CSIR

Defence, Peace, Safety and Security

Pyradi: an open-source toolkit for infrared calculation and data processing

SPIE S+D 8543-19

our future through science

Willers, Willers, Santos,
van der Merwe, Calitz, de Waal & Mudau

Infrared Data Analysis

Who has done infrared measurement data analysis?

You will understand the next slide….

our future through science

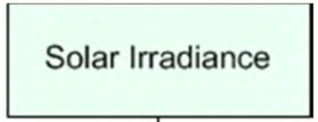
Infrared Data Analysis

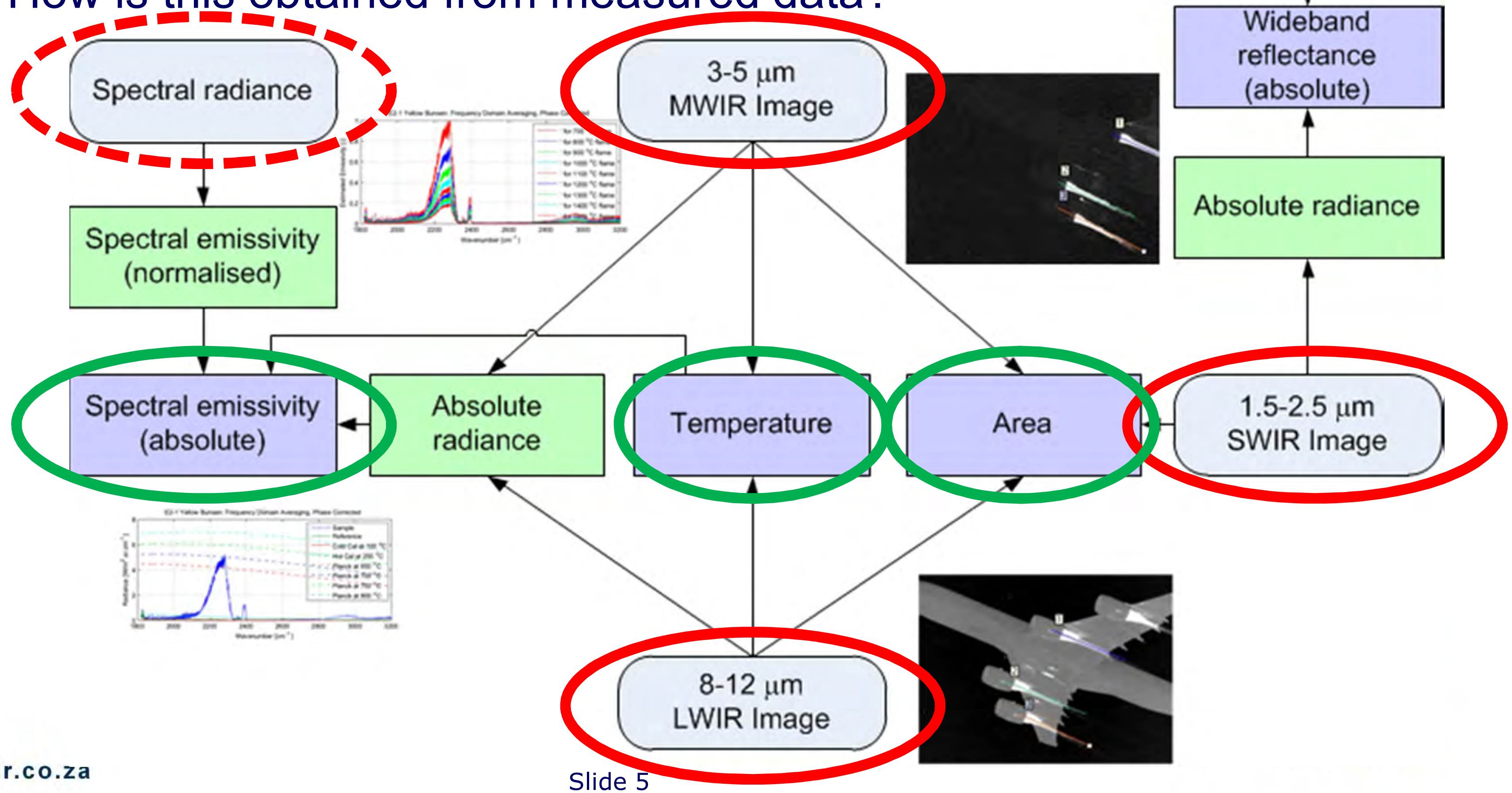
Signatures From Measurements

Main focus of infrared target modelling:

- temperature
- spectral emissivity
- area

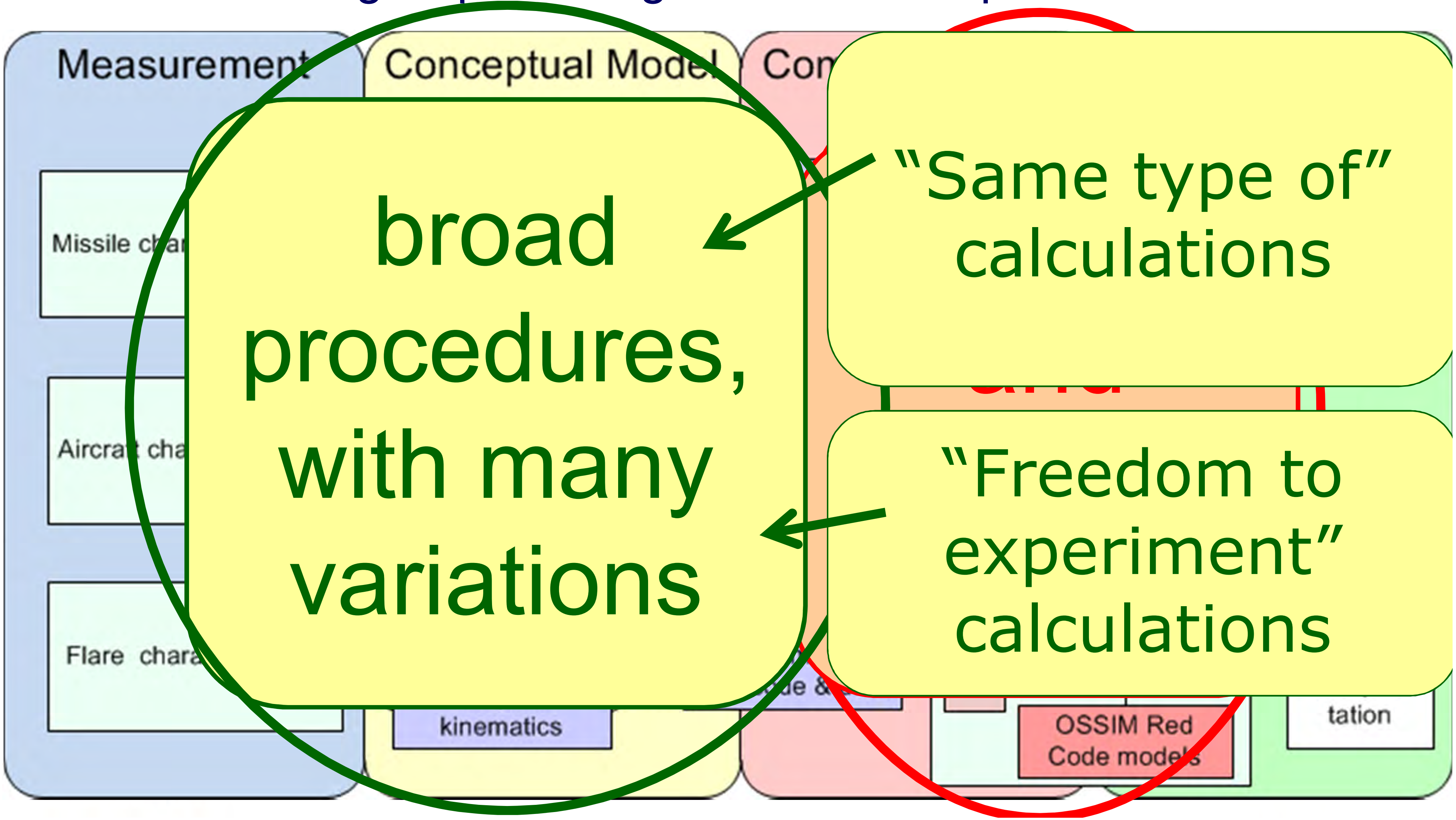
How is this obtained from measured data?





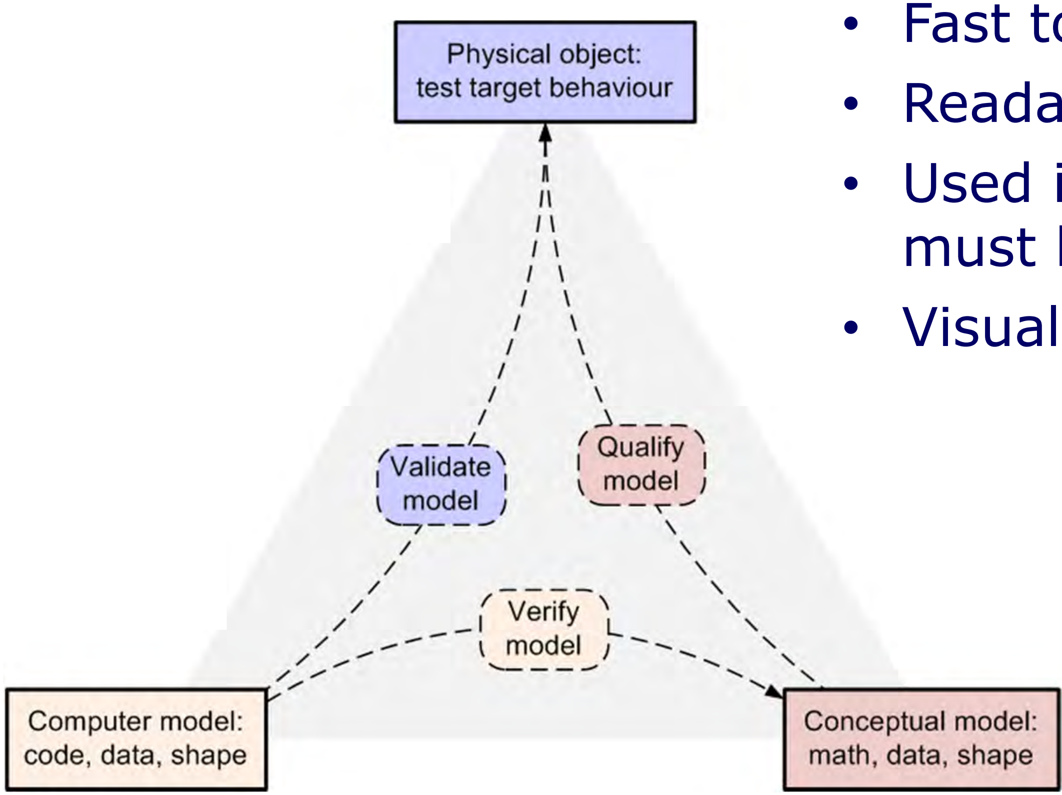
From Measurements to Results

Model building requires huge amount of repetitive calculation



Vital to Insight!

Key Requirements



- Easy to use.
- Fast to prototype.
- Readable, uncluttered code.
- Used in validation/verification: must be a well-tested library.
- Visualisation is critical!

our future through science

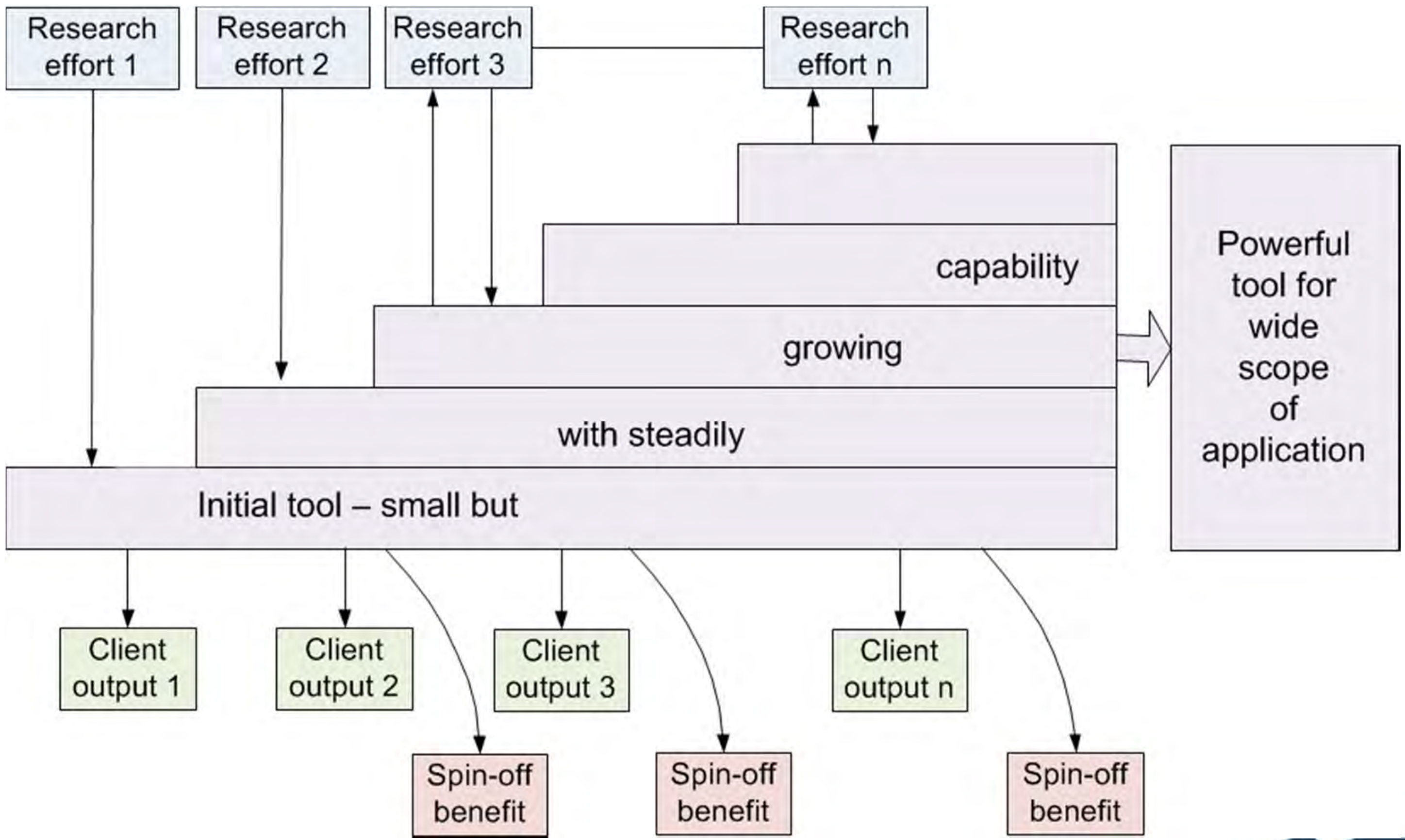
pyradi

Focussed on electro-optics calculations.

