Images & turbulence — 26

-> Detection noises:

• At first: *photon noise* (or *shot noise*), poissonian, actually a transformation of the image.

$$p(n) = \frac{N^n e^{-N}}{n!}, \text{with} : N = \frac{L\Delta t}{h\nu}, L = \text{luminosity}, \Delta t = \text{time exp.}$$

p(n) = probability to detect n photons when N are expected

For large N: ~gaussian…

$$p(n) \simeq \exp\left(-\frac{(n-N)^2}{2N}\right)$$

Images & turbulence — 27

-> Detector noises:

• At first: *photon noise* (or *shot noise*), poissonian, actually a transformation of the image.

• At last: *read-out noise* (*RON*), gaussian with zero mean and rms σ_e [e-/px], additive noise.

• In between: *dark current noise*, *amplification noise* & *exotic dark current noise* in the case of EMCCDs, noise due to the *calibration* of the *flat field*, *'salt & pepper' noise* ('hot' and 'cold' pixels), etc.

Images & turbulence — 28

```
;; Photon noise (Poisson)
if keyword_set(PHOT_NOISE) then begin
   idx=where((image GT 0.) AND (image LT 1E8),c)
                                                 ; For values higher than 1E8, should one
   if (c NE 0) then for i=0l,c-1l do $          ; really has to worry about photon noise ?
      noisy_image[idx[i]]=randomn(seed_pn,POISSON=image[idx[i]],/DOUBLE)
endif

;; Additive dark-current noise (Poisson)
if keyword_set(SIGMA_DARK) then begin
   if not(keyword_set(DELTA_T)) then begin
      message, "dark-current noise calculation does need a time exposure value!!"
   endif else noisy_image+=randomn(seed_dark,npx,npy,POISSON=sigma_dark*delta_t,/DOUBLE)
endif

;; EMCCD noises
; Additive exotic (time-exposure-independent) dark-current noise (Poisson)
if keyword_set(EXODARK) then noisy_image+=randomn(seed_xd,npx,npy,POISSON=exodark,/DOUBLE)

; Additive main EMCCD noise (Gamma)
if keyword_set(GAIN_L3CCD) then begin
   idx=where(image GT 0, c)
   if (c NE 0) then for i=0l,c-1l do $
      noisy_image[idx[i]]+=gain_l3ccd*randomn(seed_l3ccd,GAMMA=image[idx[i]],/DOUBLE)
;   noisy_image=long(temporary(noisy_image))
endif

;; Flat-field calibration residuals
if keyword_set(FFOFFSET) then begin
   ffres=randomn(seed_ff,npx,npy)*ffoffset+1.
   idx = where(ffres LE 0., c)
   if (c NE 0) then ffres[idx]=1.            ; Put possible<=0 ff values to 1.
   noisy_image*=ffres
endif

;; Additive read-out noise (Gaussian)
if keyword_set(SIGMA_RON) then $
   noisy_image+=randomn(seed_ron,npx,npy,/NORMAL,/DOUBLE)*sigma_ron

; Force to zero negative values
if keyword_set(POSITIVE) then begin
   idx=where(noisy_image LT 0, c)
   if (c GT 0) then noisy_image[idx]=0.
endif
```

```
CALLING SEQUENCE:
  noisy_image = addnoise(input_image,                          $
                         PHOT_NOISE=phot_noise,                $
                         SIGMA_DARK=sigma_dark,                $
                         DELTA_T=delta_t,                      $
                         EXODARK=exodark,                      $
                         GAIN_L3CCD=gain_l3ccd,                $
                         FFOFFSET=ffoffset,                    $
                         SIGMA_RON=sigma_ron,                  $
                         POSITIVE=positive,                    $
                         OUT_TYPE=out_type                     )
```

