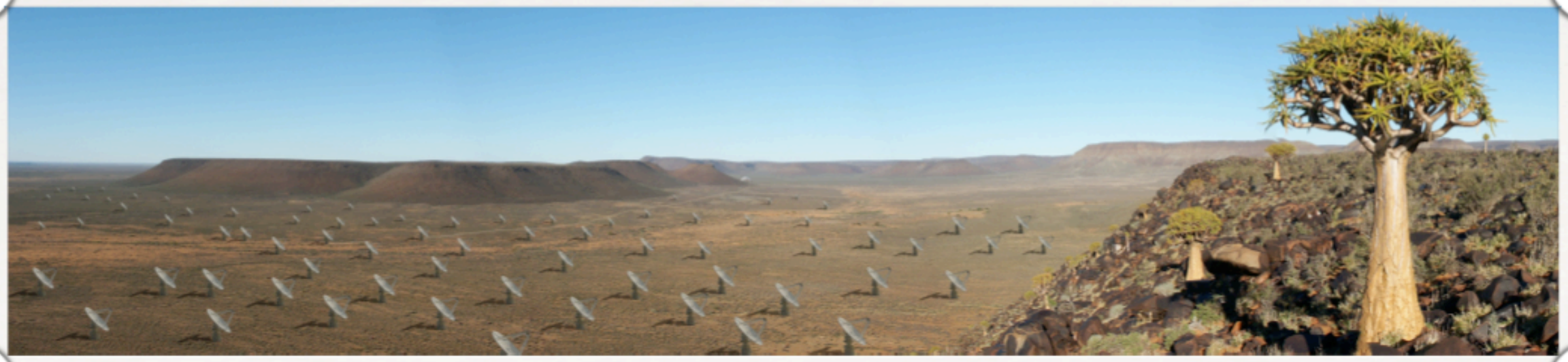


CASA Tutorial

Bradley S. Frank

University of Cape Town
SKA South Africa

This Session

A southern-hemisphere non-blackbelt user's guide to CASA

aka

How I Learned to Stop Worrying and Learned to ~~Love the Bomb~~ Use CASA

aka

A crash-course in CASA usage

CASA Links

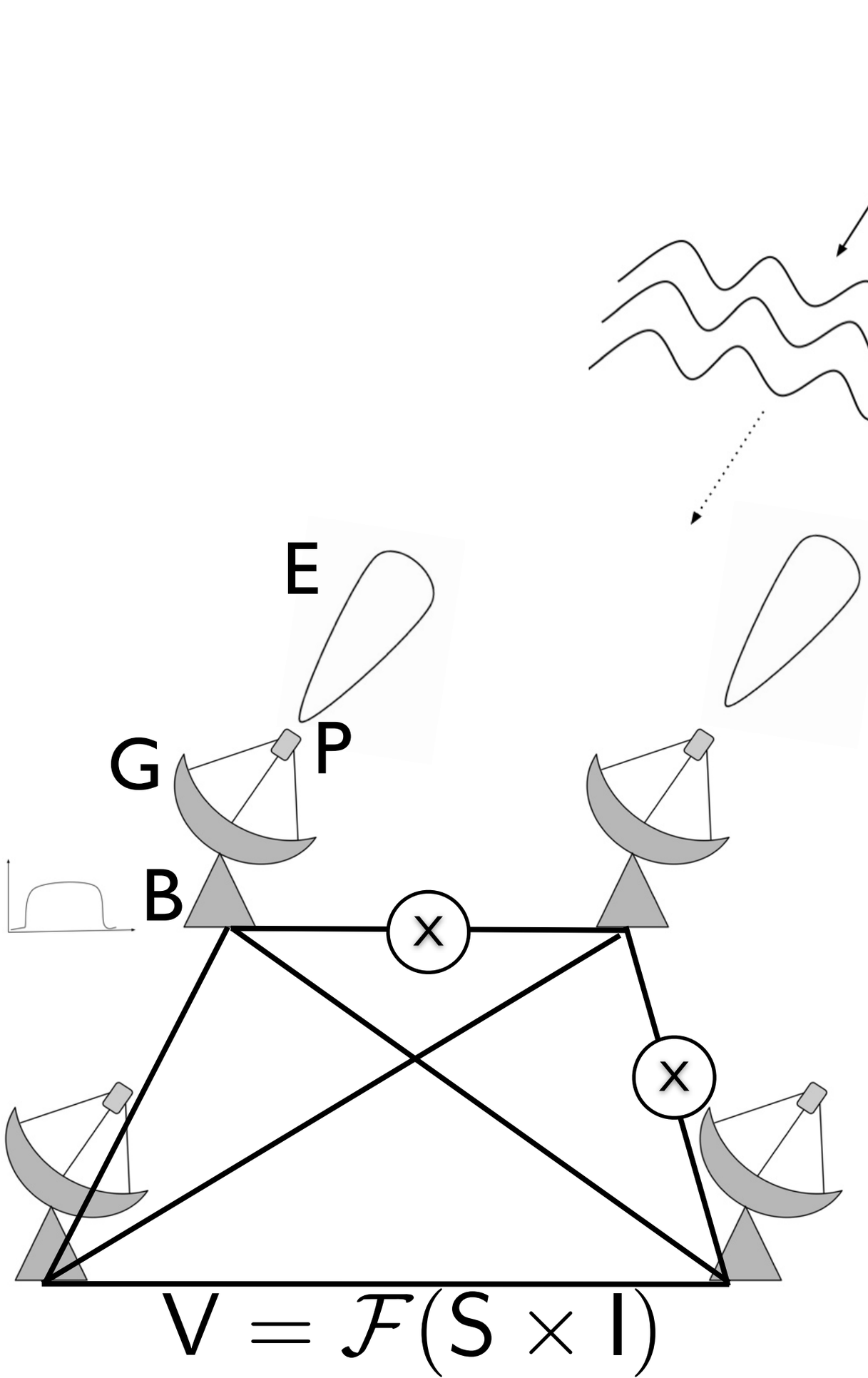
- **CASA Home**: http://casa.nrao.edu
- **CASA Cookbook**: http://casa.nrao.edu/Doc/Cookbook/casa_cookbook.pdf
- **Tutorials and Examples:** http://casaguides.nrao.edu
- **Videos:** http://www.cv.nrao.edu/~jstoke/Demo_Videos/
- **NRAO Summer School:** http://www.aoc.nrao.edu/events/synthesis/2010/

This Tutorial

- Introduction
- Case Study: Calibration: KAT-7, 3C 273
- Simulation Overview
- Your turn: Group Exercises
  - Calibration: KAT-7, 3C 286
  - Simulation: KAT-7, 3C 286: Using L-Band Model
- Discussion

Next/Now?: Group Work

- Available for reference: Pre-calibrated 3C 273: data and scripts
  - /home/frank/3c273.demo/
- Calibration Exercise: KAT-7 Observations
  - /home/frank/3c273.tarbz2
  - /home/frank/3c273_3c286.hh_vv.ms.tar
  - /home/frank/scripts/
- Simulation Exercise: 3C 286
  - Tomorrow?
- Discussion of Results



$$V_{ij} = B_{ij} G_{ij} E_{ij} P_{ij} Z_{ij} V_{ij}^{ideal}$$

Jones	Effect
Z	Ionosphere/ Farady-rotation
E	Primary Beam
P	Parallactic Rotation
G	Electronic Gains
B	Bandpass

Some Background

$$V_{ij} = B_{ij} G_{ij} V_{ij}^{ideal}$$ Simple KAT-7 ME

$$V_{ij} = J_i \beta J_j^H$$ Definition of visibility

$$V = \mathcal{F}(S \times I)$$ Alternative definition

Some Background

$$V_{ij} = B_{ij} G_{ij} V_{ij}^{ideal}$$

Simple KAT-7 ME

$$V_{ij} = J_i \beta J_j^H$$

Definition of visibility

$$V = \mathcal{F}(S \times I)$$

Alternative definition

Visibilities: What we **measure**
and what we calibrate

Some Background

bandpass instrumental gains

$$V_{ij} = B_{ij} G_{ij} V_{ij}^{ideal}$$

Simple KAT-7 ME

$$V_{ij} = J_i \beta J_j^H$$

Definition of visibility

$$V = \mathcal{F}(S \times I)$$

Alternative definition

Jones Matrices: Instrumental/ Environmental corruption terms that we **solve for**

