ACTIVE OPTICAL SYSTEMS

AN043

Centroid Thresholding Technique Analysis

Author: Dr. Justin Mansell
Revision: 11/19/14

1 Introduction

The key image processing technique for Shack-Hartmann wavefront sensing is sub-aperture centroiding. Camera noise and diffraction side-lobes can cause error in the spot estimation, but proper thresholding of the image can minimize the effect. This

2 Technique

There are two common thresholding techniques:

1. Subtract the threshold amount from the image and then zero the values below zero (subtract-and-zero), or
2. Zero the image below the threshold (zero-only).

Each of these two techniques were applied to a focused spot from a 150-micron diameter lens with a 6.7mm focal length at a 532nm wavelength on a pixel grid with ~7 micron pixels. Figure 1 shows an example image of the focal plane on the pixel grid.

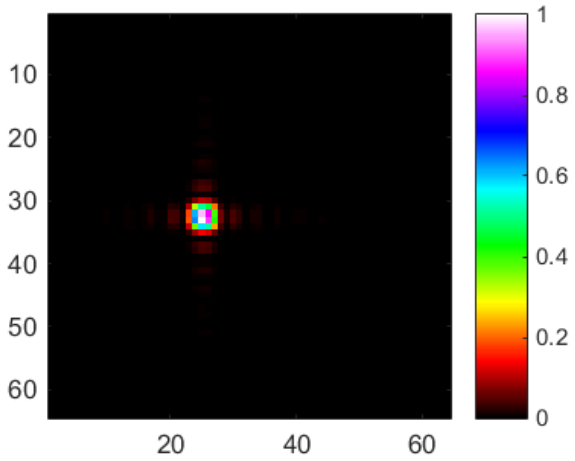

Figure 1: Example Focussed Spot Image

The centroid operation was performed on the pixelated images with each centroid technique. The full resolution image (0.87 micron pixels) was also centroided with no threshold. The full resolution image centroid was subtracted from the pixelated image centroids with each threshold to establish and error. Figure 2 shows the RMS error for each technique for different intensity digitization. The second technique ("Zero Only") shows clearly more error than the subtract-and-zero technique. Our analysis attributes this to the fact that the intensity is no longer on a pedestal in the subtract-and-zero technique which does not bias the results toward the center of the intensity profile.

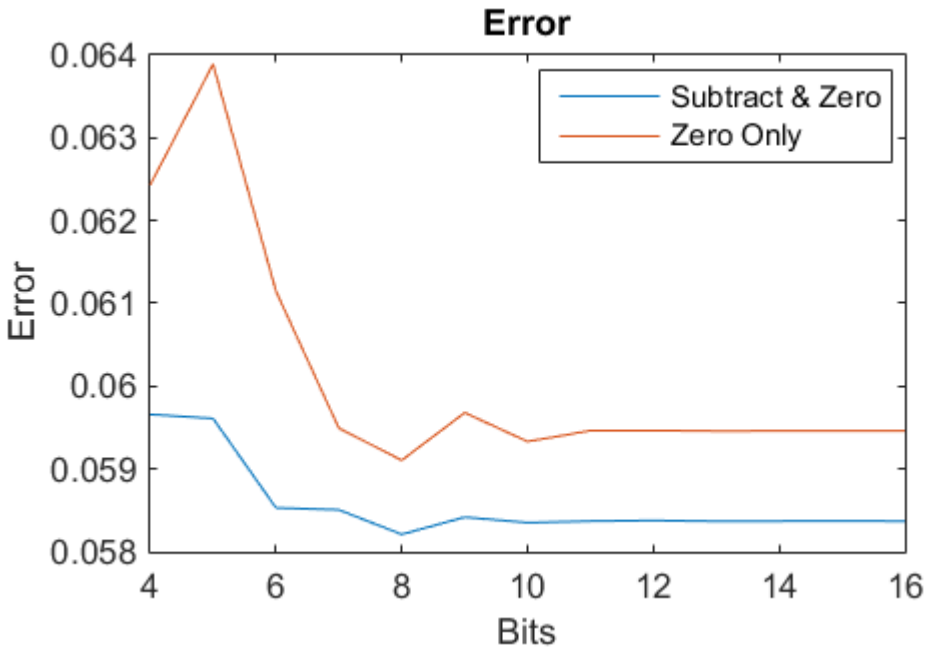

Figure 2: Centroid Error for each of the Threshold Techniques

3 Conclusions

The subtract-and-zero threshold technique is clearly better than the zero-only technique when calculating a centroid for Shack-Hartmann wavefront sensing.