<u>img formation w/noise:</u>

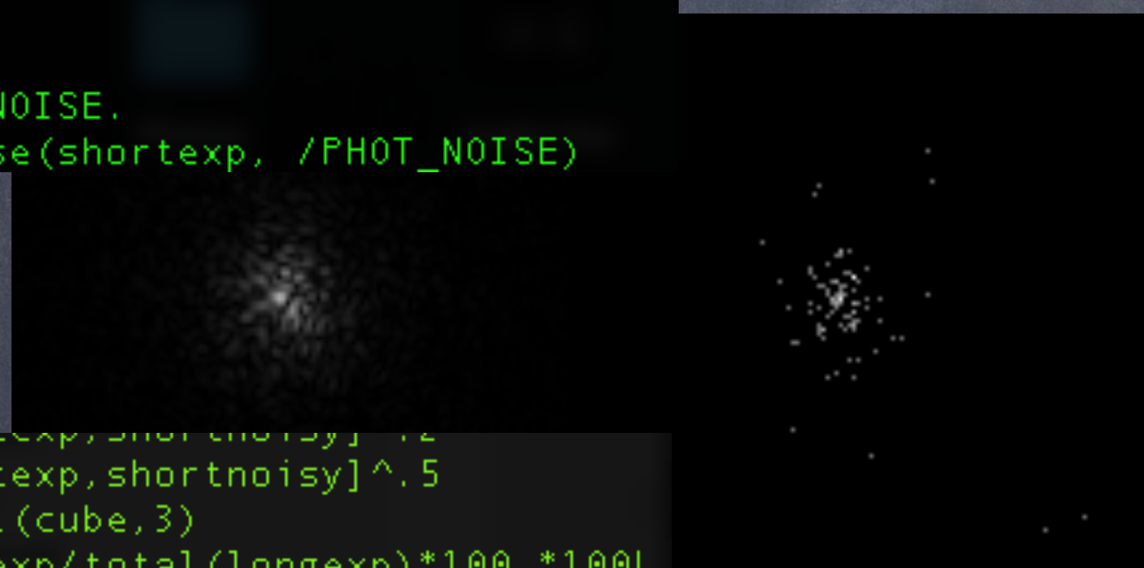
1- ‘add’ photon noise on one short-exp. PSF (in function of N…),
2- long-exp. PSF (100N photons!),
3- ‘add’ photon noise on the long-exp. PSF,
4- compare long-exp. & short-exp. noisy images (and ‘clean’ images).

Images & turbulence - 27

```
IDL> restore, 'PSF_r0=10cm_L0=10m_lambda=500nm.sav'
IDL> help
% At $MAIN$
CUBE_PSF          FLOAT     = Array[128, 128, 100]
SHORTEXP          DOUBLE    = Array[128, 128]
Compiled Procedures:
    $MAIN$

Compiled Functions:

IDL> shortexp=cube_PSF[*,*,0]
IDL> total(shortexp)
      0.19702147
IDL> shortexp=shortexp/total(shortexp)*100.
IDL> total(shortexp)
       99.999664
IDL> .r addnoise
% Compiled module: ADDNOISE.
IDL> shortnoisy=addnoise(shortexp, /PHOT_NOISE)
```



```
IDL> tvscl, [shortexp,shortnoisy]^.5
IDL> longexp=total(cube,3)
IDL> longexp=longexp/total(longexp)*100.*100L
IDL> longnoisy=addnoise(longexp, /PHOT_NOISE)
IDL> tvscl, [longexp,longnoisy]^.5
```