Some Background

$$V_{ij} = B_{ij} G_{ij} V_{ij}^{ideal}$$

Simple KAT-7 ME

$$V_{ij} = J_i \beta J_j^H$$

Definition of visibility

$$V = \mathcal{F}(S \times I)$$

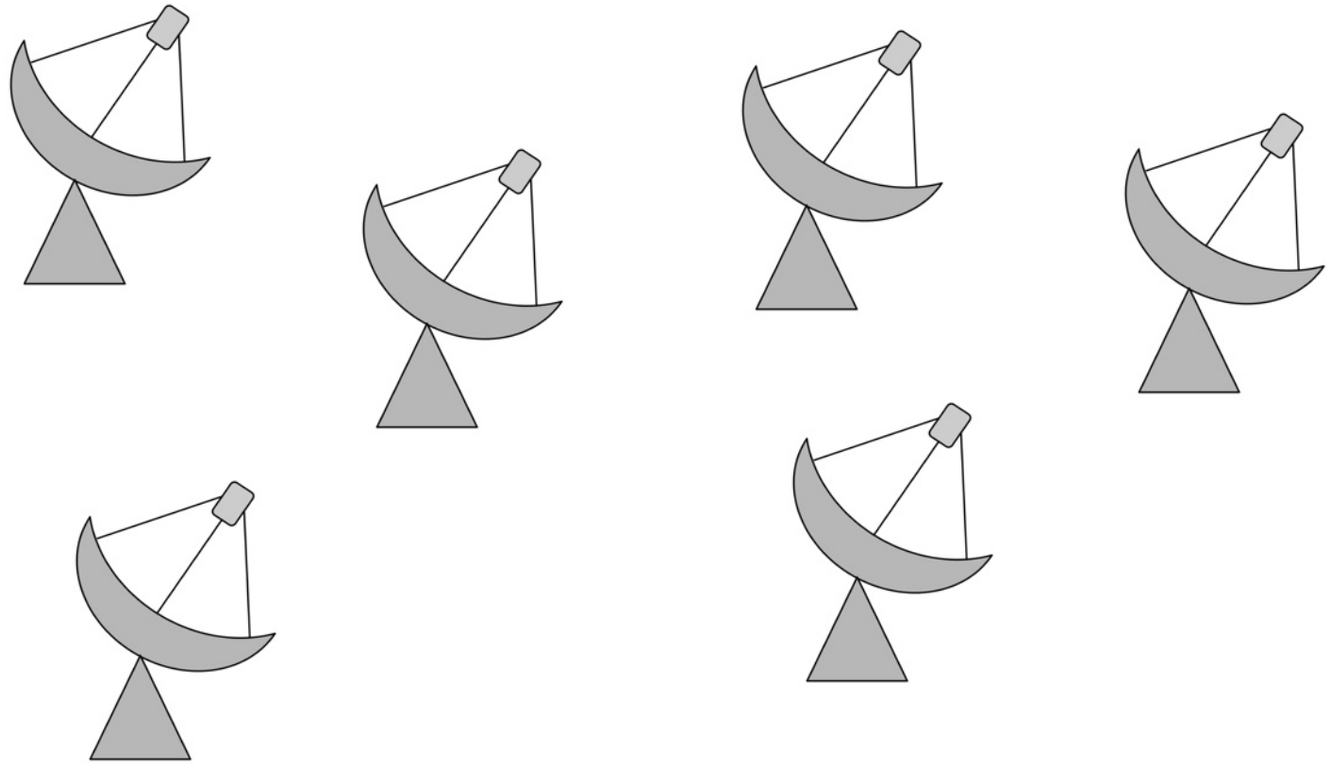
Telescope PSF

Alternative definition

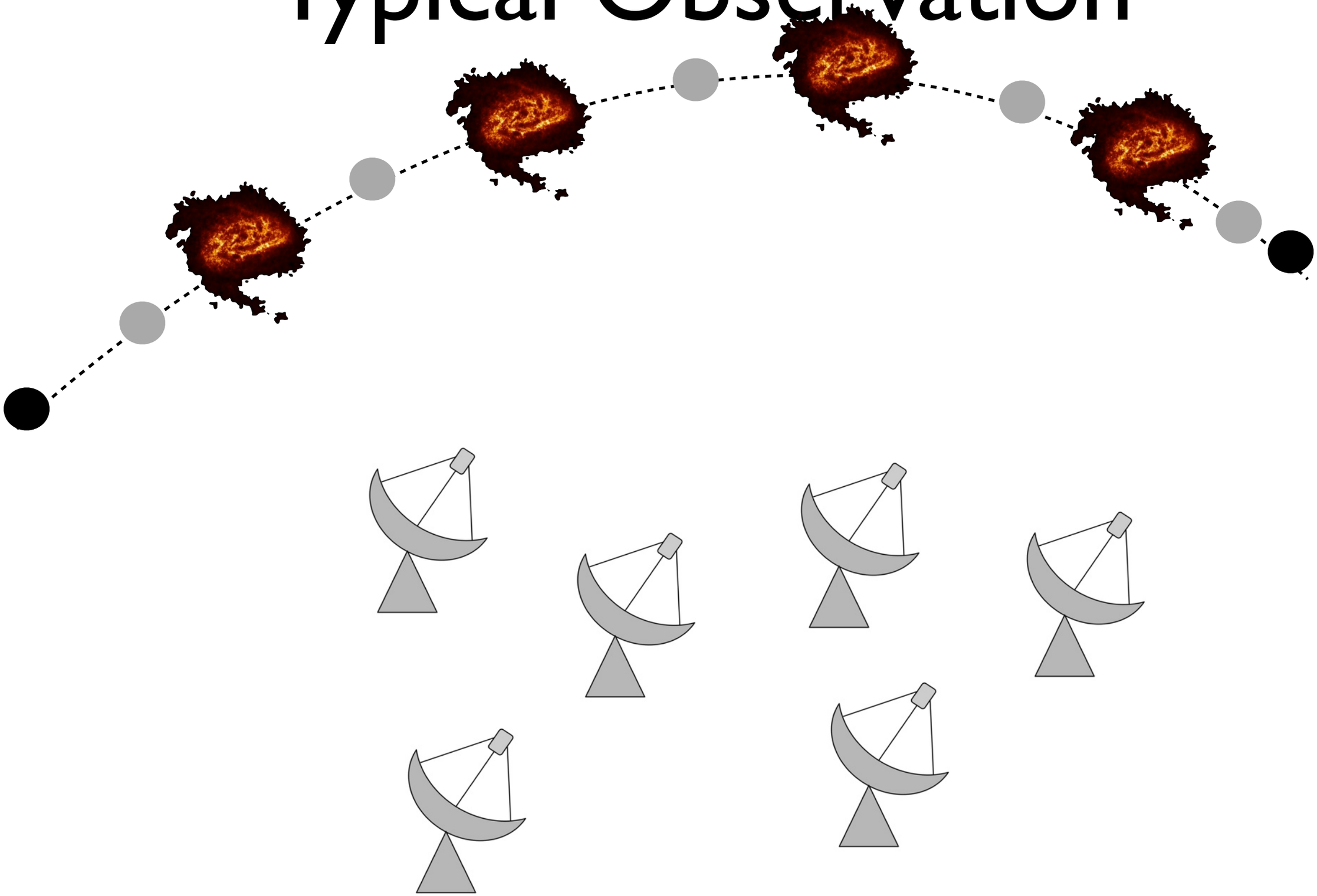
Source Intensity Distribution:

What we want

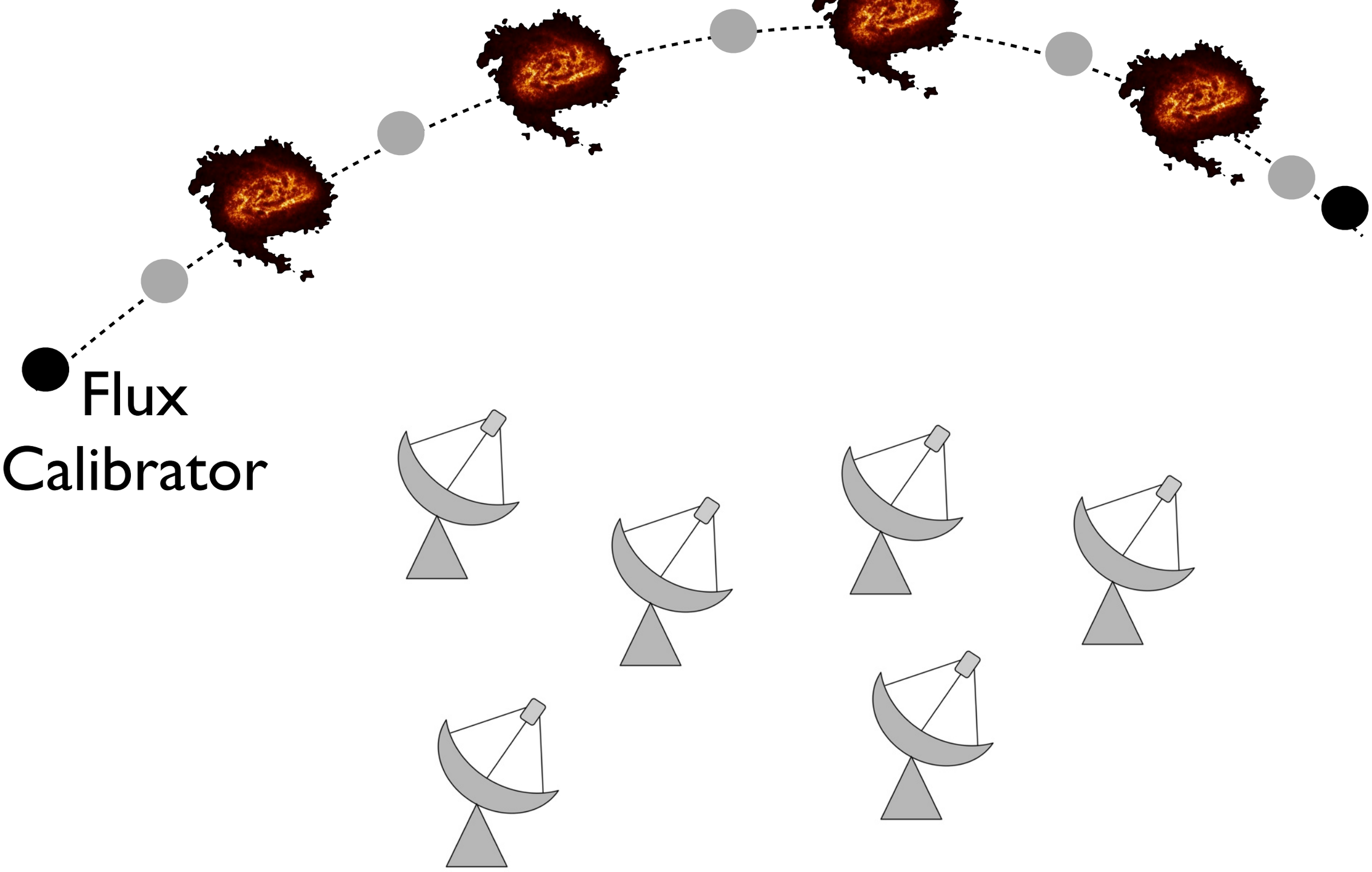
Typical Observation



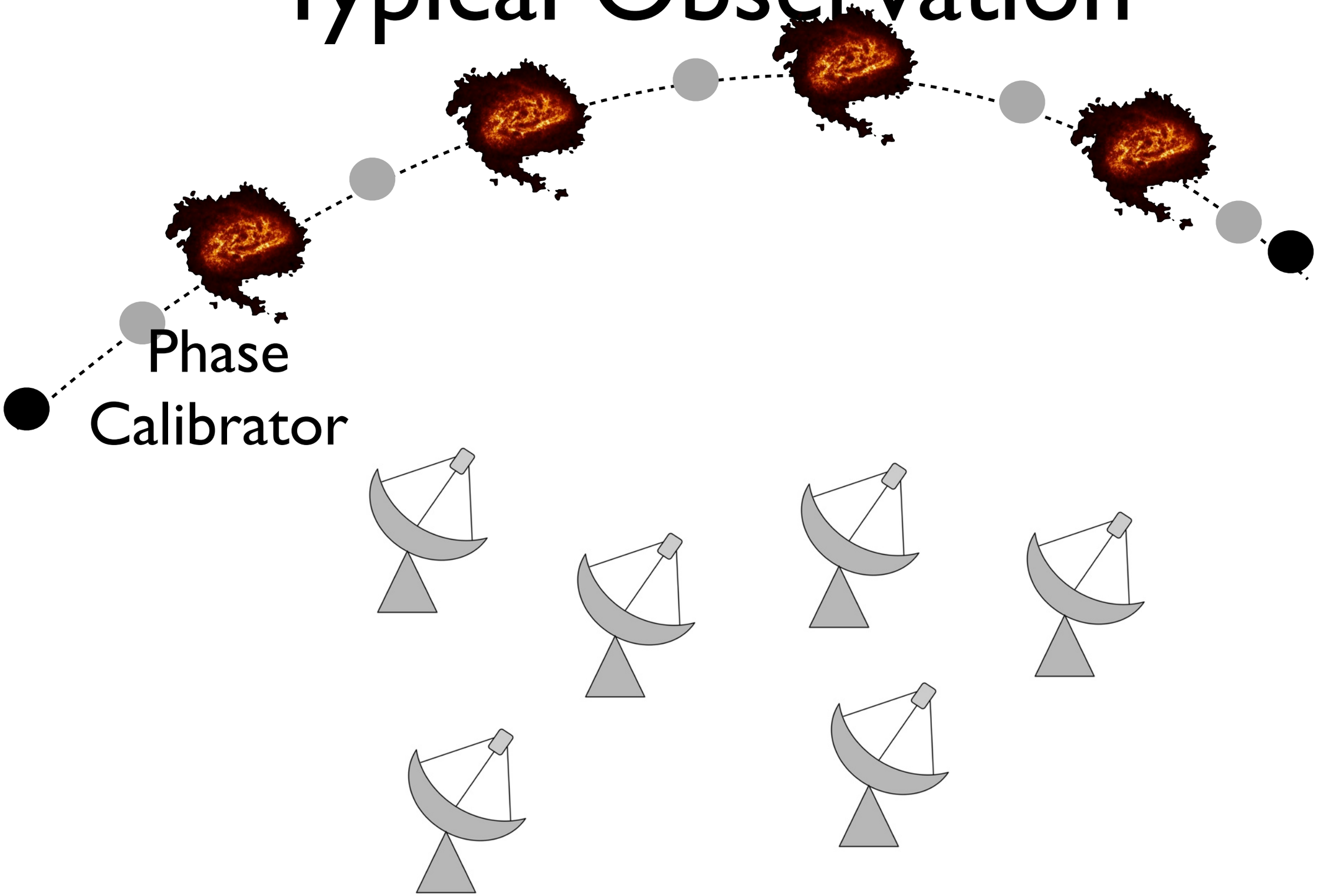
Typical Observation



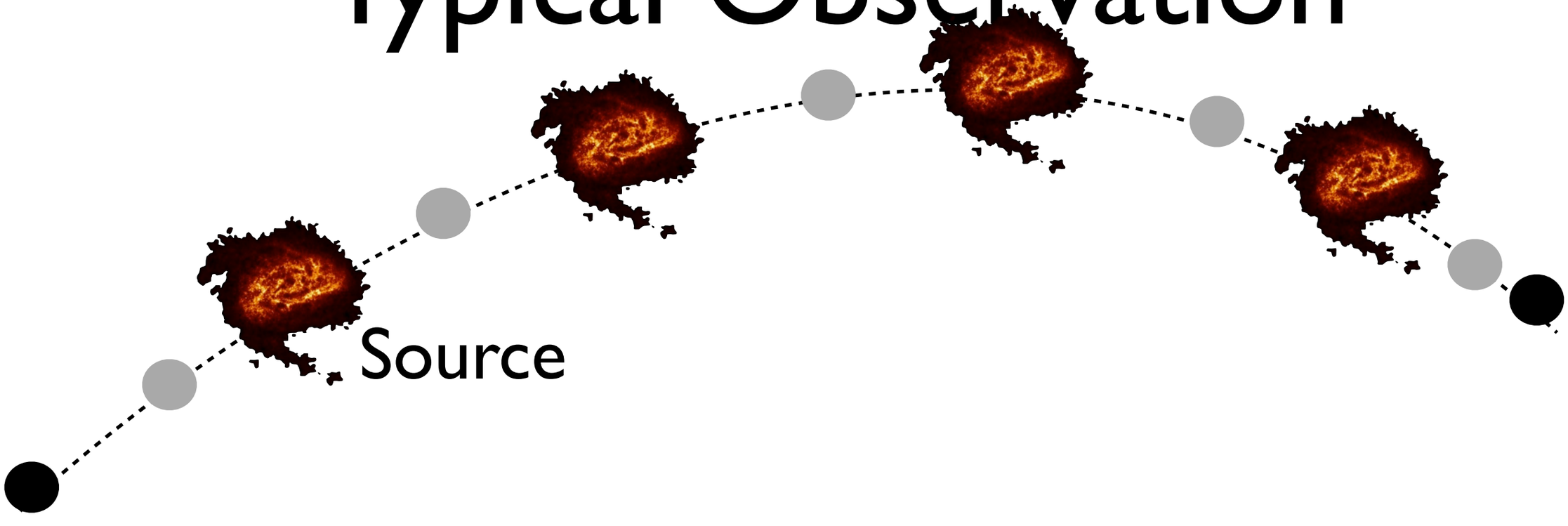
Typical Observation