The toolkit covers

- models of physical concepts,
- mathematical operation,
- data manipulation, and
- graphical visualisation.

Cumulative Development



our future through science

Scripting Language

Matlab vs Python.

Extensive experience with both languages.

Python won:

- General purpose, mainstream, language.
- A more powerful and productive language.
- Many extensions in very many application areas.
- Numpy & Scipy equal to Matlab for scientific work.
- Superior visualisation & graphics tools.
- Easy to learn & very strong open source support.
- Zero acquisition cost & low usage cost.
- Negative: no Simulink, fewer toolboxes.

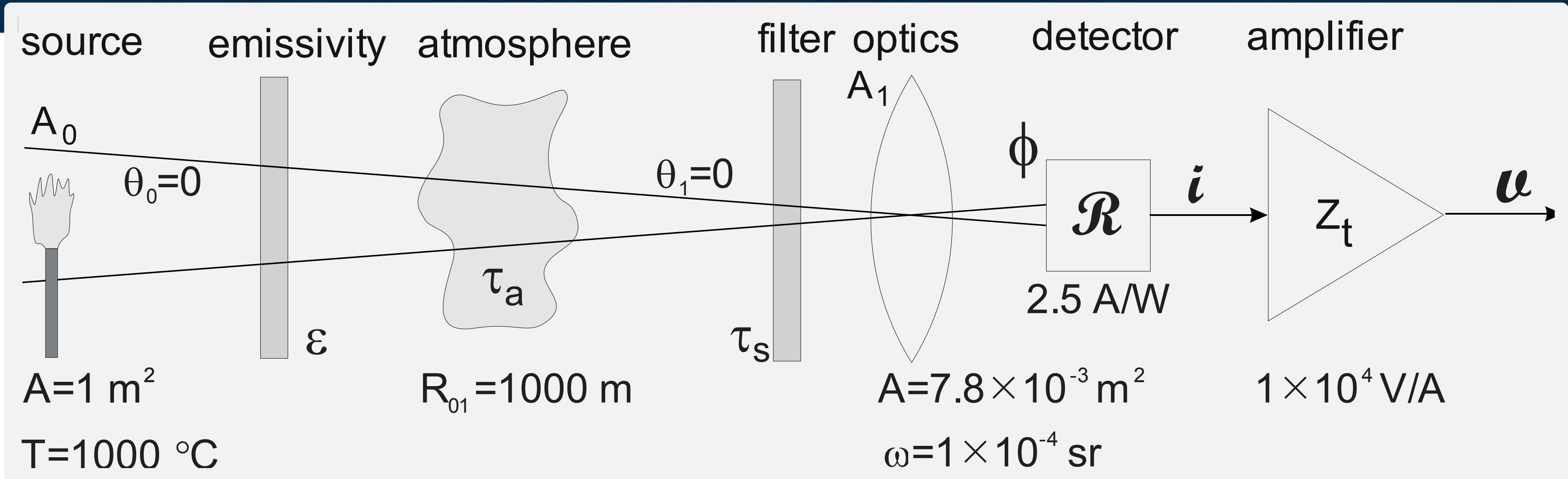
Unix Tool Philosophy

Write programs that do one thing and do it well.

Write programs to work together.

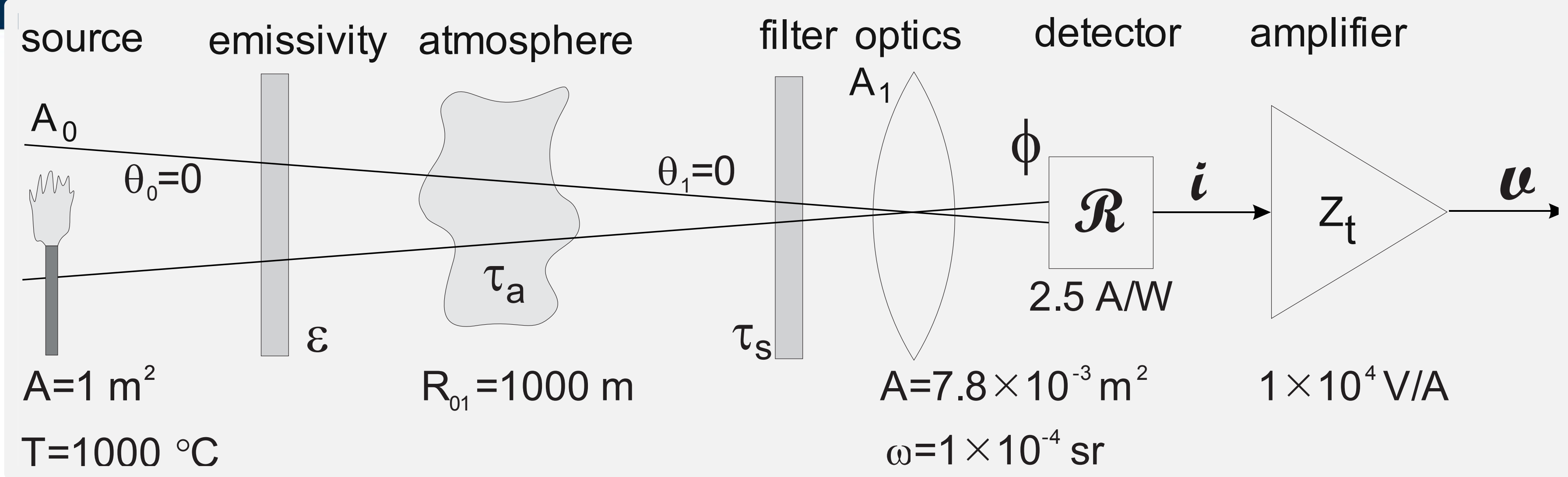
- Simple parts connected by clean interfaces.
- Clarity in coding.
- Write big programs putting together building blocks.
- Programmer time is expensive; conserve it.
- Quick prototyping before polishing.
- Design for the future, because it will be here sooner than you think.

Example: Flame Detection



Detect the presence or absence of a flame, through the atmosphere in 3-5 μm band.

Example: Flame Detection



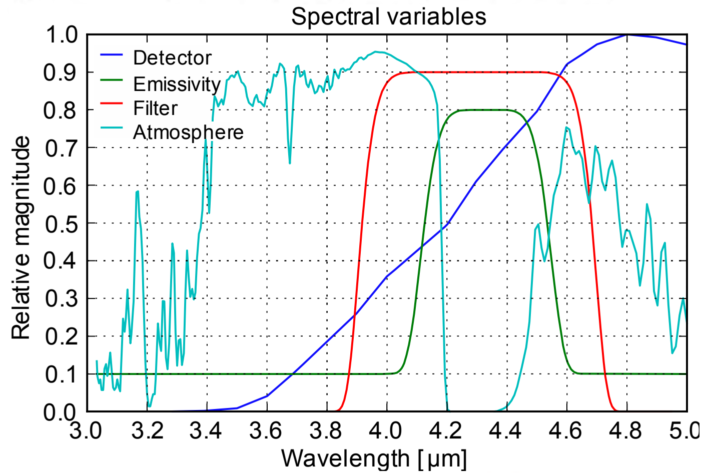
Signal from the flame:

$$v = Z_t \int_{A_0} \int_{A_1} \frac{1}{r_{01}^2} \int_0^\infty \epsilon_\lambda L_\lambda(T, A_0) \tau_{a\lambda} \tau_{s\lambda}(A_1) \mathcal{R}_\lambda \, d\lambda \, d(\cos\theta_0 A_0) \, d(\cos\theta_1 A_1).$$

Signal from the atmospheric radiance:

$$v = Z_t \omega_{\text{optics}} A_{\text{optics}} \int_0^\infty L_{\text{path}\lambda} \tau_{s\lambda} \mathcal{R}_\lambda \, d\lambda,$$

Modtran spectral transmittance & path radiance



Solution Page 1 of 2

```
#load atmospheric transmittance from file created in Modtran in wavenumbers
# the transmittance is specified in the wavenumber domain with
# 5 cm-1 intervals, but we want to work in wavelength with 2.5 cm-1
waven = numpy.arange(2000.0,  3300.0,  2.5).reshape(-1, 1)
wavel= ryutils.convertSpectralDomain(waven,  type='nw')

#remove comment lines, and scale path radiance from W/cm2.sr.cm-1  to W/m2.sr.cm-
1
tauA = ryfiles.loadColumnTextFile('data/path1kmflamesensor.txt',
        [1],abscissaOut=waven,  comment='%')
lpathwn = ryfiles.loadColumnTextFile('data/pathspaceflamesensor.txt',
        [9],abscissaOut=waven,  ordinateScale=1.0e4,  comment='%')

#convert path radiance spectral density from 1/cm^-1 to 1/um, at the sample
#wavenumber points
(dum, lpathwl) = ryutils.convertSpectralDensity(waven,  lpathwn, type='nw')

#load the detector file in wavelengths, and interpolate on required values
detR = ryfiles.loadColumnTextFile('data/detectorflamesensor.txt',
       [1],abscissaOut=wavel,  comment='%')

#construct the flame emissivity from parameters
emis = ryutils.sfilter(wavel,center=4.33, width=0.45, exponent=6, taupass=0.8,
         taustop=0.1 )
#construct the sensor filter from parameters
sfilter = ryutils.sfilter(wavel,center=4.3, width=0.8, exponent=12,
          taupass=0.9, taustop=0.0001)
```

Only two pages

our future through science

Solution Page 2 of 2

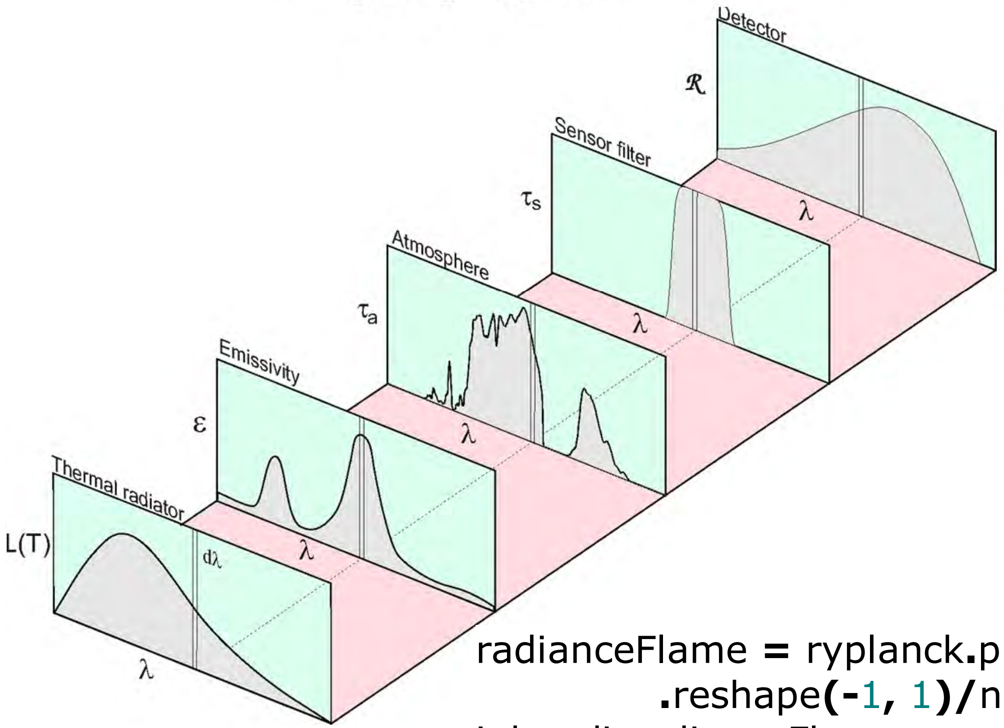
```
# define sensor scalar parameters
opticsArea=7.8e-3 # optical aperture area [m2]
opticsFOV=1.0e-4 # sensor field of view [sr]
transZ=1.0e4 # amplifier transimpedance gain [V/A]
responsivity=2.5 # detector peak responsivity =A/W]

# define the  flame properties
flameTemperature = 1000+273.16    # temperature in [K]
flameArea = 1  # in [m2]
distance = 1000  # [m]
fill = (flameArea /distance**2) /  opticsFOV # how much of FOV is filled
fill = 1 if fill > 1 else fill # limit target solid angle to sensor FOV

# flame get spectral radiance in  W/m^2.sr.cm-1
radianceFlame = ryplanck.planck(waven, flameTemperature,  type='en')\
           .reshape(-1, 1)/numpy.pi
inbandirradianceFlame = radianceFlame * detR * tauA * emis * sfilter *\
           fill * opticsFOV
totalirradianceFlame = numpy.trapz(inbandirradianceFlame.reshape(-1, 1),
          waven, axis=0)[0]
signalFlame = totalirradianceFlame *transZ*responsivity *opticsArea

# now do path
inbandirradiancePath =  lpathwn * detR * sfilter * opticsFOV
totalirradiancePath = numpy.trapz(inbandirradiancePath.reshape(-1, 1),waven, axis=0)[0]
signalPath = totalirradiancePath * transZ*responsivity *opticsArea
```