4 Appendix: Matlab Code

```
% Analysis of the effect of large intesity variations on the focal spot
% positon in a Shack-Hartmann wavefront sensor.
Setup;
ppt=0;

wavelength = 0.532e-6;
Dap = 150e-6;
Dpixel = 5.5e-6;
f=6.7e-3;
nxy = 512;
dxy = Dap*3/nxy;
tiltRange = (Dap - 2.0*f*wavelength/Dap)/f/2
thr = 0.05;
xxx=whos;
for ii=1:length(xxx);
    n = xxx(ii).name;
    eval(sprintf('val=%s;',n));
    txt{ii} = sprintf('%s = %.2e',n,val);
end;
if (ppt)
    TitleSlidePowerPoint('Types of Thresholding');
    TextToPowerPoint(txt,'vars',18,20);
    p = mfilename('fullpath')
    TextFileToPPT(sprintf('%s.m',p));
end;

bitsVec = [4:1:16];
ccc=0;
for bits = bitsVec;
    ccc=ccc+1;
    levels = 2.^bits;


    g = makeGrid(nxy,dxy);
    ap = (abs(g.xx)<=Dap/2) .* (abs(g.yy)<=Dap/2);

    nf;
    show(g.x,g.y,ap); title('1');
    if (ppt) ToPPT('Lens Array Aperture'); end;

    leg{1} = 'Subtract & Zero';
    leg{2} = 'Zero Only';
    leg{3} = 'Ideal';

    deltaSFvec = -0.1:0.01:0.1;
    c=0;
    nf([100 100 1000 1000]);
    tiltSFVec = 0:0.01:1.0;
    SF = 2^(ceil(log(Dpixel/dxy)./log(2)));
    for tiltSF = tiltSFVec;
        c=c+1;
        tilt = tiltSF .* tiltRange;
```

```
        E = ap.*QPF(nxy,dxy,f,wavelength).*exp(-
1j*2.0*pi/wavelength*tilt.*g.xx);
        Eff = fresnelShort2D(E,dxy,f,wavelength);
        Iff{c} = abs(Eff).^2;
        Iff{c} = Iff{c}./max(Iff{c}(:));
        Iffv = Iff{c}(:);

        Isub = image_downsample(Iff{c},SF.*[1 1],'mean');
        Isub = Isub ./ max(Isub(:));
        Isub = round(levels .* Isub) ./ levels;

        if (c==1)
            [Nx,Ny]=size(Isub);
            xv = (1:Nx) - 0.5 - Nx/2; yv = (1:Ny) - 0.5 - Ny/2;
            [xx,yy]=meshgrid(xv,yv);
        end;

        I{1} = (Isub - thr).*(Isub>thr); %subtract and zero sub-threshold
intensity
        I{2} = (Isub).*(Isub>thr); %zero sub-threshold intensity
        %I{3} = (Isub>thr); %binary

        for ii=1:2;
            xc(c,ii) = sum(xx(:).*I{ii}(:))./sum(I{ii}(:));
            yc(c,ii) = sum(yy(:).*I{ii}(:))./sum(I{ii}(:));
        end;
        xc(c,3) = sum(g.xx(:).*Iffv)./sum(Iffv) / SF / dxy;
        yc(c,3) = sum(g.yy(:).*Iffv)./sum(Iffv) / SF / dxy;

        clf;
        subplot(2,2,1);
        for ii=1:3;
            plot(xc(:,ii),GetLineSpec(ii));
            hold on;
        end;
        legend(leg,'Location','Best');
        subplot(2,2,2);
        for ii=1:2;
            plot(xc(:,ii)-xc(:,3),GetLineSpec(ii));
            deltavec = xc(:,ii)-xc(:,3);
            hold on;
        end;
        subplot(2,2,3);
        show(Isub);
        drawnow;

    end;

    %% test
    nf;
    for ii=1:2;
        plot(xc(:,ii)-xc(:,3),GetLineSpec(ii));
        deltavec = xc(:,ii)-xc(:,3);
        err(ccc,ii) = RMS(deltavec)
        hold on;
```

```
    end;

    %% summary
    nf;
    plot(tiltSFVec,xc,'b-');
    hold on;
    plot(tiltSFVec,yc,'r--');
    if (ppt) ToPPT(); end;

end;

%% summary
nf;
plot(bitsVec,err);
title('Error');
xlabel('Bits');
ylabel('Error');
legend(leg{1},leg{2},'Location','Best');
if (ppt)
    ToPPT();
end;
```