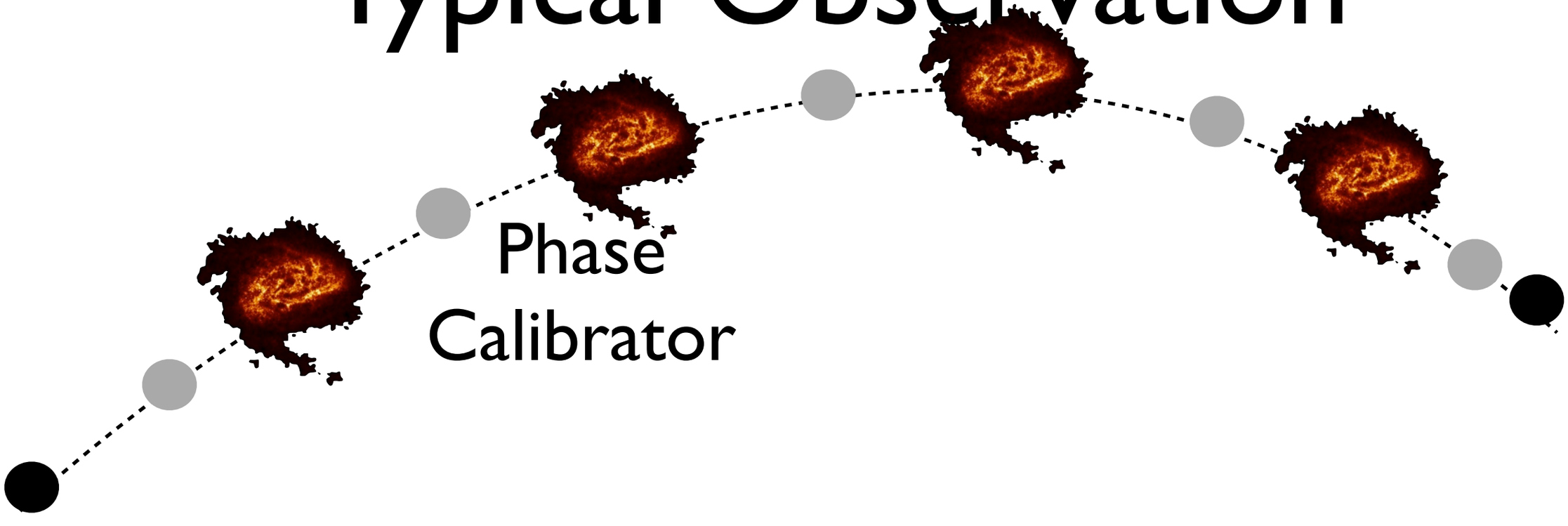
Typical Observation



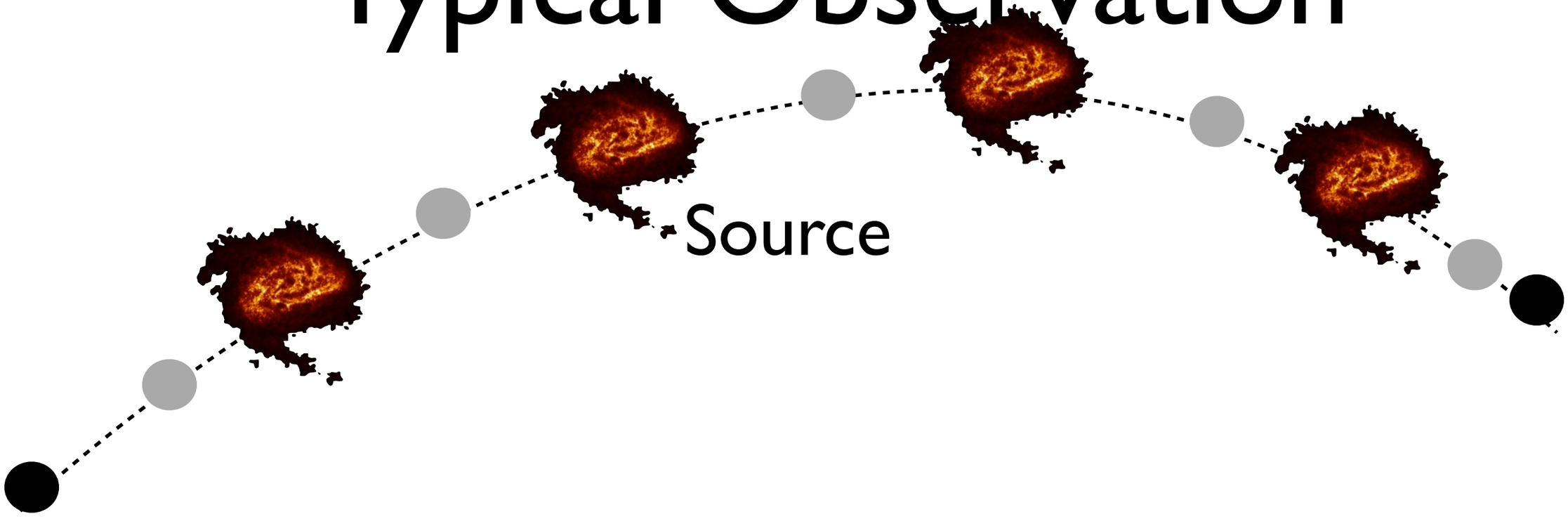
Typical Observation

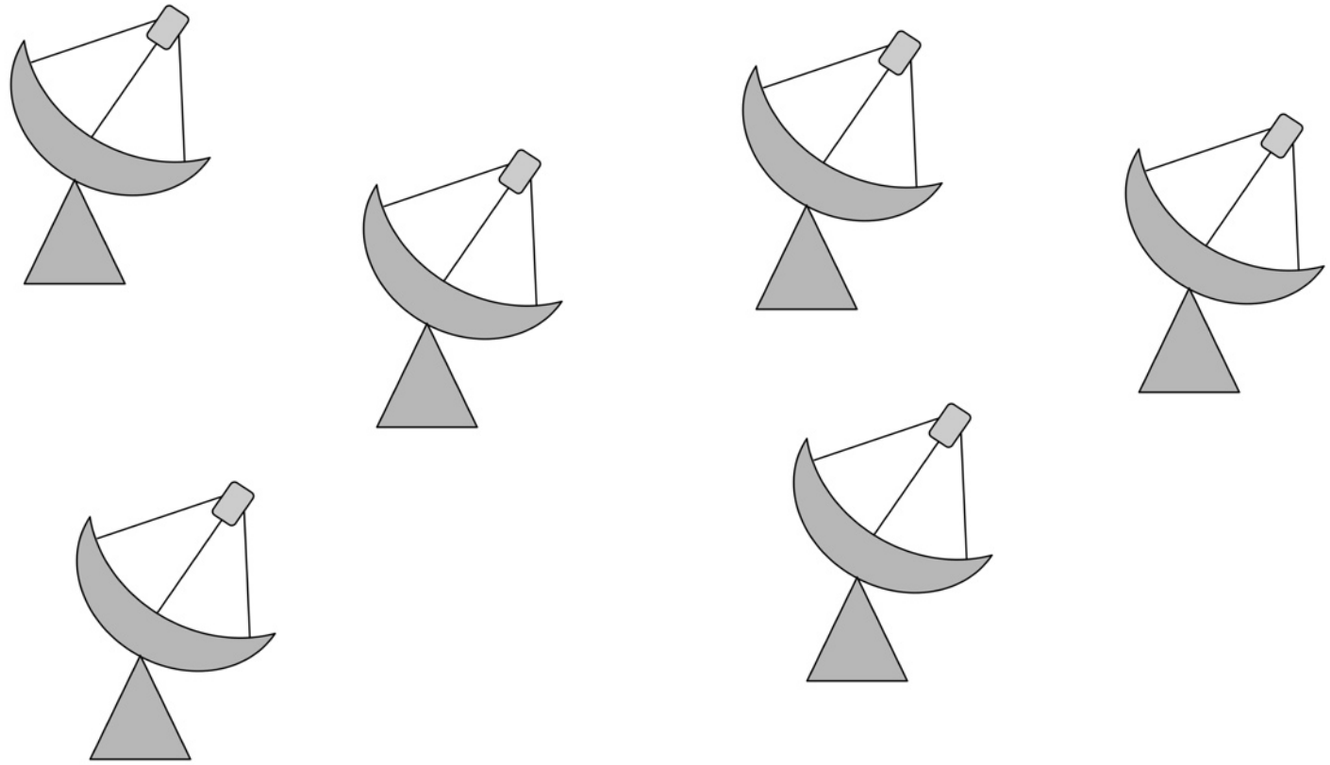


Typical Observation

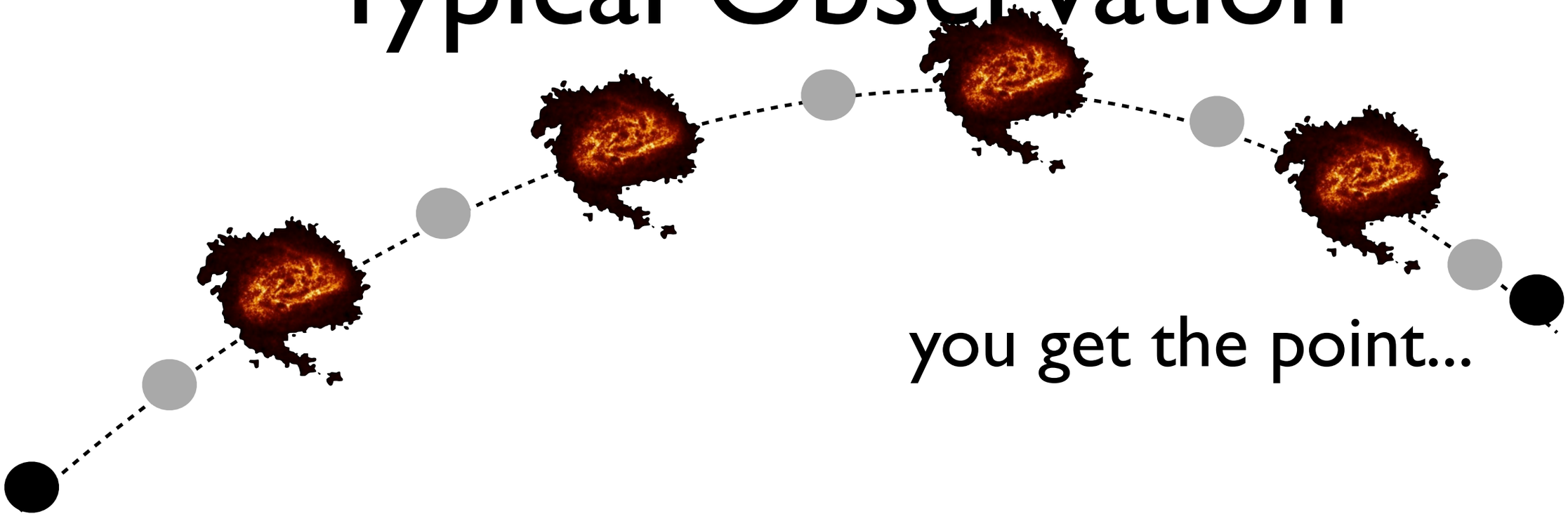