The code is:

- Compact
- Clean
- Intuitive

Multi-spectral Calculations

$$v = Z_t \int_{A_0} \int_{A_1} \frac{1}{r_{01}^2} \int_0^\infty \epsilon_\lambda L_\lambda(T, A_0) \tau_{a\lambda} \tau_{s\lambda}(A_1) \mathcal{R}_\lambda \, d\lambda \, d(\cos\theta_0 A_0) \, d(\cos\theta_1 A_1).$$

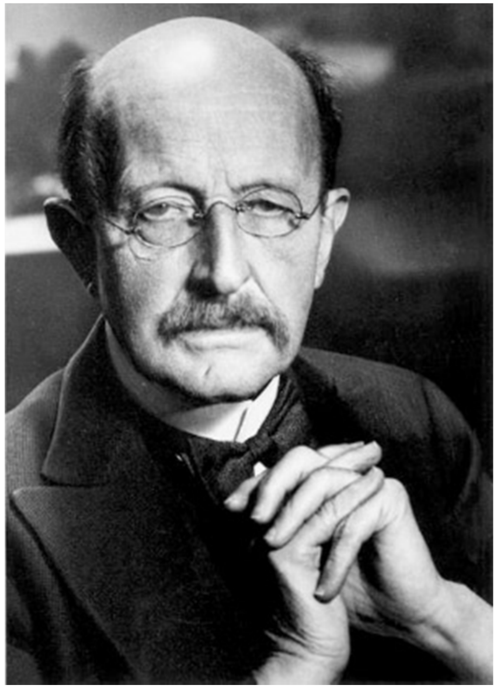


- Spectral variables are vectors.
- Calculate irradiance at each wavelength 'bin' by vector element-wise multiplication.
- Add flux at all wavelengths.

```
radianceFlame = ryplanck.planck(waven, flameTemperature,  type='en')\
            .reshape(-1, 1)/numpy.pi
inbandirradianceFlame = radianceFlame* emis* tauA  * sfilter * detR *\
            opticsFOV
totalirradianceFlame = numpy.trapz(inbandirradianceFlame.reshape(-1, 1),
            waven, axis=0)[0]
signalFlame = totalirradianceFlame *transZ*responsivity *opticsArea
```

Planck's Law

$$M_{e\lambda}(T) = \frac{2\pi h c^2}{\lambda^5 \left(e^{\frac{hc}{\lambda kT}} - 1\right)}$$

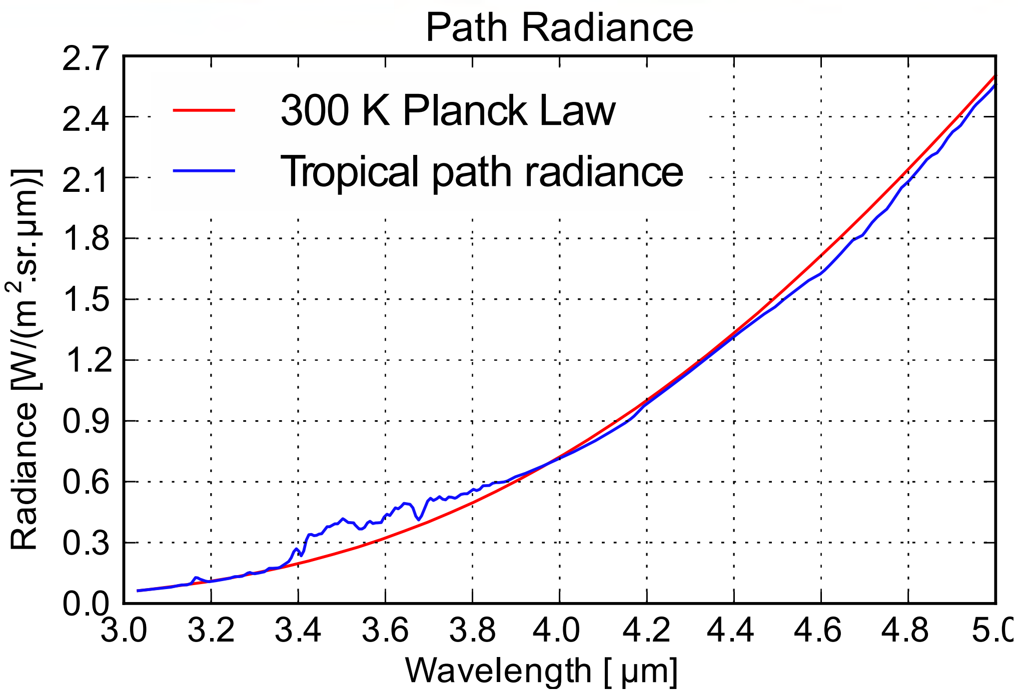


Pyradi provides functions for Planck Law emittance, as well as Planck Law temperature derivative in terms of:

[W] or [q/s] and [μm, cm^{-1}, Hz].

```
Mel = planck(wl, temperature[0], type='el') # [W/(m$^2$.$\mu$m)]
Mql = planck(wl, temperature[0], type='ql') # [q/(s.m$^2$.$\mu$m)]
Men = planck(n, temperature[0], type='en')  # [W/(m$^2$.cm$^{-1}$)]

Mqn = dplanck(n, temperature[0], type='qn')  # [q/(s.m$^2$.cm$^{-1}$.K)]
Mef = dplanck(f, temperature[0], type='ef')  # [W/(m$^2$.Hz.K)]
Mqf = dplanck(f, temperature[0], type='qf')  # [q/(s.m$^2$.Hz.K)]
```

Visualising Path Radiance



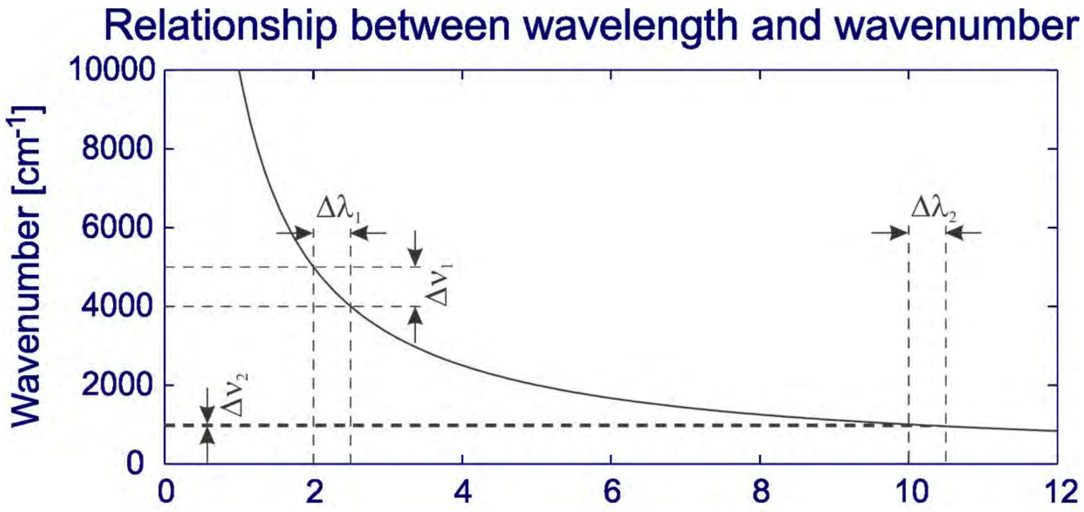
Flame sensor path:

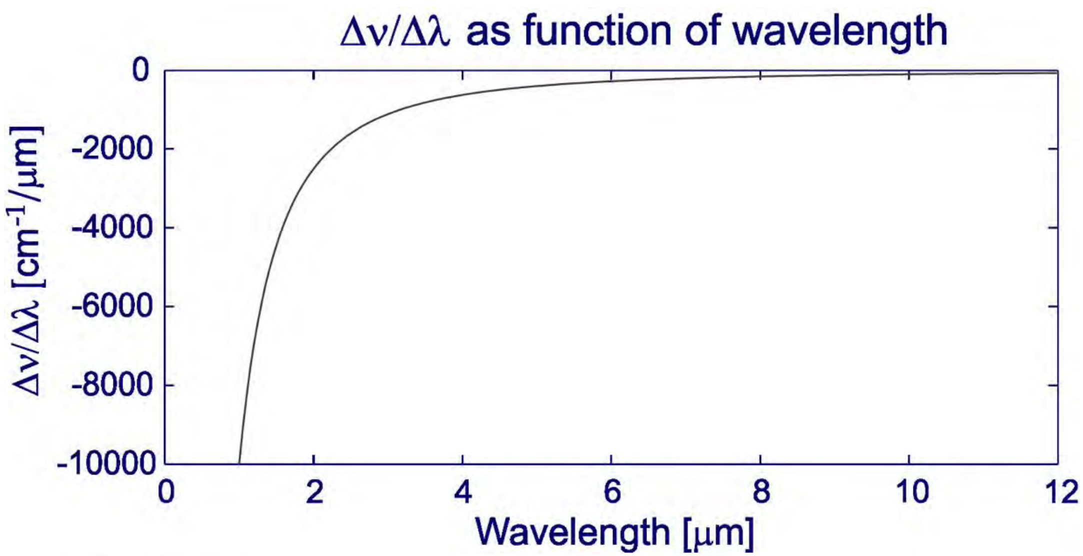
- Horizontal path to space.
- Notice correspondence.
- Visualisation important!

```
#check path radiance against Planck's Law for atmo temperature
LbbTropical = ryplanck.planck(wavel, 273+27, type='el').reshape(-1, 1)/numpy.pi
plot1.plot(2, wavel, LbbTropical, plotCol=['r'], label=['300 K Planck Law'])
plot1.plot(2, wavel, lpathwl, plotCol=['b'], label=['Tropical path radiance'])
currentP = plot1.getSubPlot(2)
currentP.set_xlabel('Wavelength [$\mu$m]')
currentP.set_ylabel('Radiance [W/(m$^2$.sr.$\mu$m)]')
currentP.set_title('Path Radiance')
```

our future through science

Wavelength and Wavenumber





Wavenumber is given by

$$\nu = 10^4/\lambda$$

where λ is in units of μm.

The conversion of a spectral quantity requires the derivative, as follows:

$$d\nu = \frac{-10^4}{\lambda^2} d\lambda = \frac{\nu^2}{-10^4} d\lambda$$

$$dL_\nu = dL_\lambda \frac{\lambda^2}{10^4} = dL_\lambda \frac{10^4}{\nu^2}$$

Pyradi converts between spectral variables:

[μm, cm^{-1}, Hz],

as well as spectral densities:

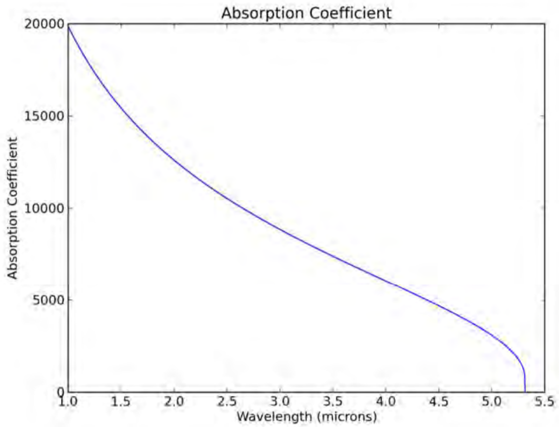
[W/($m^2\cdot\mu$m)],

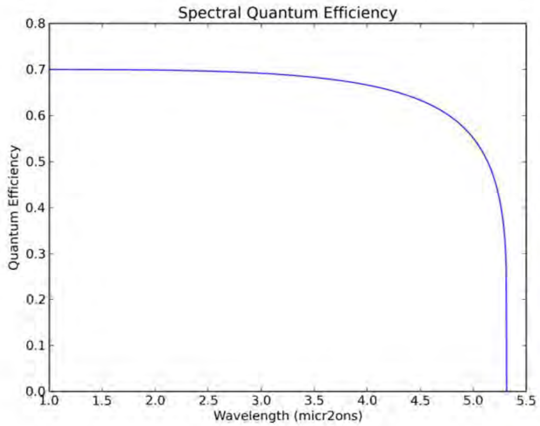
[W/($m^2\cdot$cm-1)],

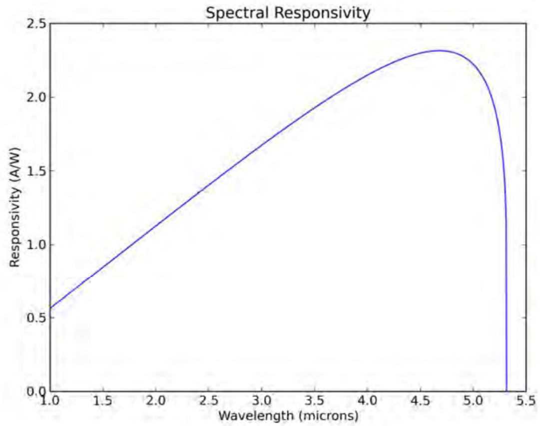
[W/($m^2\cdot$Hz)].

