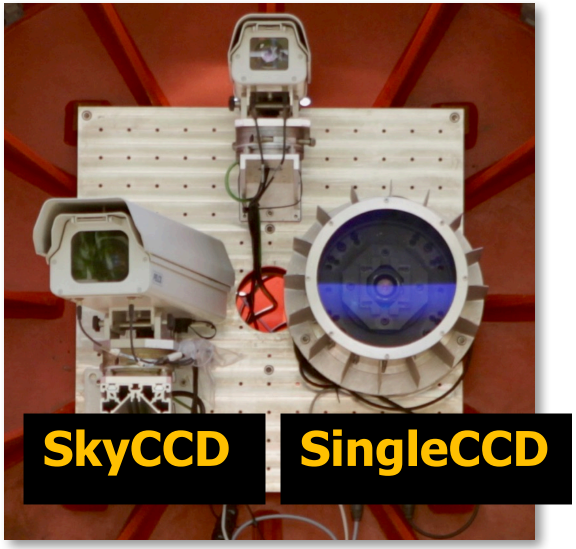
July 11, 2018

A Pointing Model for a Starguider CCD Camera with a Large Offset wrt to the Telescope Optical Axis

Ulli Schwanke, Jan-Lukas Krieg, Gerrit Spengler
(Humboldt University Berlin)

Scope

- The SingleCCD concept is considered as the desired solution for MST pointing
- At present, a SingleCCD is being tested on MAGIC since tests at the MST prototype site in Berlin suffered from limitations (high night-sky background, FoV obscured by telescope focal plane etc.)
- The concept with two CCD cameras (LidCCD and SkyCCD/StarguiderCCD) is being considered as (unlikely?) backup solution
- Tried anyway to derive a pointing model for the SkyCCD within the framework of a Bachelor thesis (Jan-Lukas Krieg)

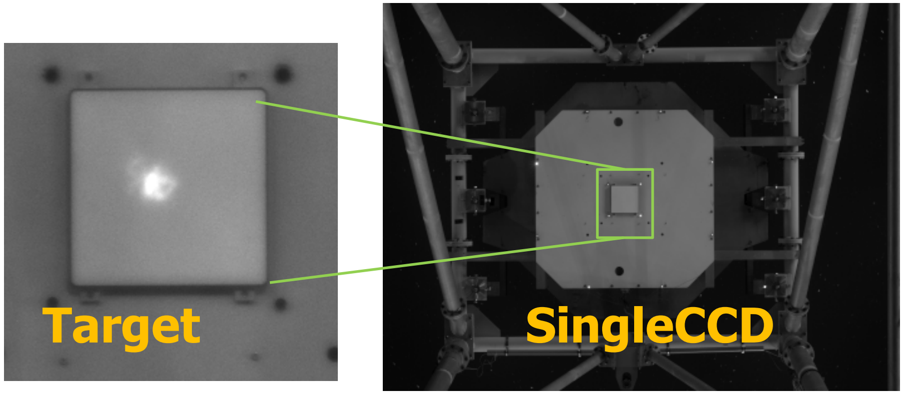


Reminder: MST CCD Cameras

- **SkyCCD is a Prosilica CCD camera**
- **75mm Walimex optics**
- **Surveillance-type housing**