Typical Observation

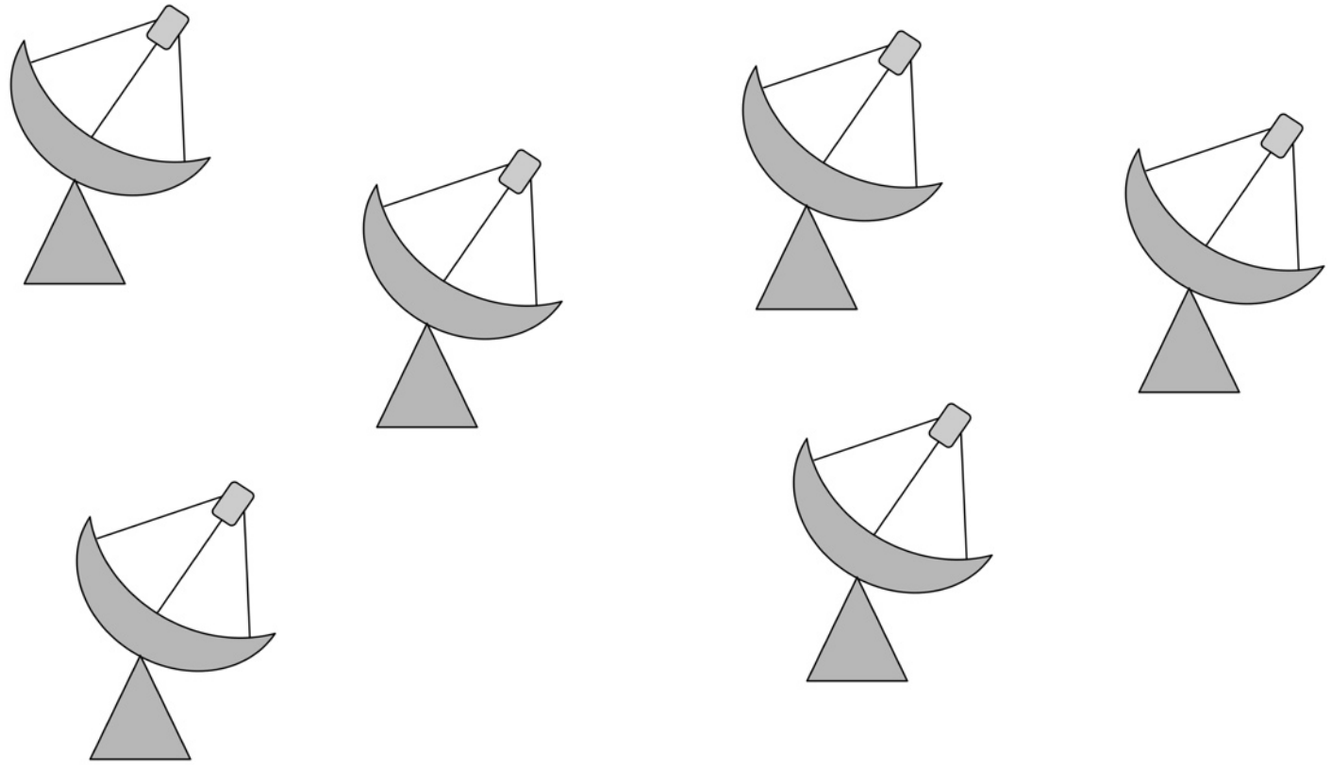




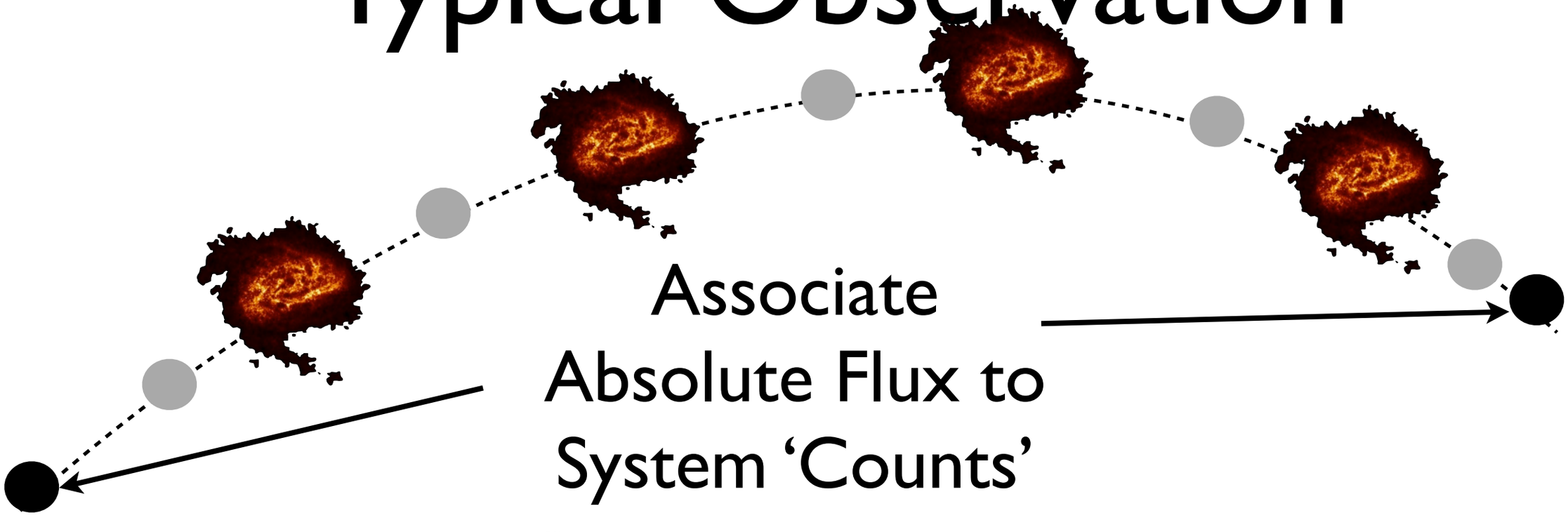
Typical Observation



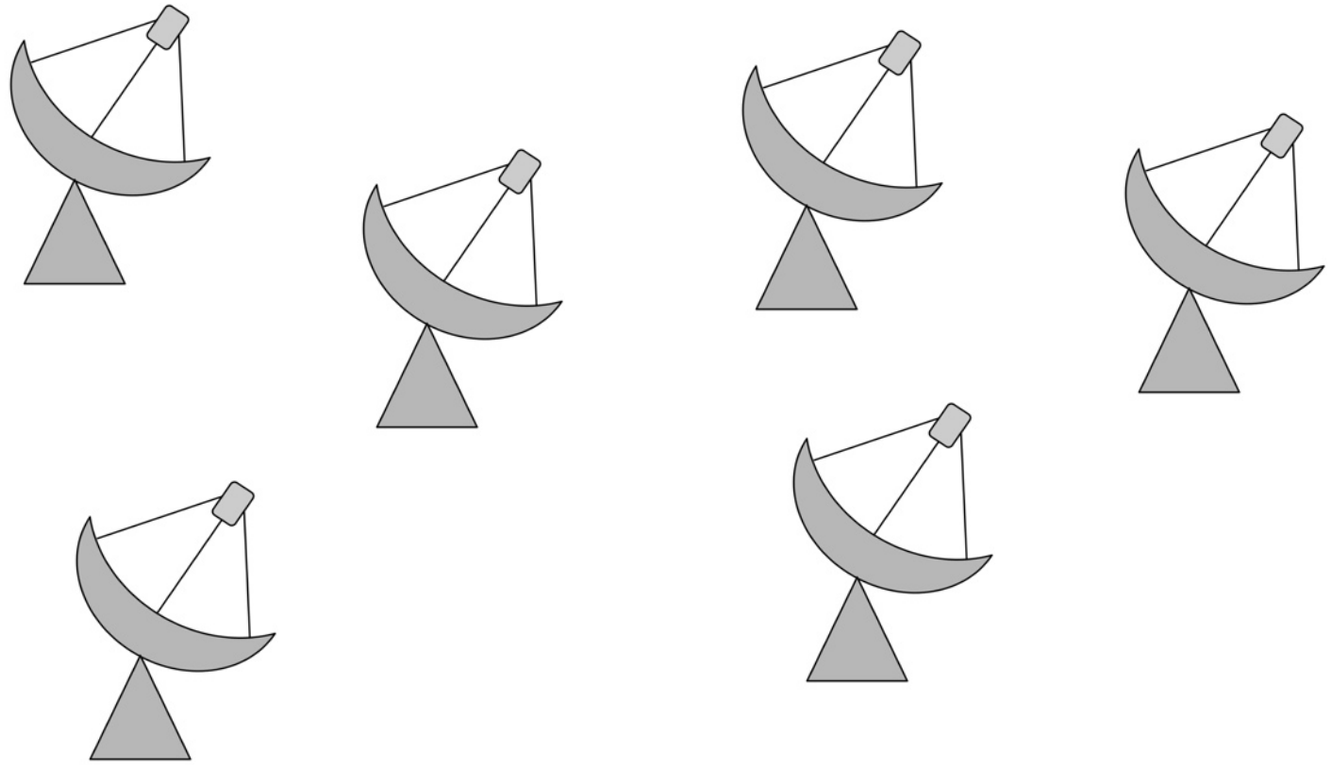
you get the point...