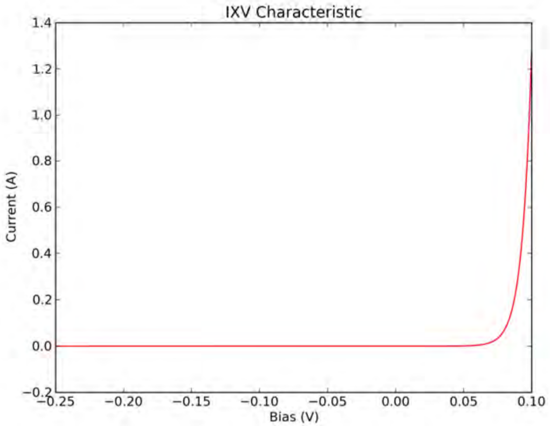
Detector Modelling

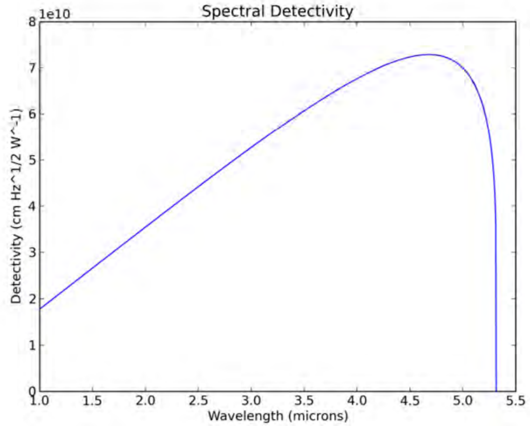
Detector model, based on first principles physics.
Example shown is for InSb detector.



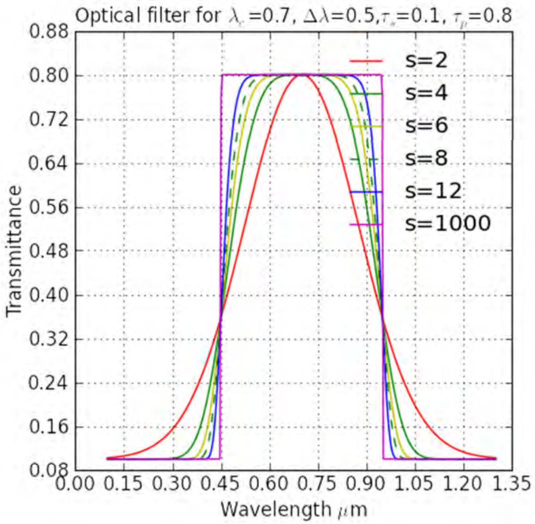


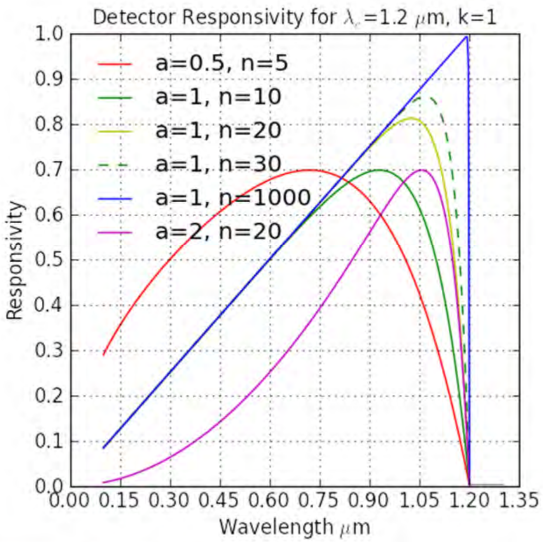


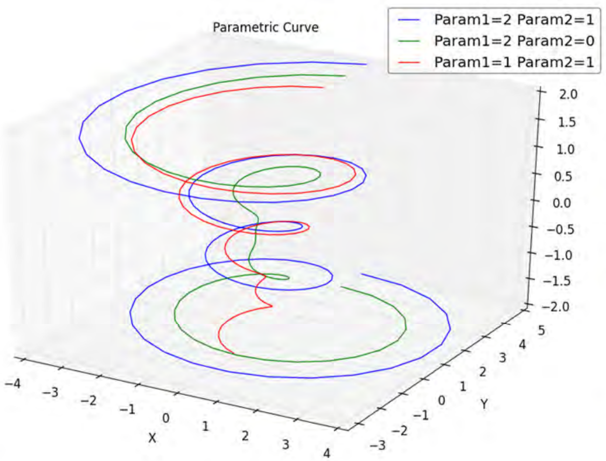




Utility Functions

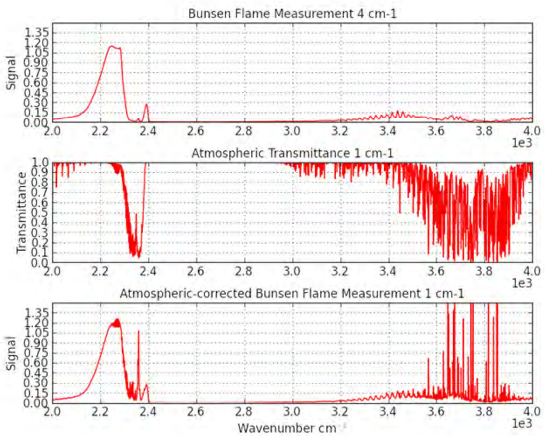


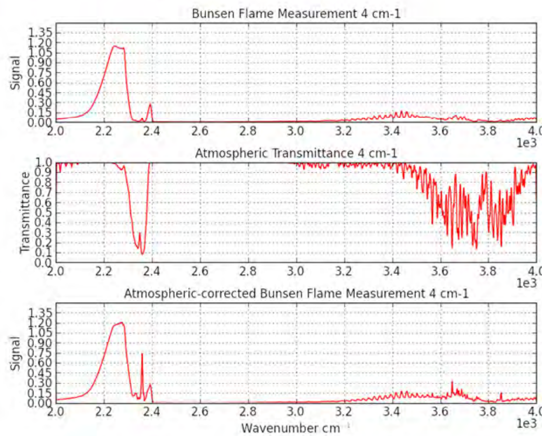




General purpose models:

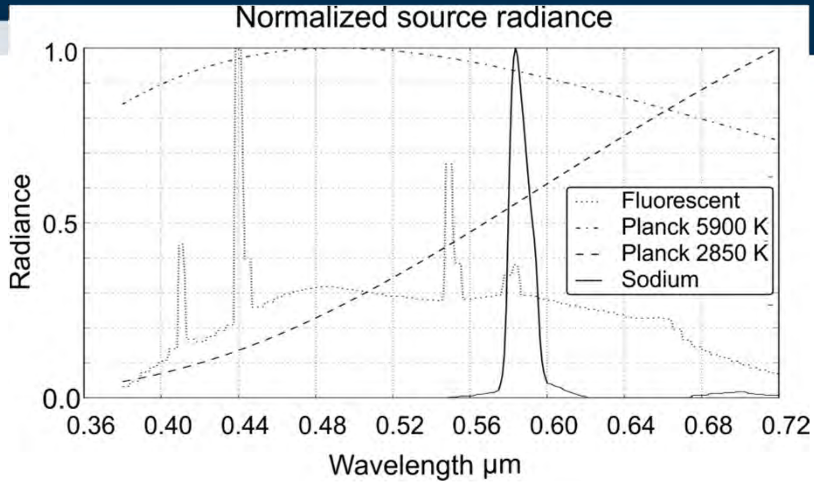
- Spectral filter shapes.
- Photon detector shapes.
- Spectral convolution.
- Effective value normalisation.
- Various 2-D & 3-D plotting functions.

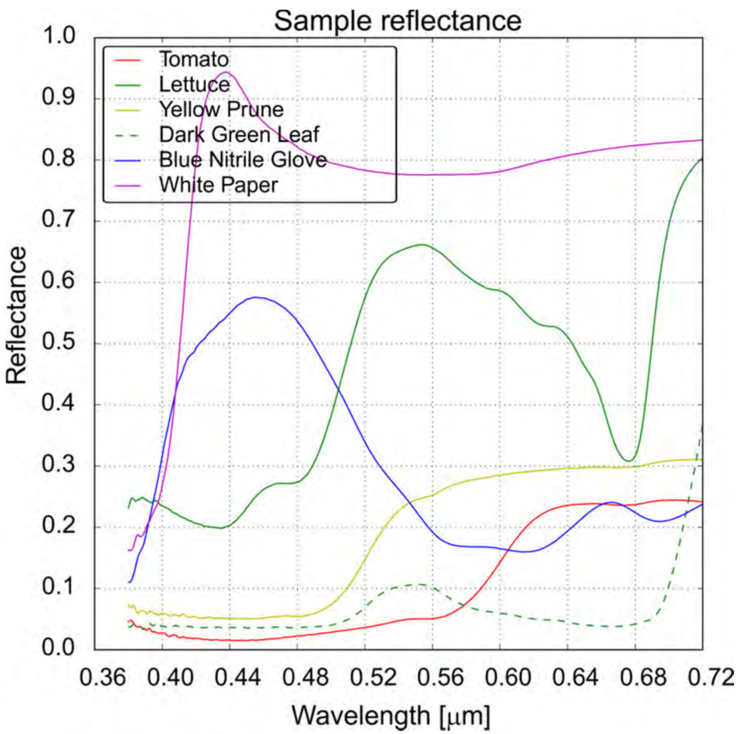


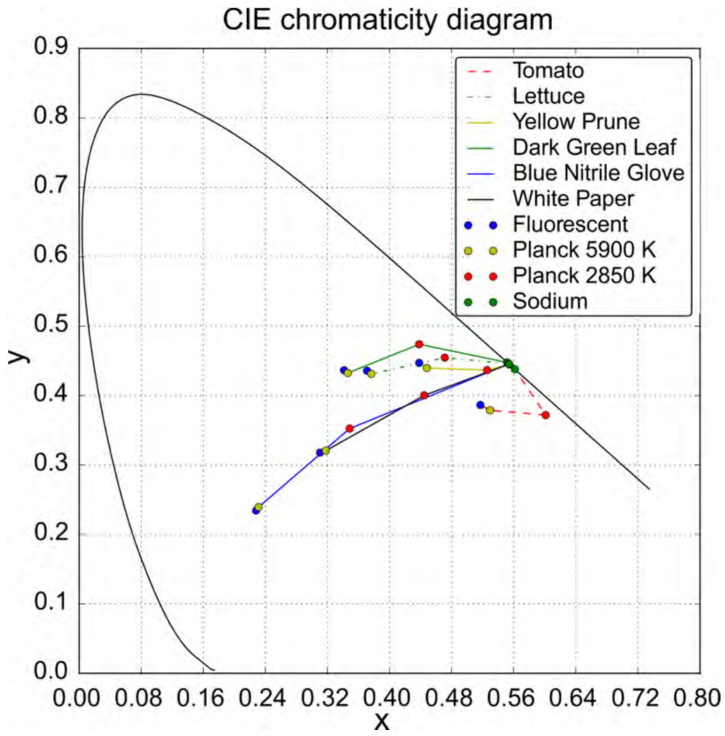


Chromaticity Coordinates

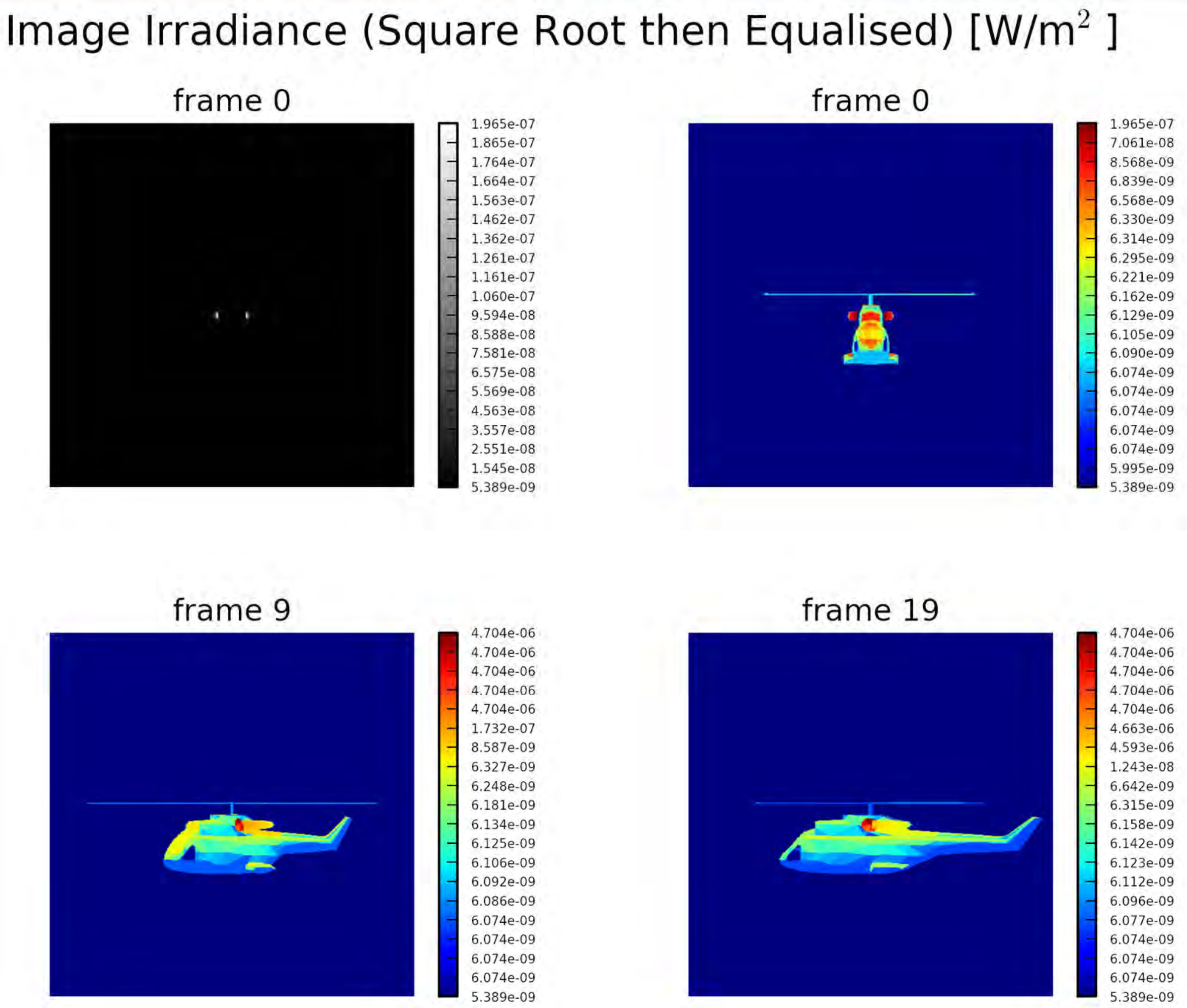
Basic functionality for CIE-1931 coordinate calculation for spectral source radiance and sample reflectance







