
idxMax = max(idx(:));
for ii=1:idxMax;
    xi(ii) = max(idxXc.*(idxc==ii));
    yi(ii) = max(idxYc.*(idxc==ii));
end;
```

```
nf; show(idx'); title('index');

nf;
dy = 0;
rspot = (rx+ry)/2
pixDynRange = (dsub/2 - (rspot))/dpix
dxvec = (0:0.1:pixDynRange).*dpix;
c=0;
noiseVal = 3;
maxIntensityValue = 255;
SF = 3;
SNRmaxval = (5*round(maxIntensityValue/noiseVal/5*SF))
SNRvec = [1:2:10 10:10:SNRmaxval];
for SNR = SNRvec;
    c=c+1;
    clear xc; clear yc;
    cc = 0;
    saturated(c)=0;
    for dx = dxvec;
        cc=cc+1;
        %Create Image
        I = (sinc(pi*(xx-dx)./(rx)).*sinc(pi*(yy-dy)./(ry))).^2;
        %nf; show(g.x,g.y,I);

        %downsample image
        for ii=1:idxMax;
            Isamp = sum(sum(I.*(idx==ii)));
            Is(xi(ii),yi(ii)) = Isamp;
        end;
        %normalize
        Is = Is - min(Is(:)); Is = Is ./ max(Is(:));
        %record true center on first run
        if (dx==0)
            [xc0,yc0]=MomentsWithThreshold(Is,thrrel);
        end;
        Is = Is * noiseVal * SNR;
        %add noise
        noise = noiseVal.* rand(size(Is,1),size(Is,2));
%         RMSnoise = RMS(noise(:))
%         RMSsignal = RMS(Is(:))
%         SNRinst = (RMSsignal./RMSnoise).^2
%         SNRsave(c,cc)=SNRinst;
        Is = Is + noise;
        Is = Is .* (Is>0);
        %add image saturation
        satVals = (Is>maxIntensityValue);
        Is = Is .* (1-satVals) + maxIntensityValue.* satVals;
        Is = round(Is);
        Is = Is ./ maxIntensityValue; %renormalize
        if (max(Is(:))==1)
            saturated(c)=1;
        end;

        %implement a global 3 count threshold
        Is = Is - 3/maxIntensityValue;
        Is = Is .* (Is>0);
```

```
        %find centroid with relative and global threshold
        [xc(cc),yc(cc)]=MomentsWithThreshold(Is,thrrel);

        %show plot
        if (dx==max(dxvec)) %to speed things up...
            clf;
            subplot(2,2,1); show(Is.*maxIntensityValue);  hold on;
plot(xc,yc,'b*','MarkerSize',5); clabel('Counts');
            subplot(2,2,2); show(I);  hold on; title('Raw Intensity');
            subplot(2,2,3); plot((xc-xc0).*dpix); hold on; plot(dxvec,'k--');
xlabel('Position'); ylabel('Centroid');
            subplot(2,2,4); plot((xc-xc0).*dpix-dxvec(1:length(xc)),'k--');
xlabel('Position'); ylabel('Error');
            title(sprintf('SNR = %.2f',SNR));
            drawnow;
        end;
    end;
    errvals = ((xc-xc0).*dpix)-dxvec;
    err(c) = RMS(errvals);
end;
%% final summary
% nf; plot(mean(SNRsave'));
% hold on;
% plot(SNRvec,'k--');

nf;
subplot(1,2,1);
plot(SNRvec,err/dpix); hold on; plot(SNRvec,1/20+SNRvec*0,'k--');
xlabel('SNR'); ylabel('RMS Centroid Error (pixels)');
ax=axis; ax(1)=0;ax(2)=300; ax(3)=0; ax(4)=0.2;axis(ax);
legend('Measured Centroid Accuracy','1/20','Location','Best');
subplot(1,2,2);
plot(SNRvec,saturated); xlabel('SNR'); ylabel('Saturated Flag');
ax=axis; ax(1)=0;ax(2)=SNRmaxval; ax(3)=-0.1; ax(4)=1.1;axis(ax);

nf;
plot(SNRvec,err/dpix); hold on; plot(SNRvec,1/20+SNRvec*0,'k--');
xlabel('SNR'); ylabel('RMS Centroid Error (pixels)');
ax=axis; ax(1)=0;ax(2)=SNRmaxval; ax(3)=0; ax(4)=0.2;axis(ax);
plot(SNRvec,saturated.*ax(4)*0.9+0.05*ax(4),'r-'); xlabel('SNR');
ylabel('Saturated Flag');
legend('Measured Centroid Accuracy','1/20','Intensity
Saturation','Location','Best');
title(sprintf('%.1f %% Threshold',100*thrrel));
```