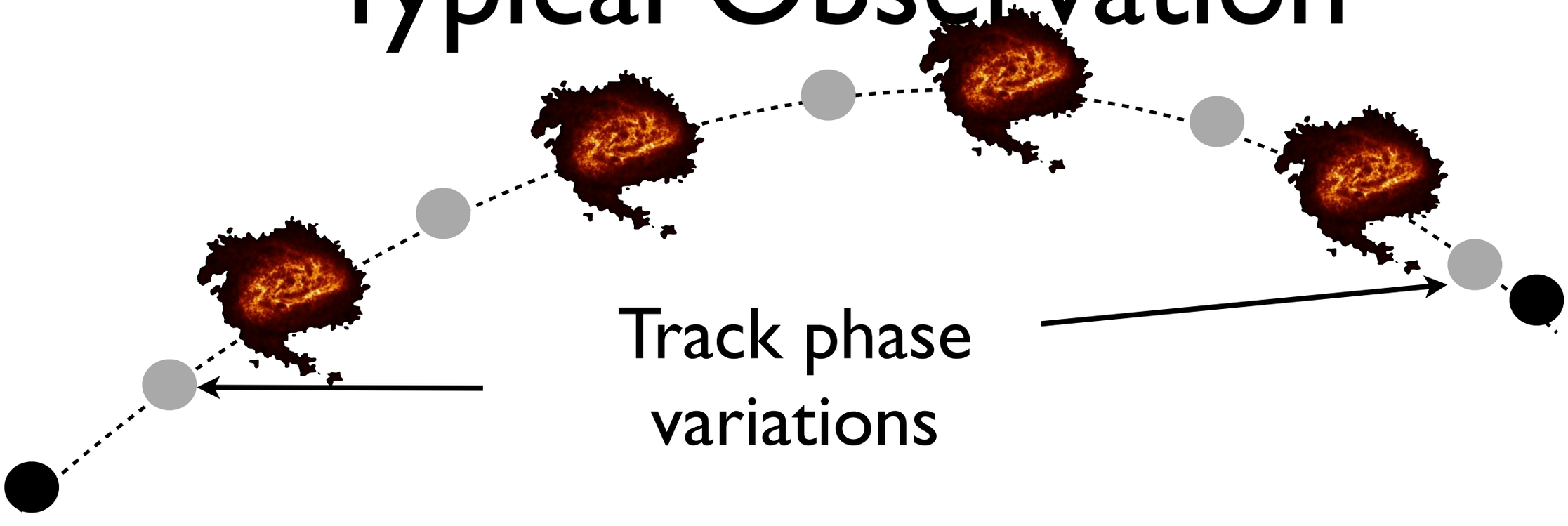
Typical Observation

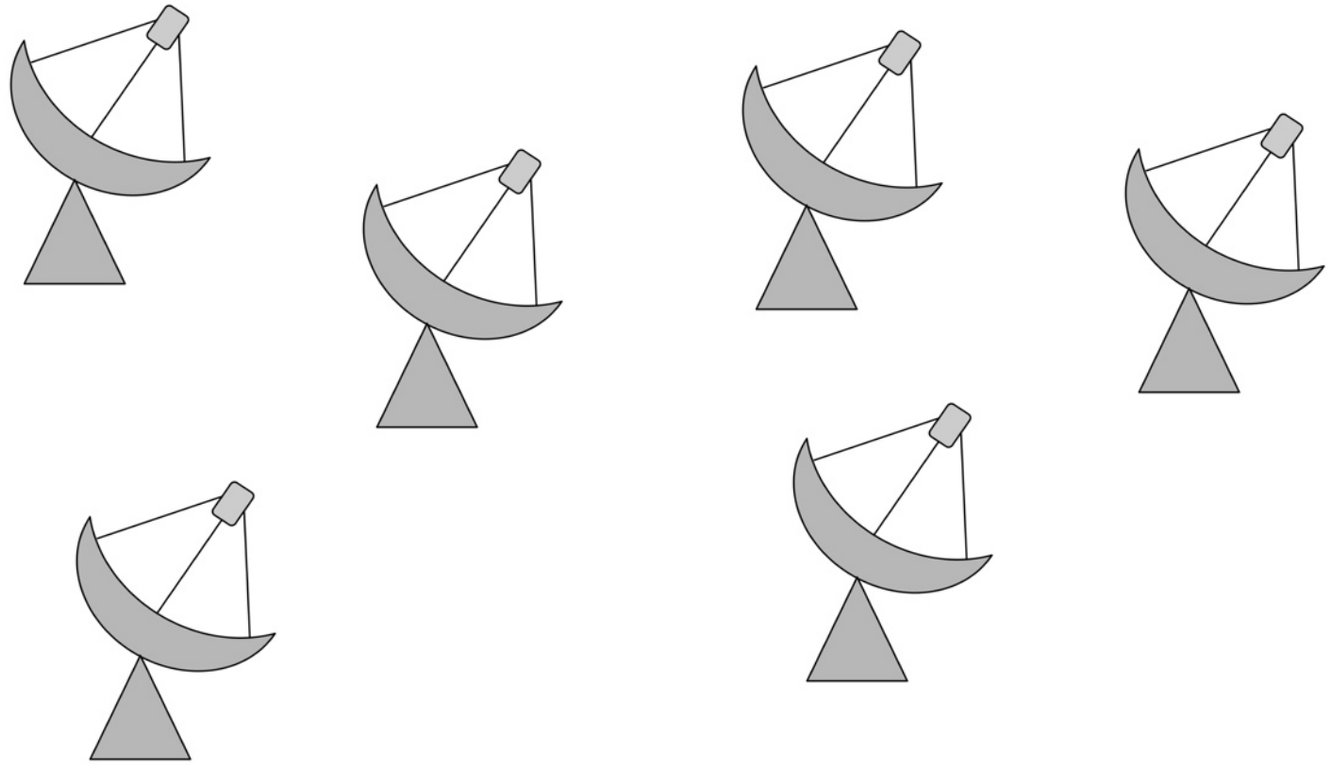


Associate
Absolute Flux to
System ‘Counts’