Histogram Equalisation



Reading Data Files

- Binary raw image files: select frames, size & data type.

```
imagefile = 'data/sample.ulong'
rows = 100
cols = 100
vartype = numpy.uint32
framesToLoad =  [1, 3, 5, 7]
frames, img = readRawFrames(imagefile, rows, cols, vartype, framesToLoad)
```

- FLIR Inc (CEDIP) *.ptw image files: select frames.

```
ptwfile  = 'data/PyradiSampleMWIR.ptw'
header = readPTWHeader(ptwfile)
#loading sequence of frames
framesToLoad = [3,4,10]
data = getPTWFrames (header, framesToLoad)
```

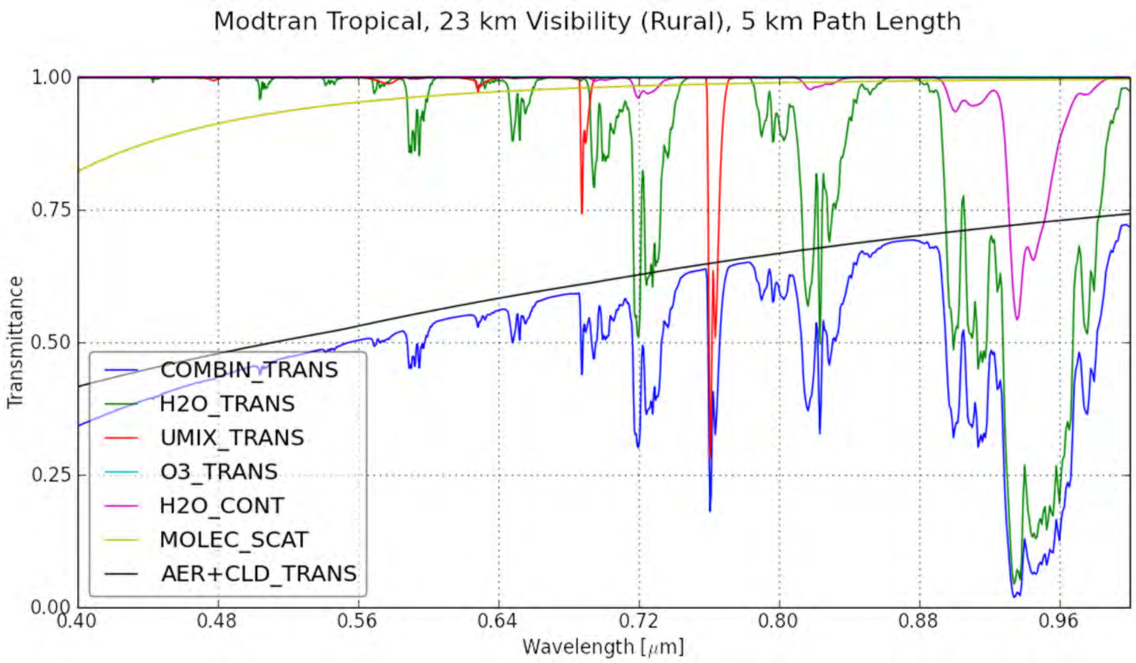
(Instrument calibration scaling still to follow).

our future through science

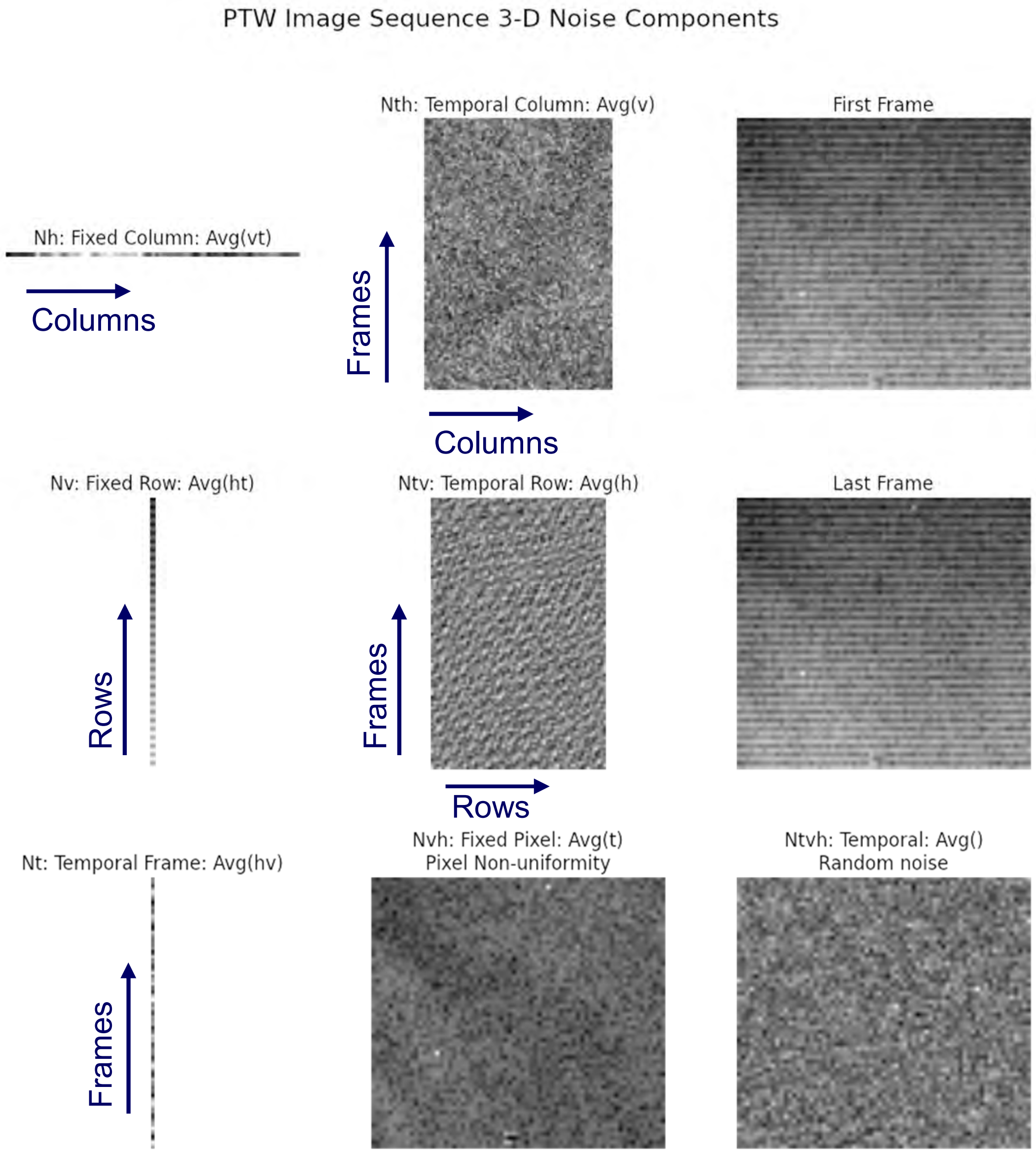
Read Modtran tape7 Files

```
colSelect =  ['FREQ_CM-1', 'COMBIN_TRANS', 'H2O_TRANS', 'UMIX_TRANS', \
             'O3_TRANS', 'H2O_CONT', 'MOLEC_SCAT', 'AER+CLD_TRANS']
tape7= loadtape7("data/tape7VISNIR5kmTrop23Vis", colSelect )

wavelen = ryutils.convertSpectralDomain(tape7[:,0],  type='nl')
mT = ryplot.Plotter(1, 1, 1,"Modtran Tropical, 23 km Visibility (Rural)"\
             + ", 5 km Path Length",figsize=(12,6))
mT.plot(1, wavelen, tape7[:,1:], "","Wavelength [$\mu$m]", "Transmittance",
             label=colSelect[1:],legendAlpha=0.5, pltaxis=[0.4,1, 0, 1],
             maxNX=10, maxNY=4, powerLimits = [-4,  4, -5, 5])
mT.saveFig('ModtranPlot.png')
```

- Select columns by header title.
- Supports all four IEMSCT file variants.



3-D Noise Analysis

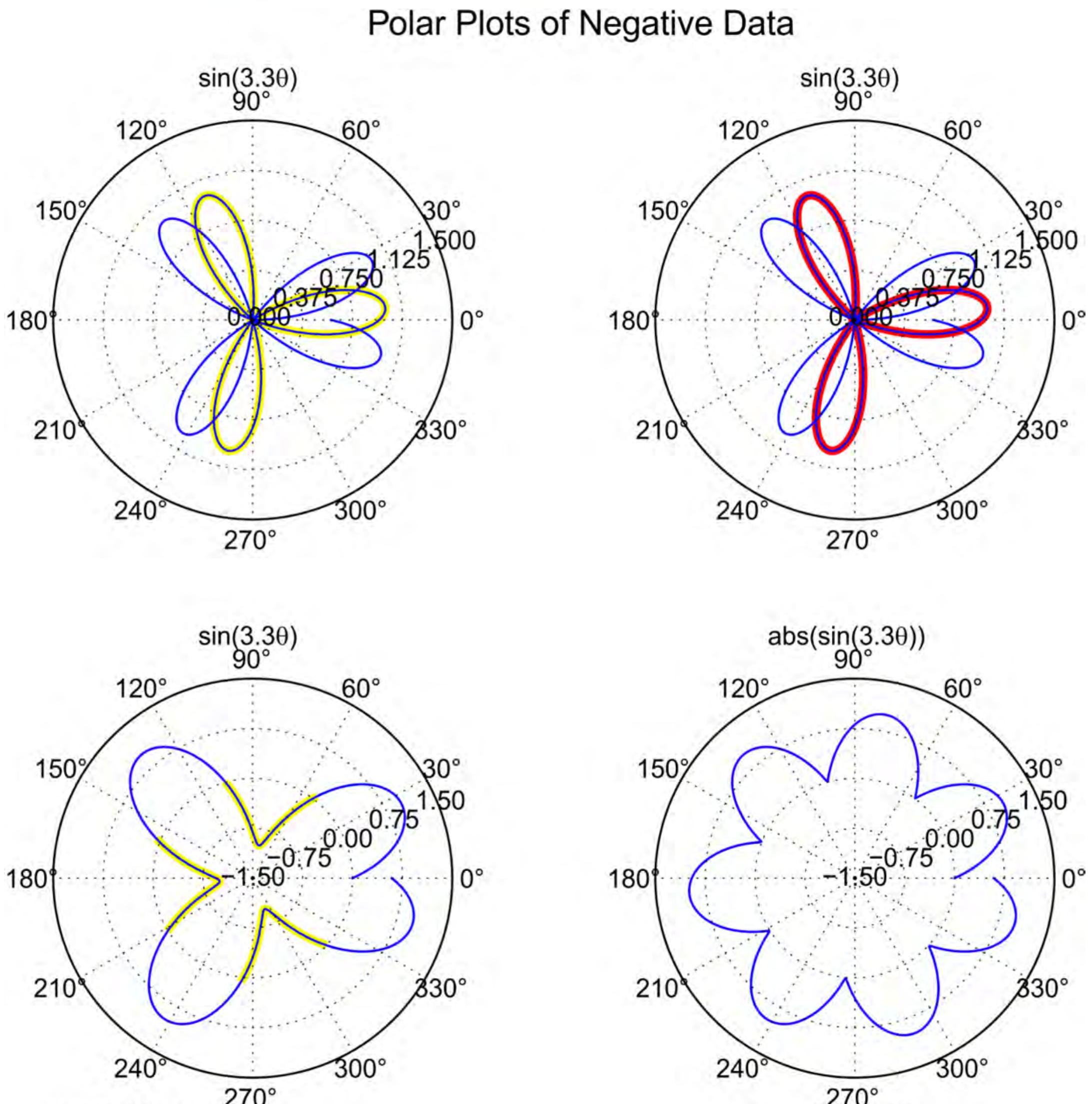


3-D noise analysis of image sequences.