comparison: visually or by means of a least mean square distance…

<u>img formation w/noise:</u>

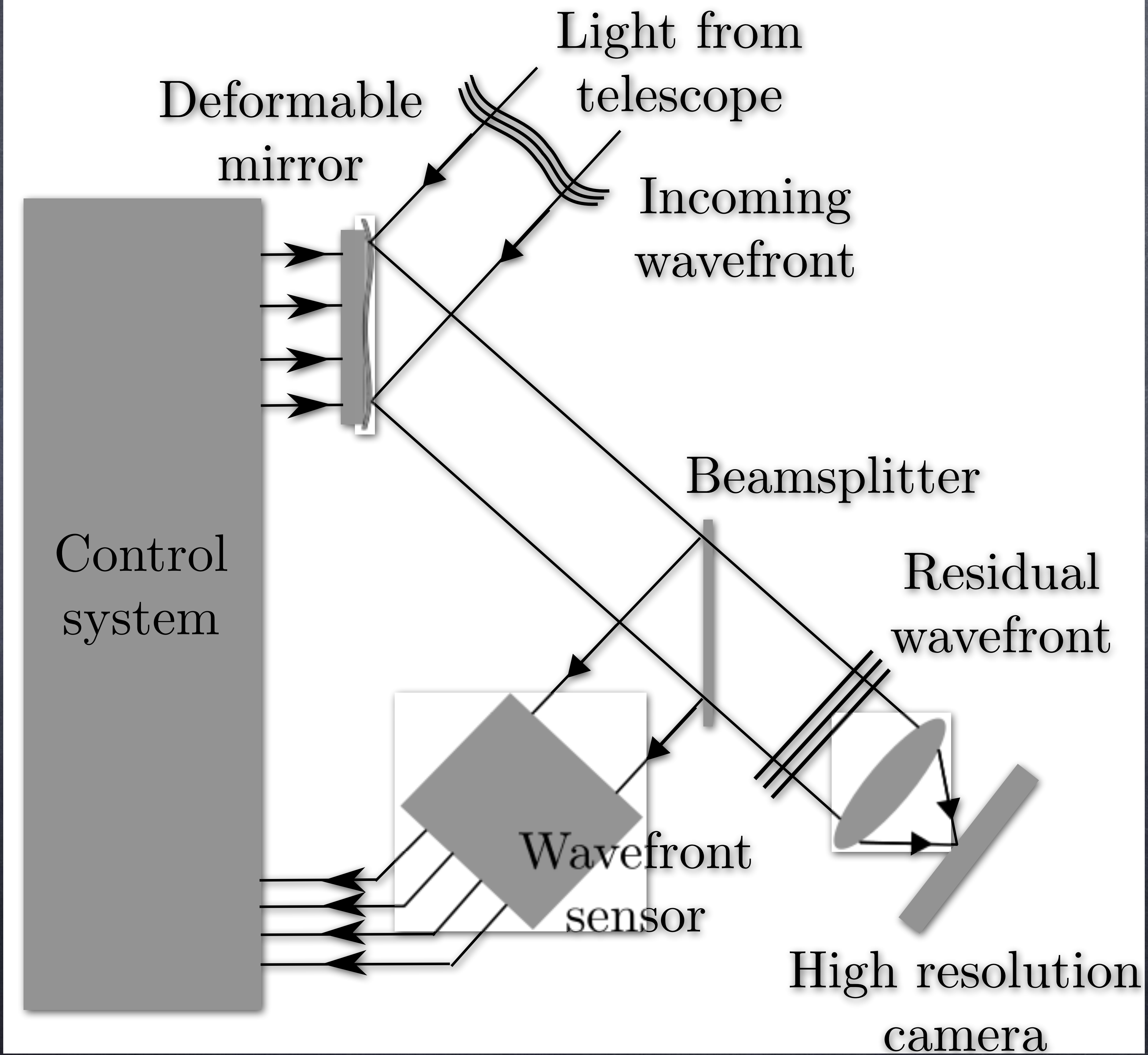
1- 'add' photon noise on one short-exp. PSF (with N photons/img),
2- long-exp. PSF (=> with 100N photons),
3- 'add' photon noise on the long-exp. PSF,
4- compare long-exp. & short-exp. noisy images (and 'clean' images).
5- possibly compare also with stacked image (from the 100 short-exp. noisy images)

```
---
REPORT
---

- Preliminary measures
+ introduction/context
+ PSD(r0, L0)
+ => influence of r0 and L0
+ rms(r0, L0)
+ => influence of r0 and L0
+ FWHM(r0 or lambda=>r0, L0)
+ => influence of r0 and L0
+ => comparison with the "seeing" lambda/r0
+ noisy images
```

Adaptive optics — 01

Adaptive optics — 02



Adaptive optics — 03

Some orders of magnitude concerning AO systems:

	@500nm	@2.2µm
spatial sampling (WFS analysis elements size)		
→ $d \approx r_0$	≈ 10 cm	≈ 60 cm
number of WFS analysis elements (≈ number of DM actuators)		
→ $N \propto (D/d)^2$, with D=10m	≈ 7500	≈ 200
temporal sampling		
→ $f \propto 10\ v/r_0$	≈ 1 kHz	≈ 0.2 kHz