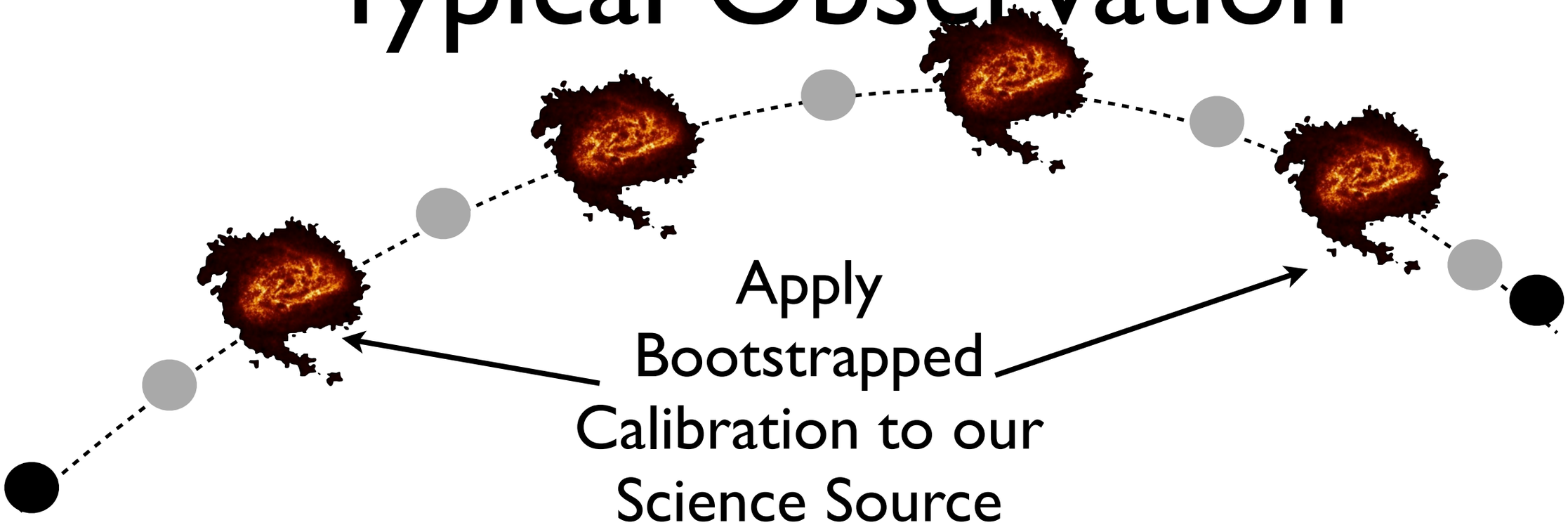


Typical Observation

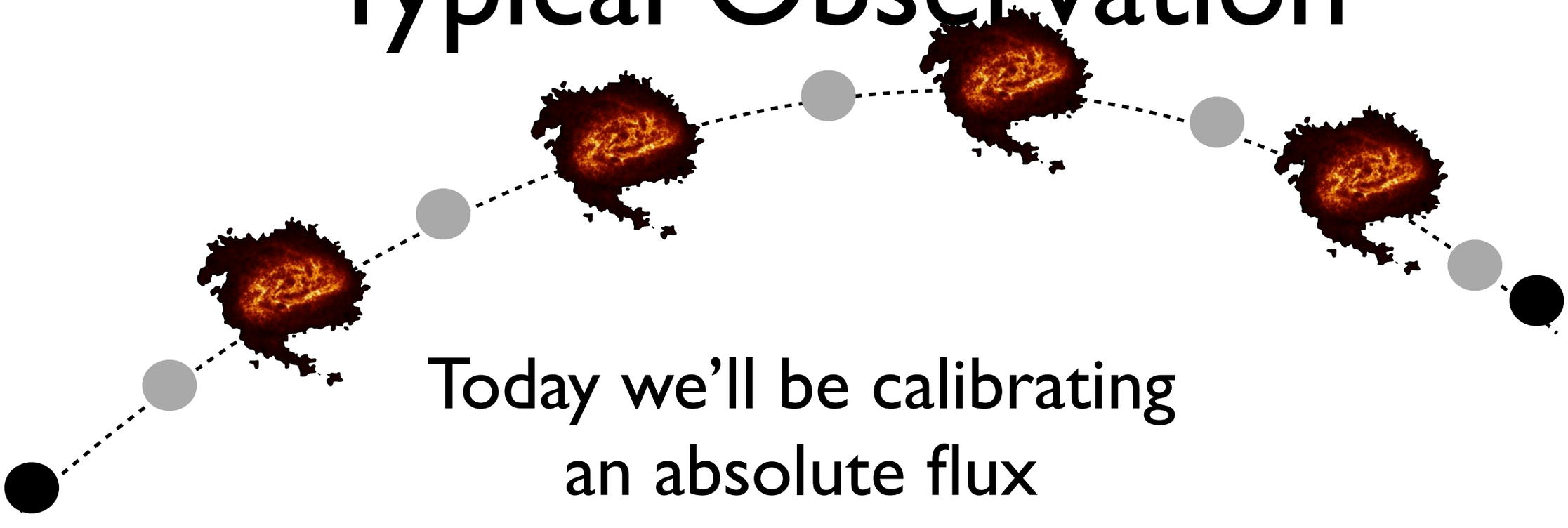




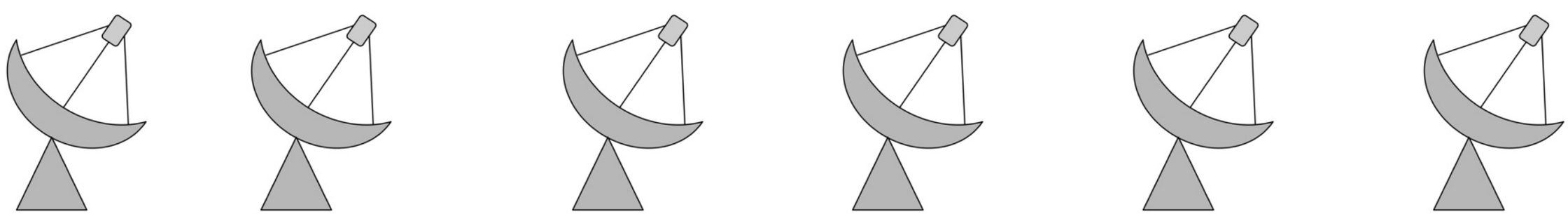
Typical Observation



Typical Observation



Today we'll be calibrating
an absolute flux
calibrator
3C 273



Plan...

View	listobs vishead
Plot	viewer plotms
Flag	flagautocorr plotms flagdata
Calibrate	setjy gaincal bandpass applycal
Image	clean split

Plan...

View	`listobs` `vishead`
Plot	`viewer` `plotms`

Flag	`flagautocorr`
	`plotms`
	`flagdata`

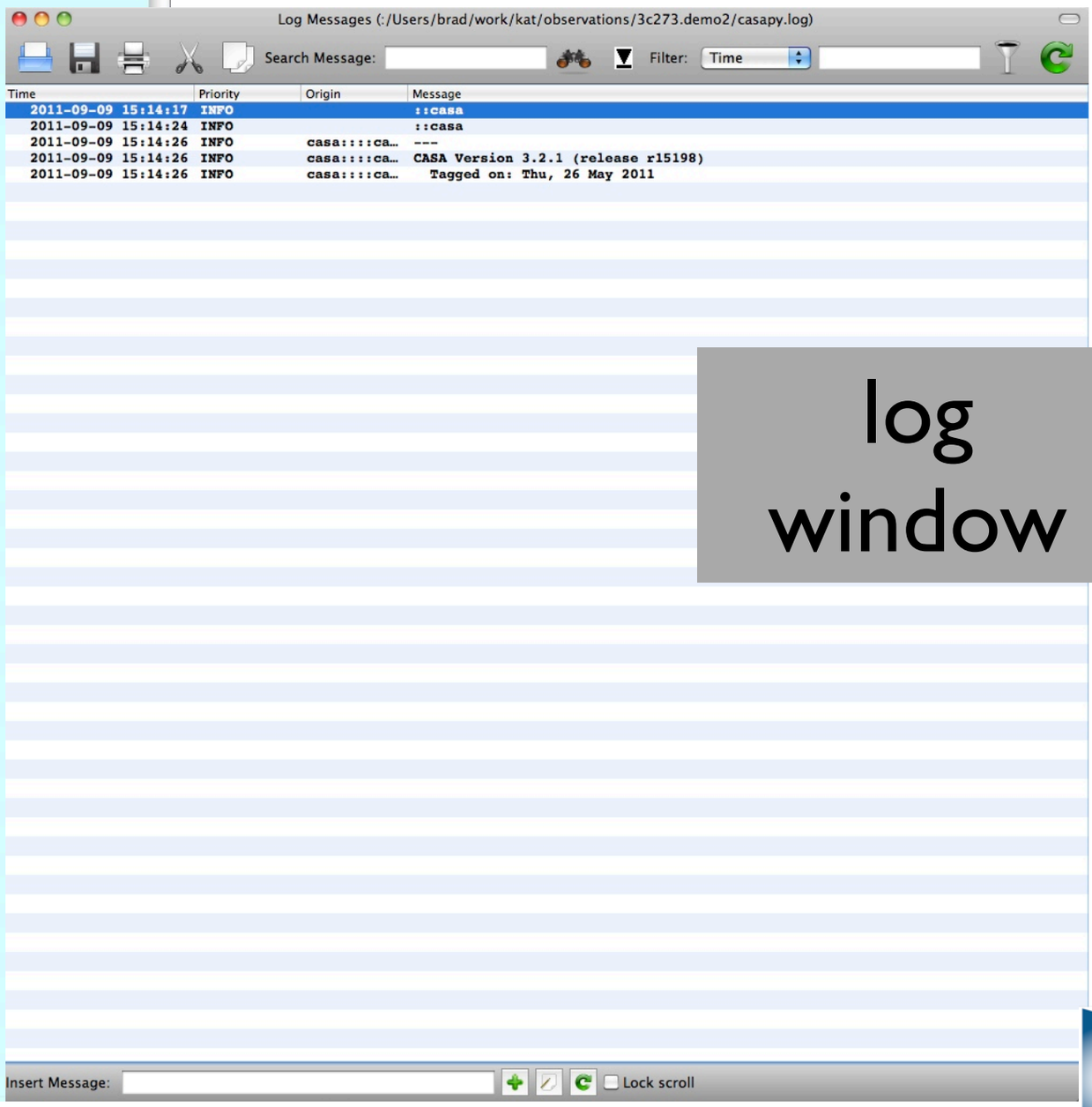
Calibrate	`setjy`
	`gaincal`
	`bandpass`
	`applycal`

Image	`clean` `split`

$$V_{ij} = B_{ij} G_{ij} V_{ij}^{ideal}$$

$$V = \mathcal{F}(S \times I)$$

your CASA environment



Toolboxes and Tasks

- **Toolboxes**

  Midlevel python interface to C++ casacore

  Examples: ia, sm, cal, qu

- **Tasks**

  Python wrapped convenience scripts

  Examples: listobs, gaincal, clean...

Task Talk

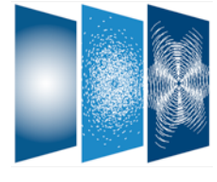
```
CASA <i>:inp(task) # Displays Task
  Inputs

CASA <i>:default(task) # Sets Task
  Inputs to default values

CASA <i>:go(task) # Go Ninja!

CASA <i>:task() # urm... Go,
  Ninja!

CASA <i>:task(args=blah) # Task go
  with specific (useful for
  scripting)
```



Task Convenience Functions

```
CASA <i>:saveinputs(taskname=task,
  outfile='task0.saved')
```

```
CASA <i>:execfile('task0.saved') #
  populate the console with the
  inputs saved in the file
```

CASA automatically saves the settings for the last execution of a task in *task*.last

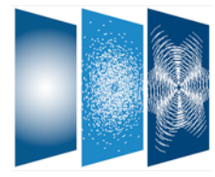
Viewing stuff...

```
CASA <i>:listobs(vis='3c273.ms')
```

Just lists observational information

```
CASA <i>:vishead(vis='3c273.ms')
```

Allows you to view MS header, or to manipulate header fields.



Plotting

```
CASA <i>:viewer(vis='3c273.ms')
```

```
CASA <i>:plotms(vis='3c273.ms',
xaxis='time', yaxis='amp')
```

`plotms()` is memory hungry! Use averaging/per-antenna to minimize RAM overhead

$$V_{ij} = B_{ij} G_{ij} V_{ij}^{ideal}$$

Flagging

```
CASA <i>:plotms(vis='3c273.ms',
  xaxis='time', yaxis='amp')
```

```
CASA <i>:flagdata2(vis='3c273.ms',
  manualflag=True, ...
```

$V_{ij} = B_{ij} G_{ij} V_{ij}^{ideal}$

Flagging: `flagdata2`

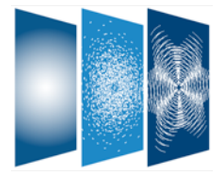
```
CASA <i>:flagdata2(vis='3c273.ms',
  manualflag=True, ...
```

manualflag

quack

Flags are kept in a *.ms.flagversions directory, and maybe accessed via **`flagmanager()`** task

$$V_{ij} = B_{ij} G_{ij} V_{ij}^{ideal}$$



setjy

- Places model visibilities in the model column of measurement set.
- For 3C48, 3C138, 3C286 & 3C147 there are standard model images.
- For custom calibrators (as is for 3C273), we can specify the point-source flux as a stokes [I,Q,U,V] vector:

```
CASA <i>: setjy(vis='3c273.ms',
  fluxdensity=[42.0, 0, 0, 0])
```

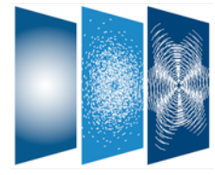
$V_{ij} = B_{ij} G_{ij} V_{ij}^{ideal}$

G-Jones

- Want to track gains over a suitable "coherence time" (i.e. as long as we expect gains to be stable): `solint='30s'`
- Use a good reference antenna: `refant='ant3'`

```
CASA <i>: gaincal(vis='3c273.ms',
  caltable='3c273.G0', solint='30s',
  gaintype='G', calmode='ap')
```