- Averages and filters along frames, rows and columns.
- Isolates individual noise causes.

our future through science

Polar Plots



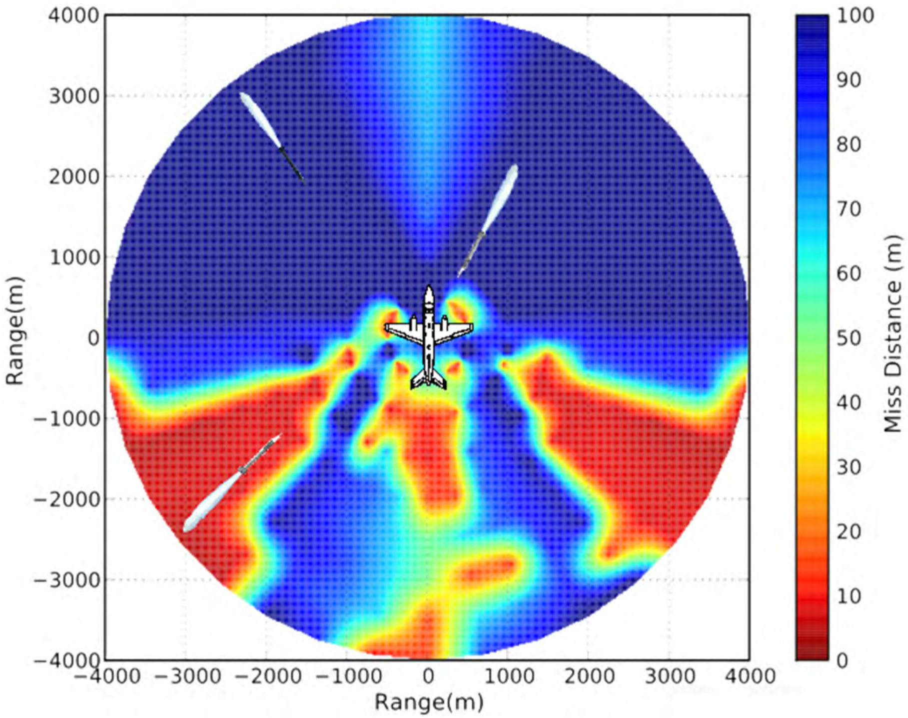
- -1 is a π phase shift: plot on ‘other’ side.
- Confuse direction with magnitude!
- Two options:
  - Highlight negative.
  - Zero not in centre.

our future through science

Aircraft Vulnerability

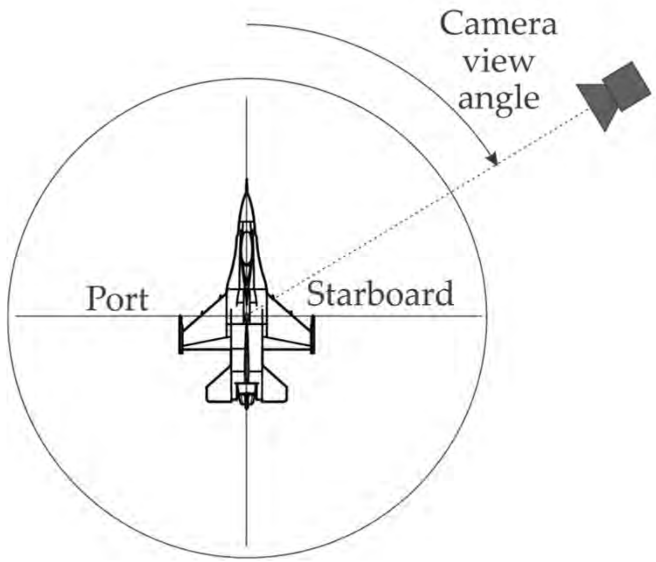
Missiles launched from red position are kills.

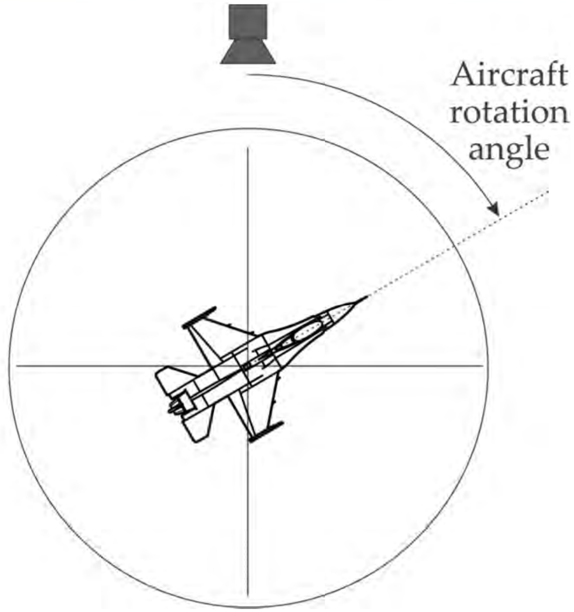
- Why are countermeasures not effective?
- Investigate Jam / Signal ratios.
- Investigate geometry: aircraft fuselage obscuration.
- Investigate angular 3-D signatures.



Welcome to the haunted house of 3-D signatures!

Two 3-D Scenarios

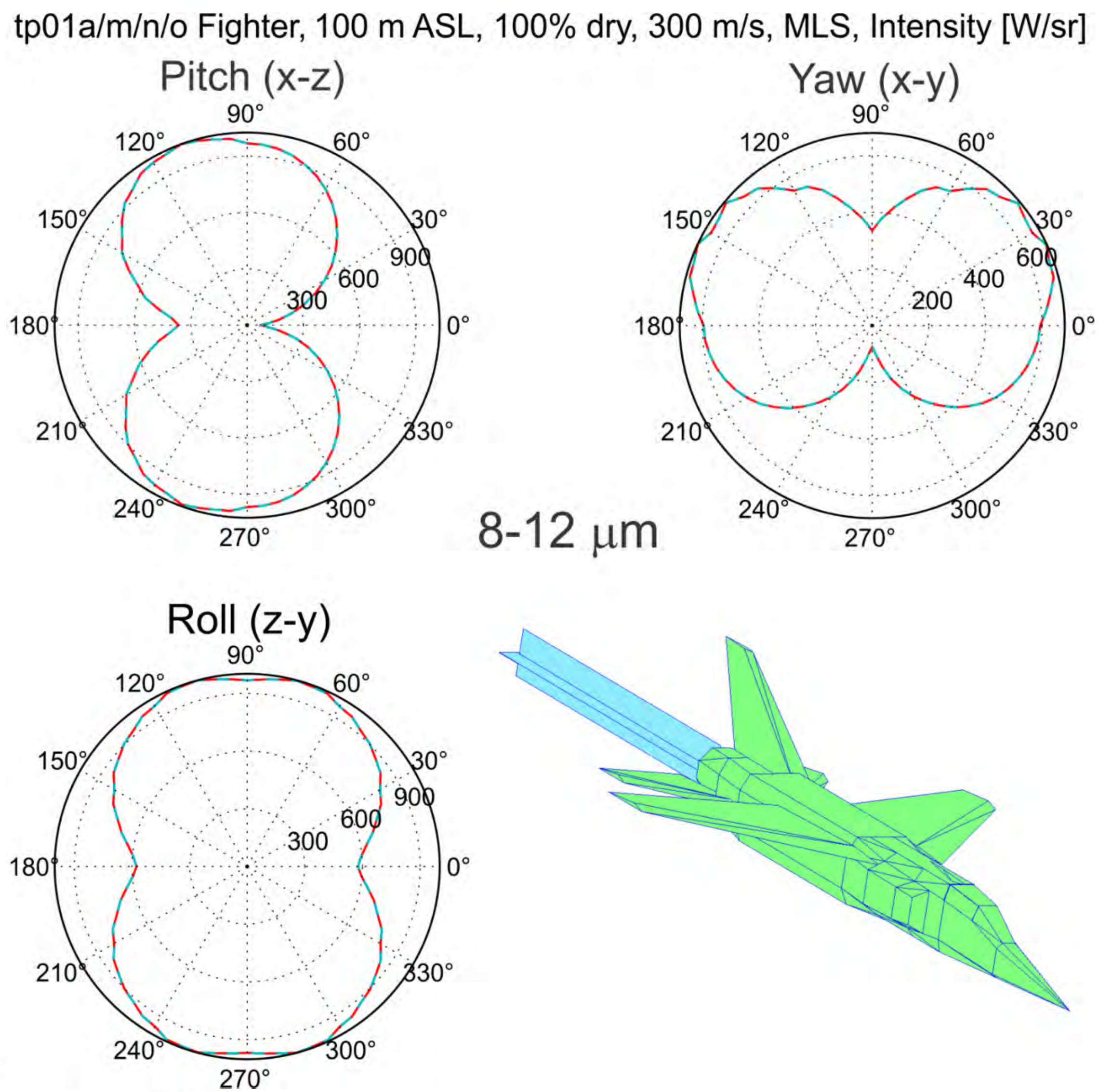




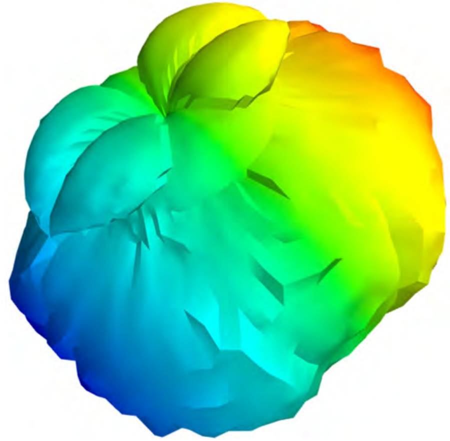
- Stationary target.
- Orbiting camera.
- Varying background.
- Stable sun position.

- Stationary camera.
- Rotating target.
- Varying sun position.
- Stable background.