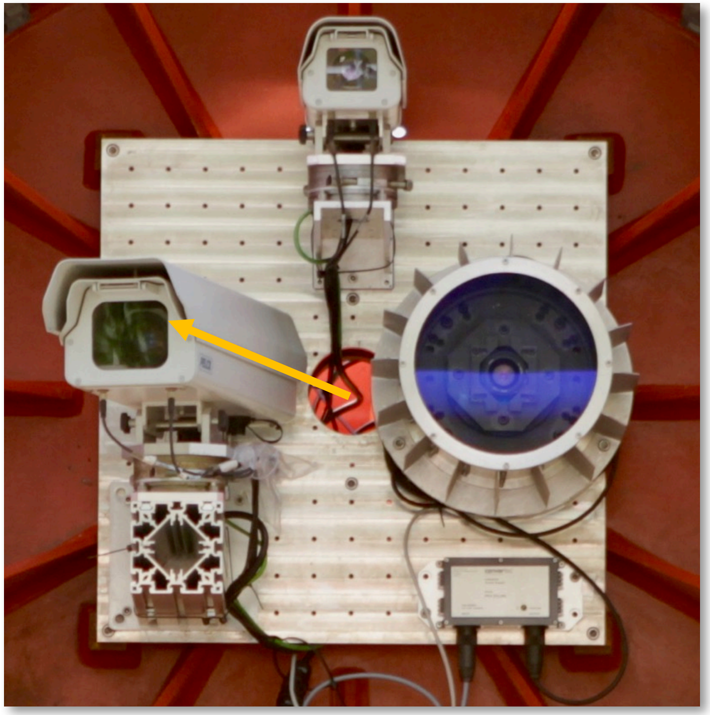
CCD camera	pixel width (μm)	pixel scale ω (arcsec)	pixels	image depth	FoV
SkyCCD	4.65	11.03	1360×1024	8 bit	$4.2° \times 3.1°$
SingleCCD	5.5	21.9	3296×2472	16 bit	$20.5° \times 15.5°$

- **SingleCCD is an Aspen astronomical CCD**
- **Dedicated housing with cooling etc**
- **Also used for mirror alignment, PSF studies etc**



The Problem

- The SkyCCD has (now) a rather large angle (≈10°) with respect to the (approximate) optical axis of the telescope
- Most bending models (i.e. the calculation of a correction in azimuth (Δaz) and elevation (Δel), c.f. slide 1 in backup) assume small deviations from the ideal situation (like the 11-parameter model shown in slide 2 of the backup)



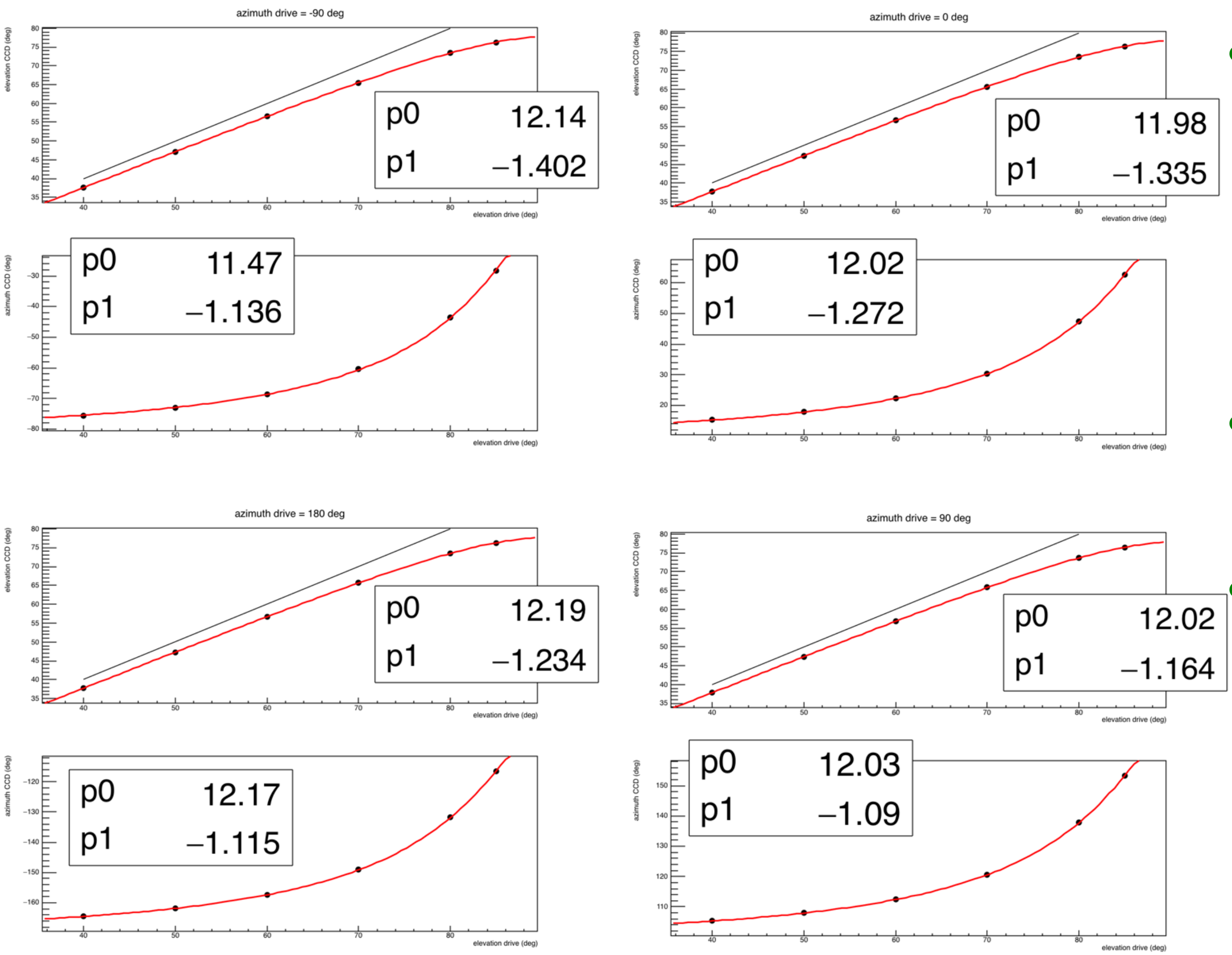
- Had to derive formulas for Δaz and Δel for this situation; found basic analytical solution that works
- Adding higher-order corrections should be simple since these are small

The Rough Solution…

```
const Float_t f = TMath::Pi()/180.0;

Double_t FitEl(Double_t* arg,Double_t* par){
  //the observed elevation is (to first order) only a function of the drive elevation
  const double el = arg[0];  //drive elevation
  const double az0 = par[0]; //mispointing of SkyCCD in azimuth
  const double el0 = par[1]; //mispointing of SkyCCD in elevation
  return asin(sin(el*f)*cos(az0*f)*cos(el0*f)+cos(el*f)*sin(el0*f))/f;
}

double AZ = -99999;

Double_t FitAz(Double_t* arg,Double_t* par){
  //the observed azimuth is both a function of elevation and of azimuth
  double el = arg[0]; //drive elevation
  double phi = AZ;   //drive azimuth (for now a global variable that is set before fitting)
  double phi0 = par[0]; //mispointing of SkyCCD in azimuth
  double el0 = par[1]; //mispointing of SkyCCD in elevation
  double A = -sin(phi*f)*(cos(el*f)*cos(phi0*f)*cos(el0*f)-sin(el*f)*sin(el0*f))-cos(phi*f)*sin(phi0*f)*cos(el0*f);
  double B = +cos(phi*f)*(cos(el*f)*cos(phi0*f)*cos(el0*f)-sin(el*f)*sin(el0*f))-sin(phi*f)*sin(phi0*f)*cos(el0*f);
  return atan2(-A,B)/f; //it's important to use atan2() not atan()
}
```

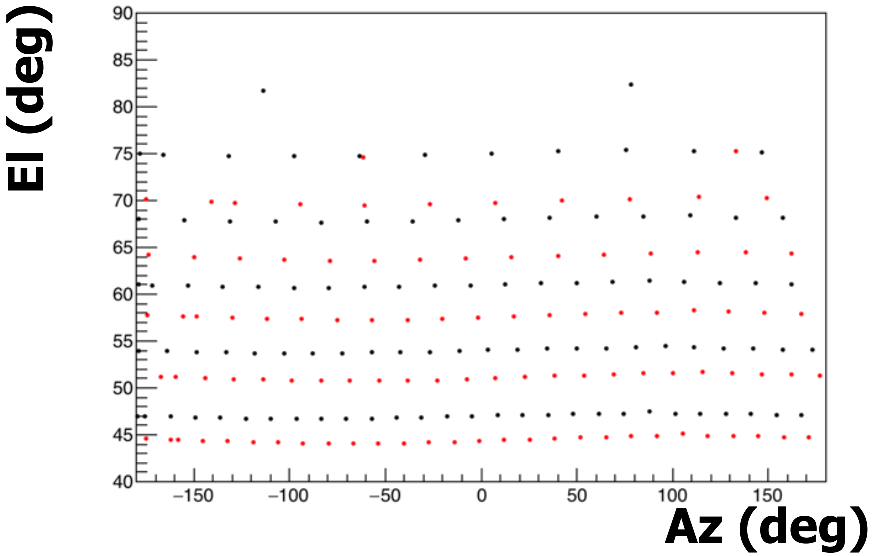
- **Simplest possible fit has 2 parameters: a mispointing angle in azimuth (az0) and a mispointing angle in elevation (el0)**
- **Shows elevation/azimuth and not Δel and Δaz (but this can be changed easily); actually one also needs to fit the drive coordinates as function of the SkyCCD coordinates**
- **Sanity check: El(az0=0,el0=0)=el and Az(az0=0,el0=0)=az**