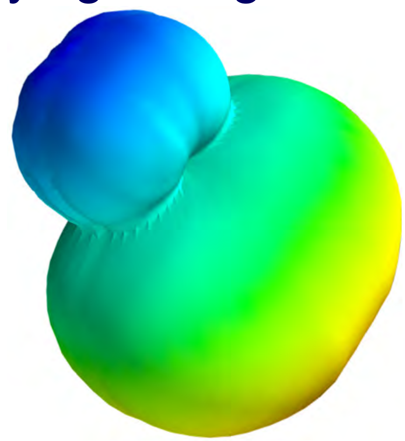
Intensity Polar Plots 8-12 μm



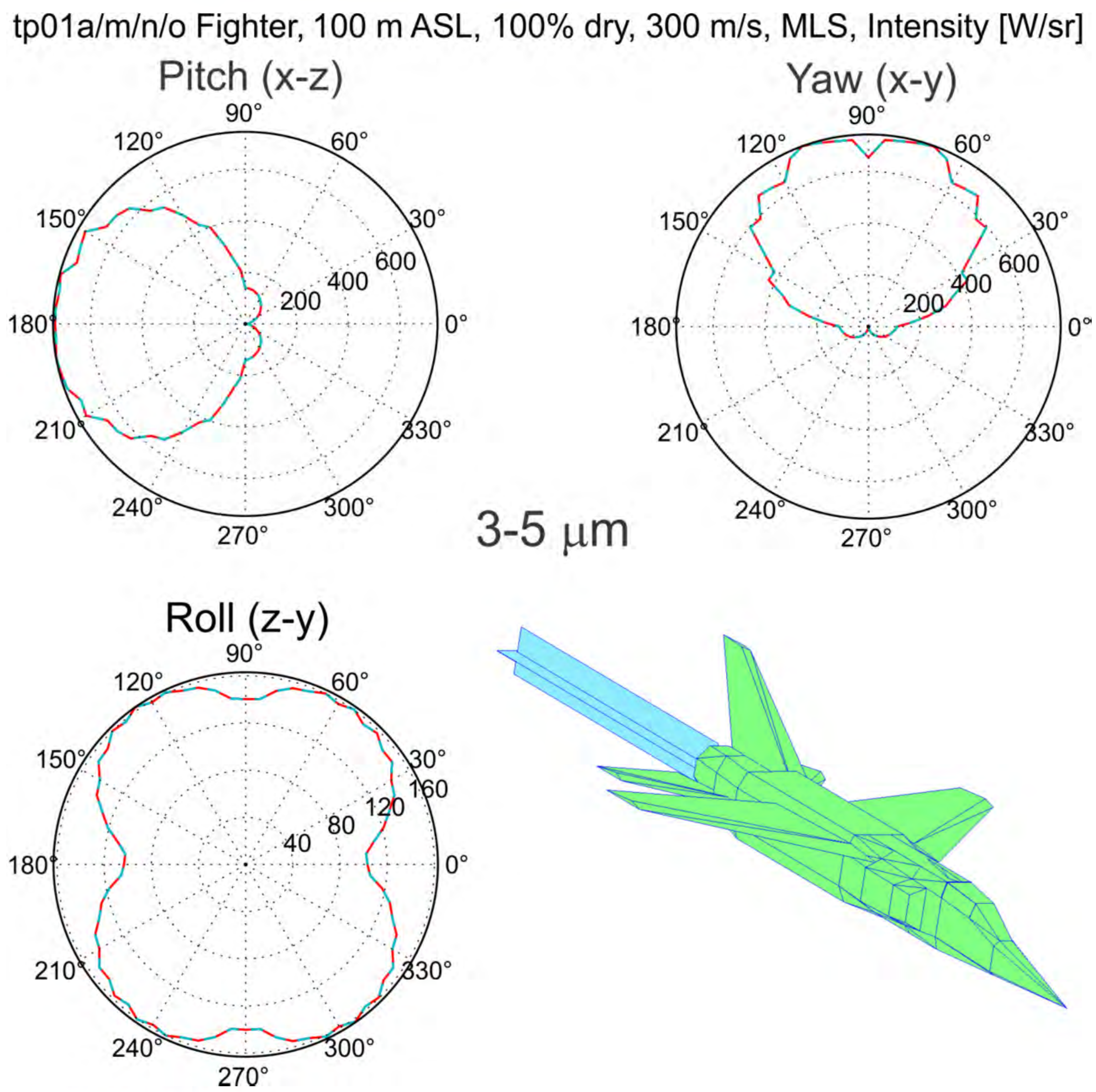
Rotating target:
constant background



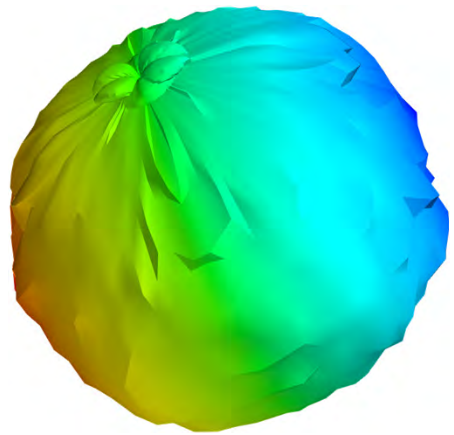
Orbiting sensor:
varying background



Intensity Polar Plots 3-5 μm

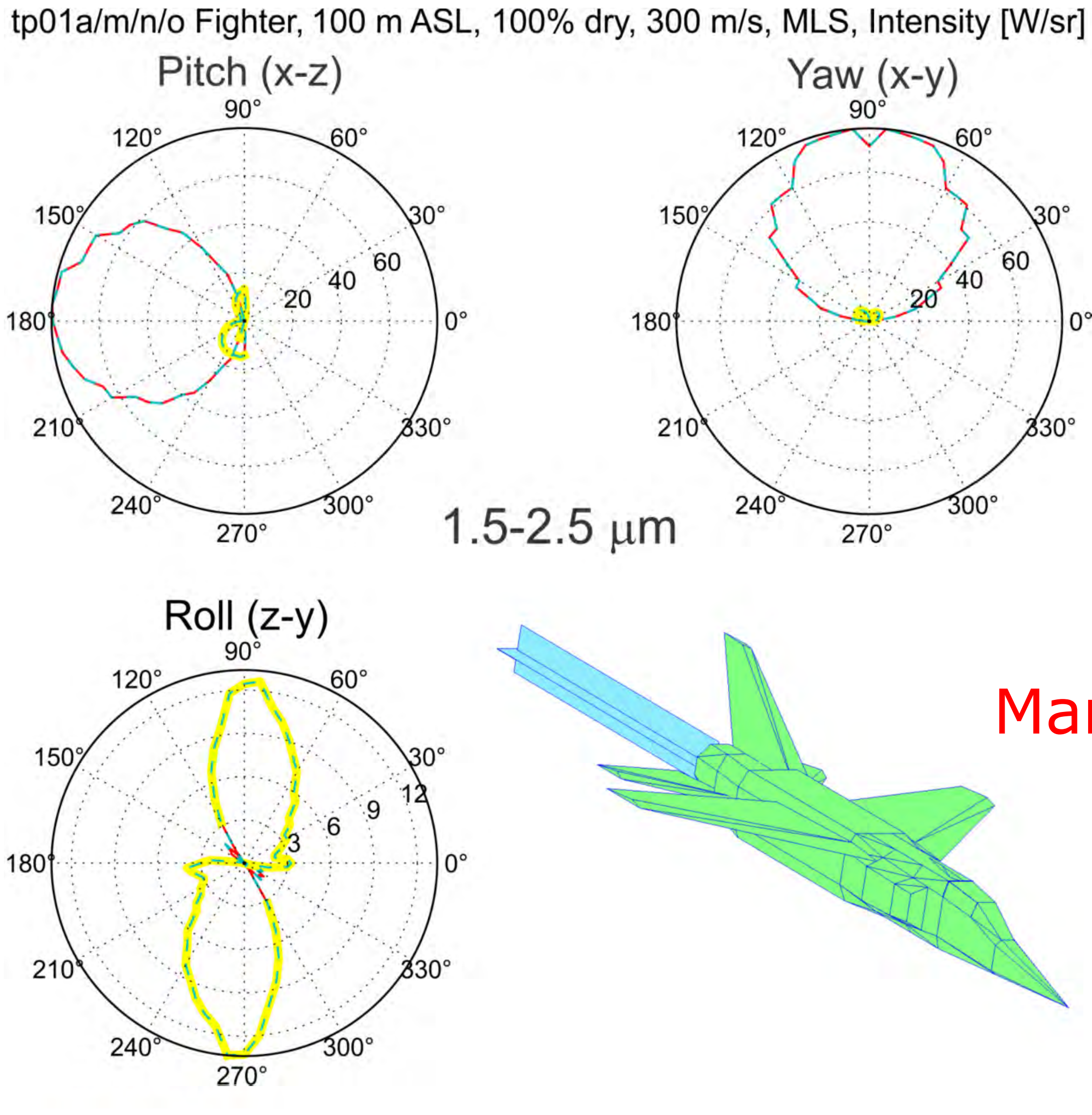


Rotating target:
constant background

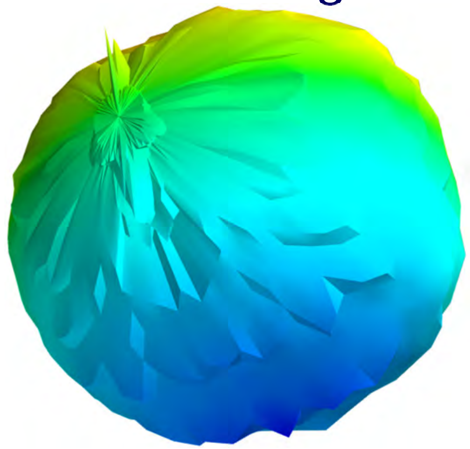


Orbiting sensor:
varying background

Intensity Polar Plots 1.5-2.5 μm

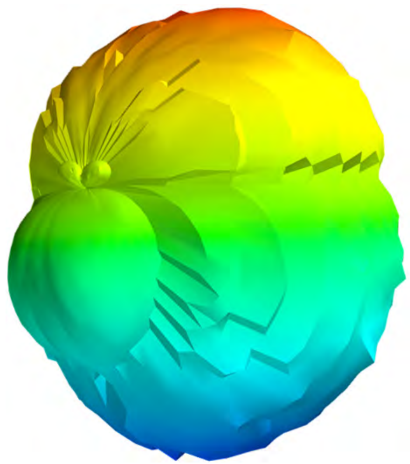


Rotating target:
constant background

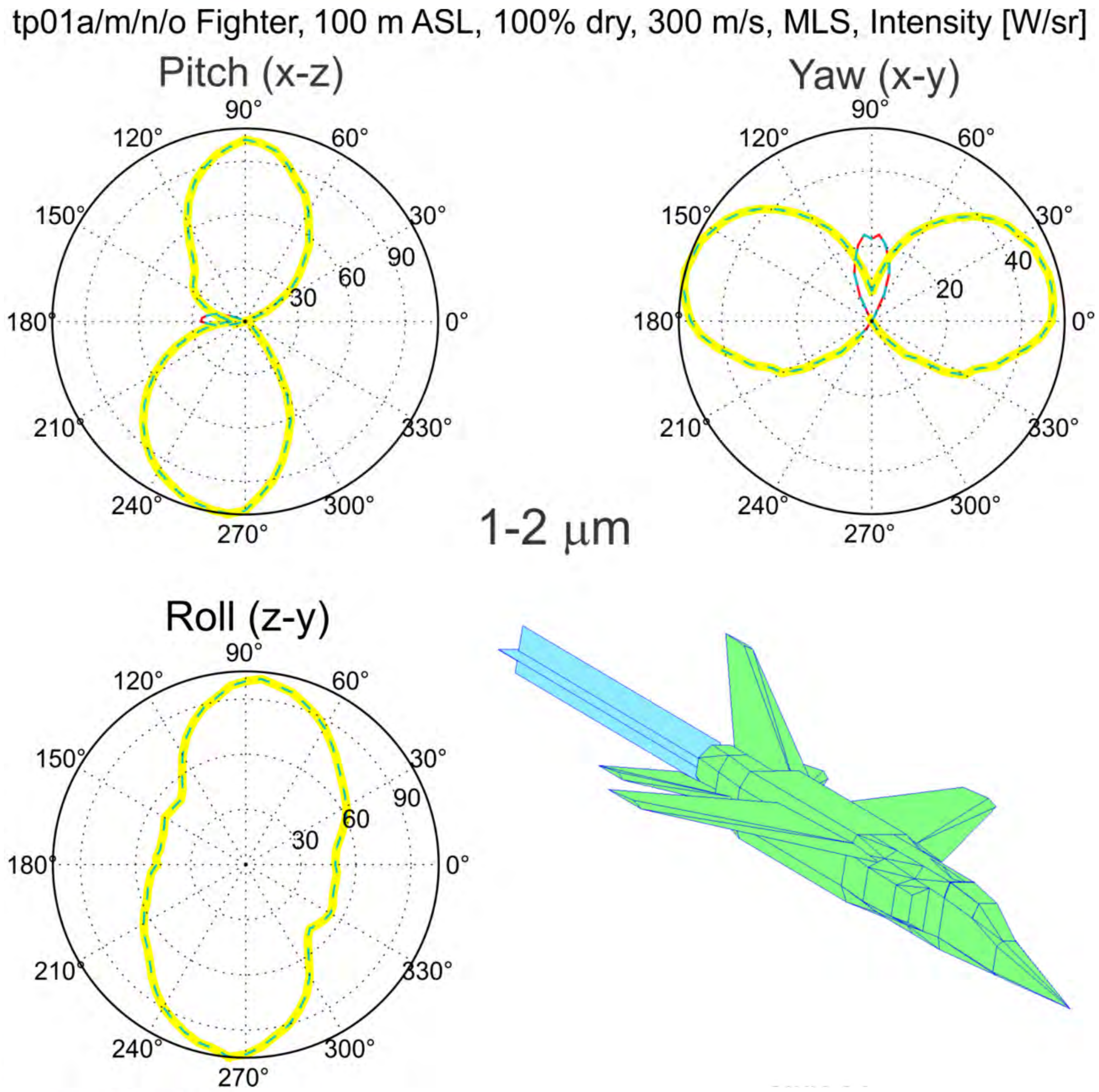


Orbiting sensor:
varying background

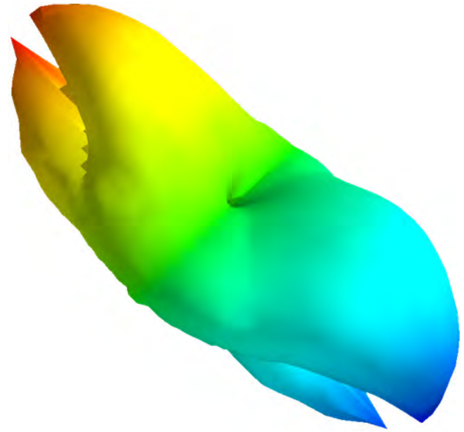
Mario!