....Works



- Moved telescope to constant azimuth (-90° (upper left), 0° (upper right) , 90° (lower right) and 180° (lower left) and varied elevations
- Parameters (in the boxes, in deg) are always the same
- Deviations are higher-order corrections (e.g. tilt of azimuth axis along South-North and East-West) and can be included easily

Summary And Next Steps



Black: drive system

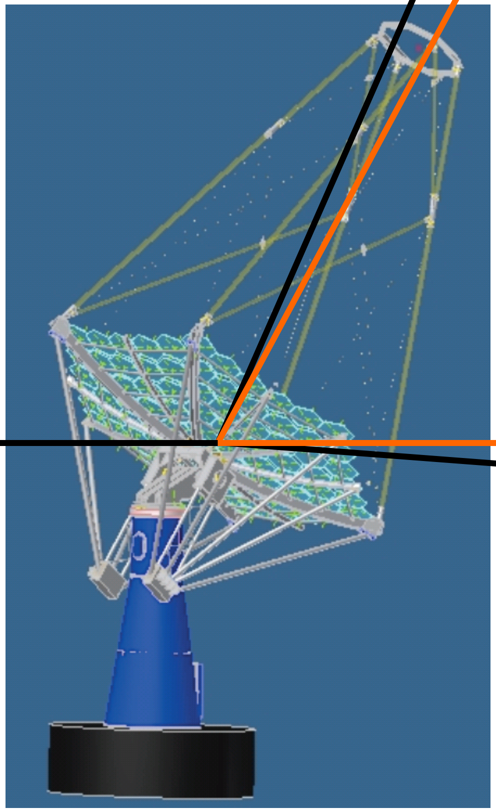
Red: center of the SkyCCD FoV (from astrometry.net)

- **Derived analytical formula and proved that it works for a few dedicated pointings (varied drive elevation at constant drive azimuth)**
- **Investigating higher-order corrections using ~100 Tpoints recorded with the MST prototype in the night July 4/5, 2018**
- **Results will be written up as Bachelor thesis of Jan-Lukas Krieg (in German)**

Backup

Pointing Models

Actually observed elevation (el‘)



Targeted elevation (el)

Actually observed azimuth (az‘)

Targeted azimuth

(az)

Pointing model:

$(el', az') = f(el, az)$ or

$az - az' = \Delta az(el', az')$

$el - el' = \Delta el(el', az')$

11-Parameter Model

```
Double_t deltaEl(Double_t az,Double_t el){
  Double_t ret = par[0] - par[2]*sin(az) + par[3]*cos(az) + par[4]*el;
  return ret;
}
```

0(1) : offset of elevation (azimuth) axis

2(3): tilt of azimuth axis along West-East (North-South)

4: camera sagging

```
Double_t deltaAz(Double_t az,Double_t el){
  Double_t ret = tan(el)*(par[1]+par[2]*cos(az) + par[3]*sin(az));
  ret += par[5]*cos(el);
  ret += par[6]*sin(el);
  ret += par[7]*sin(az);
  ret += par[8]*cos(az);
  ret += par[9]*sin(2*az);
  ret += par[10]*cos(2*az);
  return ret;
}
```

5-10: partly empirical and exaggarated Fourier expansion