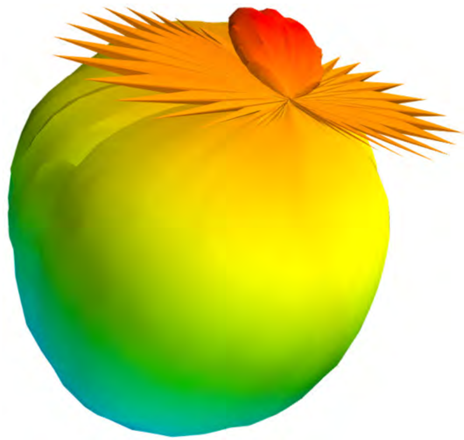
Intensity Polar Plots 1-2 μm



Rotating target:
constant background

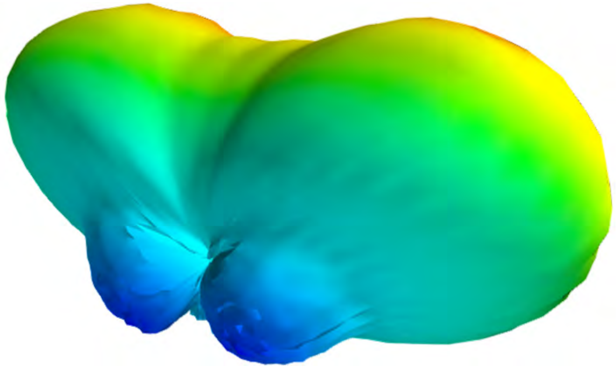


Orbiting sensor:
varying background

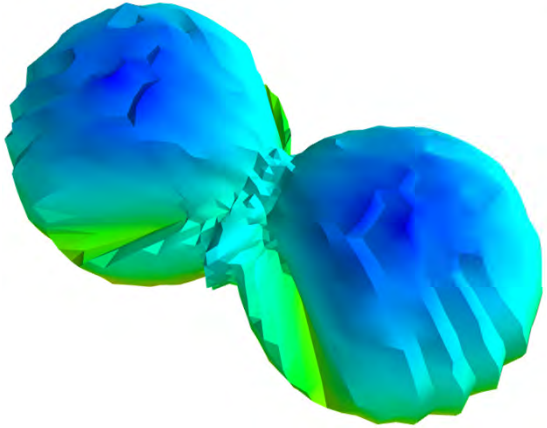


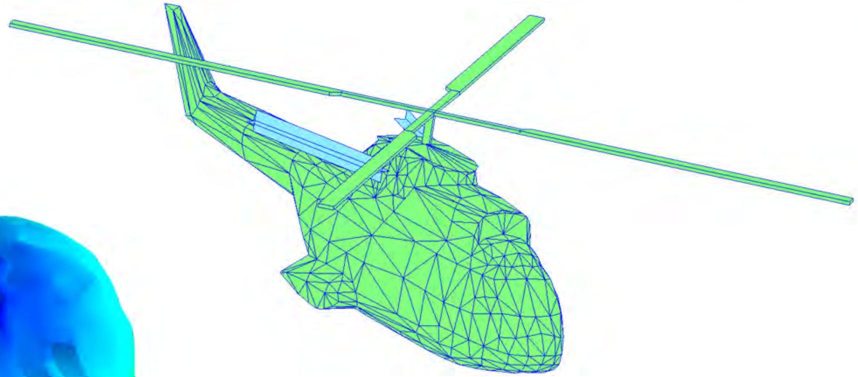
Puma Helicopter Model

8-12 μm



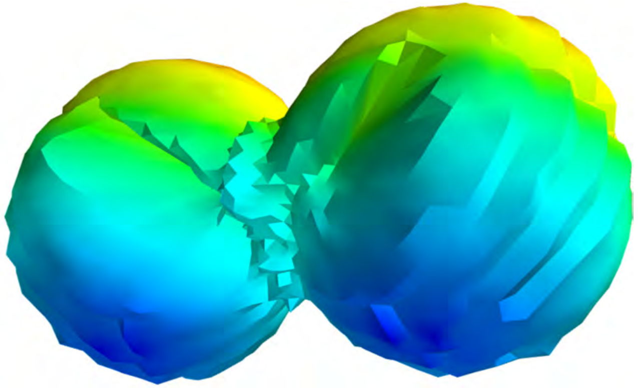
3-5 μm



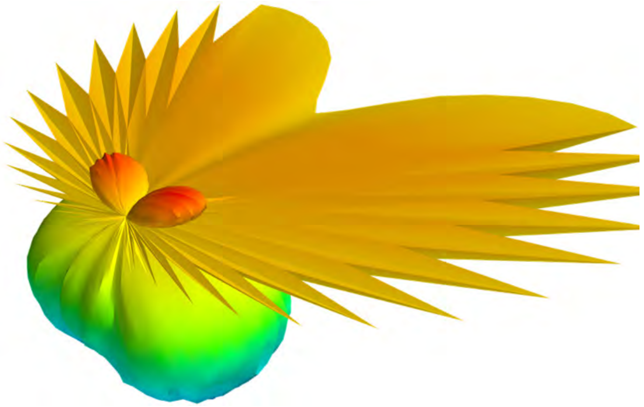


Orbiting sensor:
varying background

1.5-2.5 μm



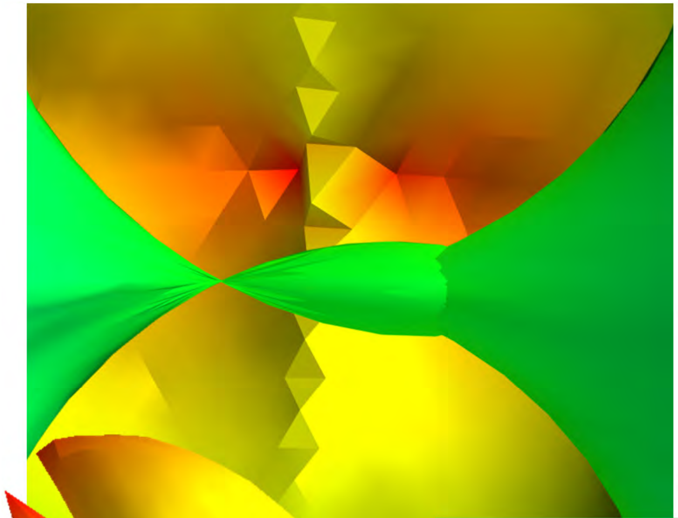
1-2 μm

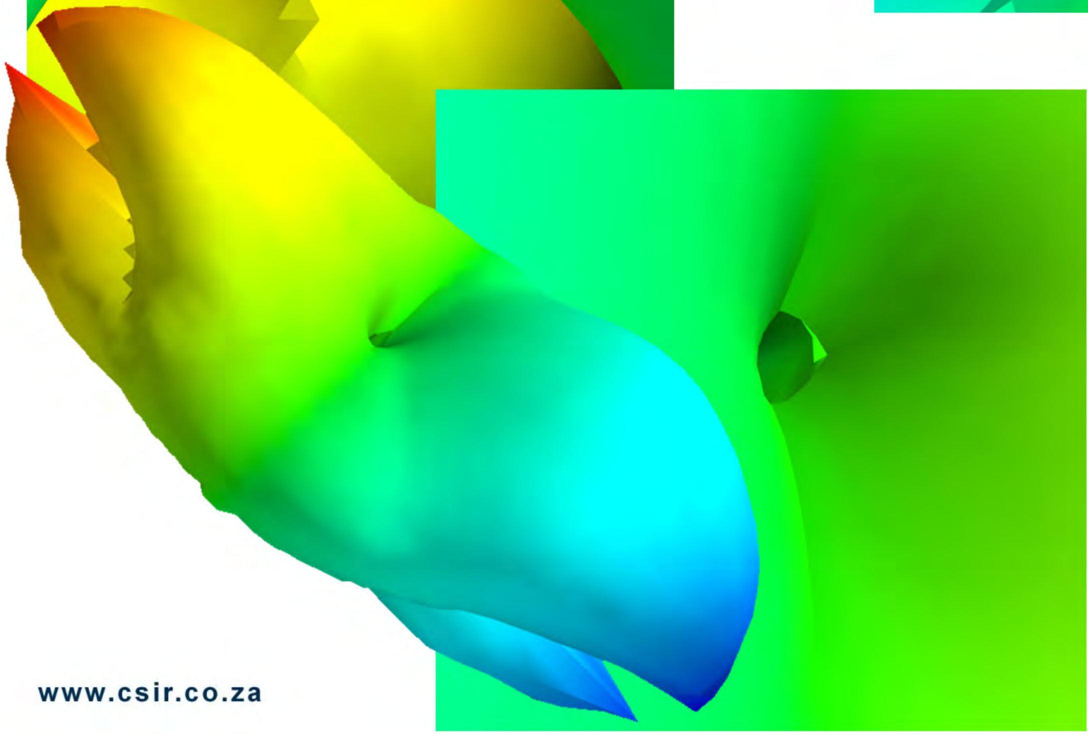


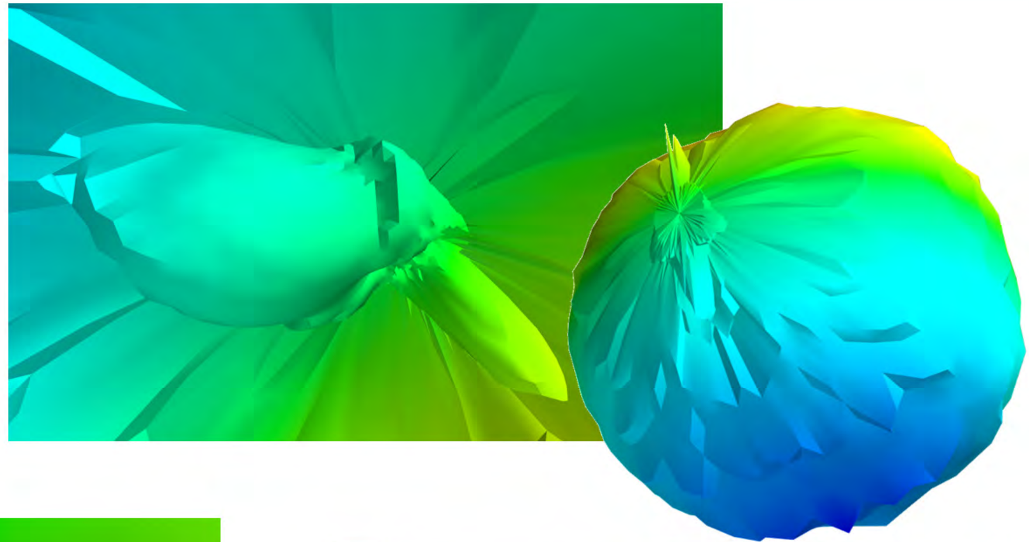
Going Inside!

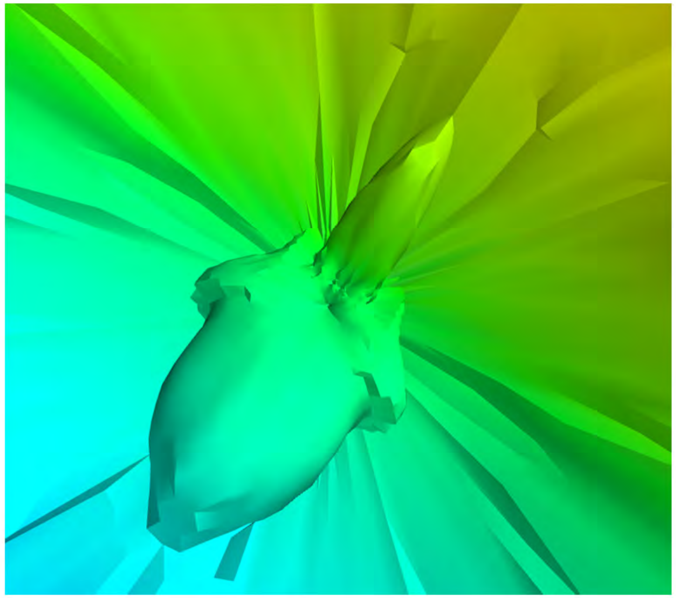
You can actually go inside these shapes!

Going Inside!









Requirements

- Moderate hardware requirements.
- Supported on Linux & Windows.
- Software requirements (all are free):
  - Python 2.7.3.
  - Numpy 1.6.1.
  - Scipy 0.10.1.
  - Matplotlib 1.1.
  - Mayavi 4.1.

our future through science

Authors, Sponsors & Users

- Collaborative effort between
  - CSIR, South Africa.
  - Denel Dynamics, South Africa.
  - Instituto Tecnologico de Aeronautica, Brazil.
- Used in CSIR and Denel for R&D, simulation model development and analysis.
- Targeting academic environment.
- Working towards the tool of choice in infrared laboratories.

Open Source Considerations

Pyradi is available as open source software.

Mozilla Public License 1.1:

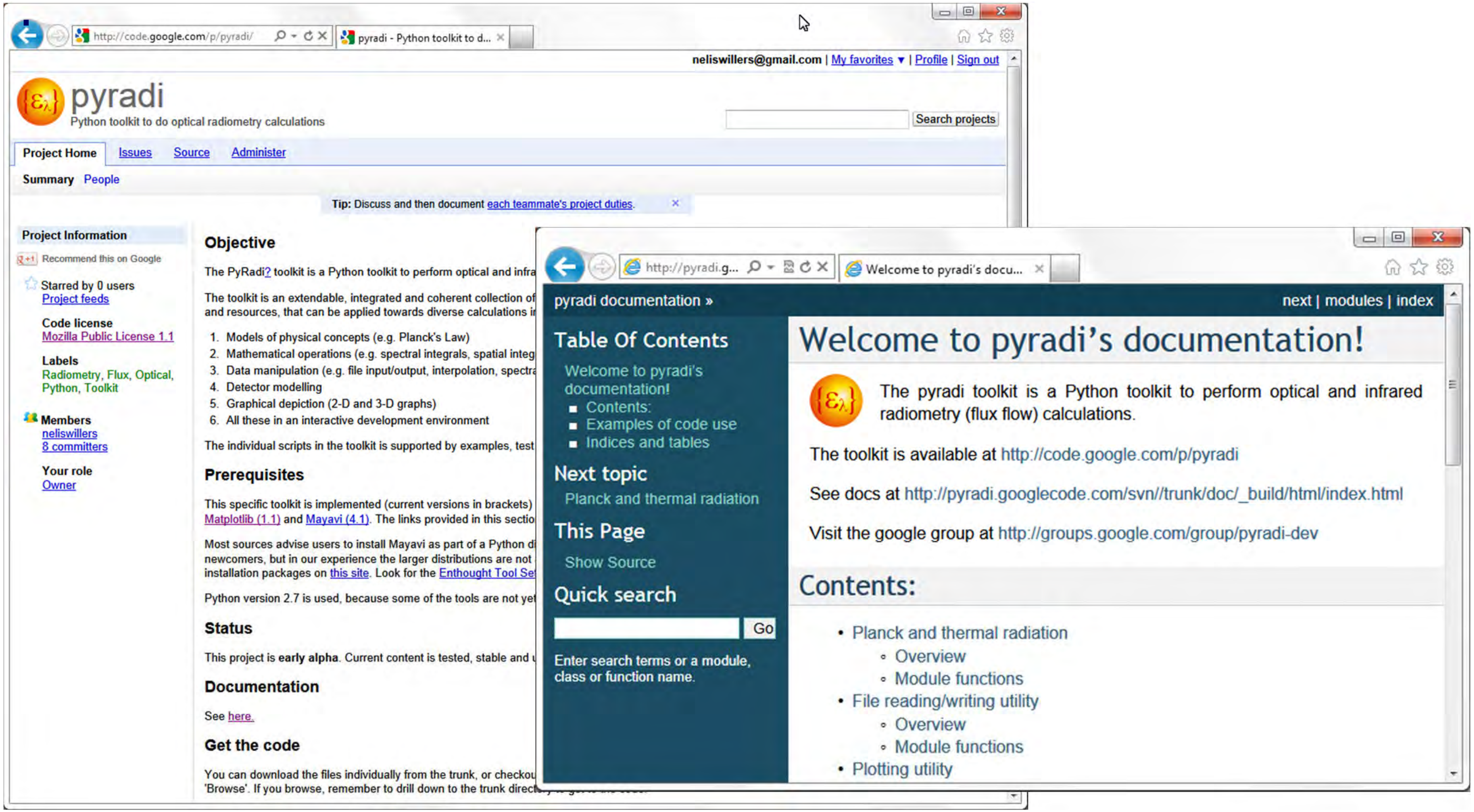
- Authors retain ownership of their respective contributions.
- Modifications by users must be released under compatible terms.

Potential users are encouraged to join in, and commit changes, updates and new contributions.

our future through science

Hosted at Google Code: pyradi

http://code.google.com/p/pyradi
http://pyradi.googlecode.com/svn//trunk/doc/_build/html/index.html



Closing

- Pyradi only recently started – already paying dividends.
- Visualisation proved to be most valuable.
- All are welcome to use and contribute.