$V_{ij} = B_{ij} G_{ij} V_{ij}^{ideal}$

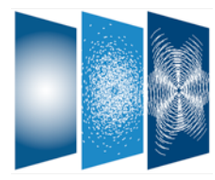


B-Jones

- Expect bandpass to be constant over each scan: `solint='inf', combine='scan'`
- Use a good reference antenna: `refant='ant3'`
- Now we "apply" G-Jones on-the-fly: `gaintable=['3c273.G0']`

```
CASA <i>: bandpass(vis='3c273.ms',
  caltable='3c273.B0', solint='inf',
  combine='scan', refant='ant3',
  bandtype='B', gaintable=['3c273.G0'])
```

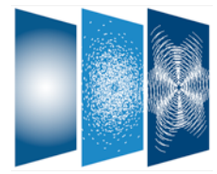
$V_{ij} = B_{ij} G_{ij} V_{ij}^{ideal}$



Apply Calibration

```
CASA <i>: applycal(vis='3c273.ms',
  gaintable=['3c273.G0', '3c273.B0'])
```

$$V_{ij} = B_{ij} G_{ij} V_{ij}^{ideal}$$



Split

- `Clean` doesn't support custom column imaging (we have data, corrected and model visibilities in the MS)
- We have to create another MS with the corrected data.

```
CASA <i>: split(vis='3c273.ms',
  outputvis='3c273_corrected.ms')
```

$V_{ij} = B_{ij} G_{ij} V_{ij}^{ideal}$

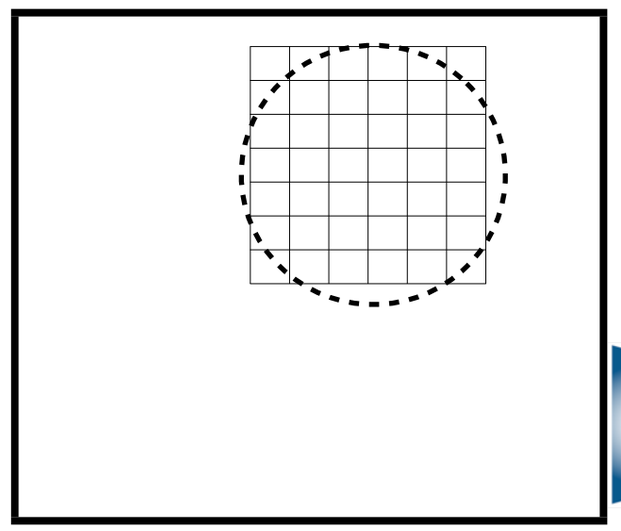
Imaging

- Inverse Fourier Transform of Visibilities to get dirty image: $S \times I$
- Deconvolve PSF S from dirty image to get I

```
CASA <i>: clean(vis='3c273_corrected.ms';
imagename='3c273_corrected.clean';
mode='mfs'; niter=5; psfmode='clark';
imsize=[512; 512]; cell='30.0arcsec';
threshold='0.9mJy');
```

$$\theta \approx \frac{\lambda}{D} \approx 3'$$

$$\sigma = \frac{\sqrt{2}kT_{sys}}{\epsilon_A A \sqrt{N(N-1)\Delta\nu\Delta t}}$$



$$V = \mathcal{F}(S \times I)$$

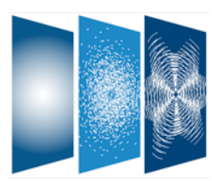
Imaging

clean products:

- Approximate Primary Beam: **3c273.clean.flux**
- Clean component model: **3c273.clean.model**
- PSF used for deconvolution: **3c273.clean.psf**
- Final cleaned image: **3c273.clean.image**
- Difference between dirty and cleaned images: **3c273.clean.residual**

Use the **`ia`** toolbox for image calcs/statistics, or the **`imstat`** task

$$V = \mathcal{F}(S \times I)$$



Simulation in CASA

- CASA task: `simdata()`
- `sm` simulation toolbox: make your own simulator

```
CASA <17>: sm.
sm.__class__          sm.__str__            sm.setapply           sm.setoptions
sm.__delattr__        sm.__subclasshook__   sm.setauto            sm.setpa
sm.__doc__            sm.close              sm.setbandpass        sm.setpointingerror
sm.__format__         sm.corrupt            sm.setconfig          sm.setseed
sm.__getattribute__   sm.done               sm.setdata            sm.setspwindow
sm.__hash__           sm.name               sm.setfeed            sm.settimes
sm.__init__           sm.observe            sm.setfield           sm.settrop
sm.__new__            sm.observemany        sm.setgain            sm.setvp
sm.__reduce__         sm.oldsetnoise        sm.setknownconfig     sm.summary
sm.__reduce_ex__      sm.open               sm.setleakage         sm.type
sm.__repr__           sm.openfromms         sm.setlimits
sm.__setattr__        sm.predict            sm.setmosaicfield
sm.__sizeof__         sm.reset              sm.setnoise
```

CASA Simulation

- http://casaguides.nrao.edu
- Corruption Guide
  - https://safe.nrao.edu/wiki/pub/ALMA/SimulatorCookbook/corruptguide.pdf
- https://safe.nrao.edu/wiki/bin/view/ALMA/Jan2010Wkshop
- Remy Indebetouw's presentations: http://www.aoc.nrao.edu/events/synthesis/2010/
- Ian Heywood ;)

`simdata` overview

- **Inputs**
  - Model Sky
  - Configuration
- **Outputs**
  - Visibilities in a MS
  - Cleaned image (if you want)
  - Some analysis

Terminal — Python — 94×53

Python | bash

```
CASA <2>: inp(simdata)
#  simdata :: mosaic simulation task:
project             =      'sim'        #  root prefix for output file names
modifymodel         =      False        #  modify model image
     skymodel       = '$project.skymodel' #  model image to observe or modify

setpointings        =       True
     integration    =      '10s'        #  integration (sampling) time
     direction      =        ''         #  "J2000 19h00m00 -40d00m00" or "" to center on model
     mapsize        =   ['', '']        #  angular size of map or "" to cover model
     maptype        = 'hexagonal'       #  hexagonal, square, etc
     pointingspacing =        ''        #  spacing in between pointings or "0.25PB" or "" for
                                        #   0.5 PB

predict             =       True        #  calculate visibilites using ptgfile
     complist       =         ''        #  optional componentlist to observe with skymodel
     compwidth      =     '2GHz'        #  optional bandwidth if simulating from components
                                        #   only
     antennalist    = 'alma.out10.cfg'  #  antenna position file or "" for no interferometric
                                        #   MS
     refdate        = '2012/05/21/22:05:00' #  time/date of observation *see help
     totaltime      =    '7200s'        #  total time of observation
     caldirection   =         ''        #  pt source calibrator [experimental]
     calflux        =      '1Jy'

     sdantlist      =         ''        #  single dish antenna position file or "" for no
                                        #   total power MS
     sdant          =          0        #  single dish antenna index in file

thermalnoise        =         ''        #  add thermal noise: [tsys-atm|tsys-manual|""]
leakage             =        0.0        #  cross polarization
image               =       True        #  (re)image $project.ms to $project.image
     vis            = '$project.ms'     #  Measurement Set(s) to image
     modelimage     =         ''        #  prior image to use in clean e.g. existing single
                                        #   dish image
     imsize         =          0        #  output image size in pixels (x,y) or 0 to match
                                        #   model
     cell           =         ''        #  cell size with units or "" to equal model
     niter          =        500        #  maximum number of iterations (0 for dirty image)
     threshold      =    '0.1mJy'       #  flux level (+units) to stop cleaning
     weighting      =   'natural'       #  weighting to apply to visibilities
     mask           =       True        #  clean mask -- see help clean
     outertaper     =         []        #  uv-taper on outer baselines in uv-plane
     stokes         =        'I'        #  Stokes params to image

analyze             =      False        #  (only first 6 selected outputs will be displayed)
graphics            =    'screen'       #  display graphics at each stage to
                                        #   [screen|file|both|none]

verbose             =      False
overwrite           =       True        #  overwrite files starting with $project
async               =      False        #  If true the taskname must be started using
                                        #   simdata(...)

CASA <3>:
```

skymodel

- FITS image, CASA Image, CASA Clean Component Image

setpointings

- single/mosaic: J2000 RA Dec, ptg-file

predict

- Observation settings, antennalist

noise

- tsys-atm, tsys-manual

leakage

- polarization

image

- clean parameters

analyze,

- plots 6 of the following in the simulations pane:

  showarray, showuv, showpsf, showmodel, showconvolved, showclean, schowresidual, showdifference, showfidelity

more on simdata...

`skymodel`

- FITS image, CASA Image, CASA Clean Component Image

`setpointings`

- single/mosaic: J2000 RA Dec, ptg-file

`predict`

- Some configurations available in: `data/alma/simmos`

`noise`

- `tsys-atm` - uses aatm, python wrapper of Juan Pardo's atm library: `pwv`, `tground`
- `tsys-manual: tground, tsky, tau` (zenith opacity)

Output Files

```
$project.skymodel* : scaled/psf-convolved skymodel

$project.ms, $project.noisy.ms, $project.quick.psf,
$project.fidelity…
```

Next: Group Work

- Available for reference: Pre-calibrated 3C 273: data and scripts
  - /home/frank/3c273.demo/
- Calibration Exercise: KAT-7 Observations
  - /home/frank/3c273.tarbz2
  - /home/frank/3c273_3c286.hh_vv.ms.tar
  - /home/frank/scripts/
- Simulation Exercise: 3C 286
  - Tomorrow?
- Discussion of Results

Stuff

Calibration	**Simulation**
• Measurement set download • Calmodel data/nrao/VLA/CalModels/ 3C286_L.im/	• Skymodel data/nrao/VLA/CalModels/ 3C286_L.im/ • Configuration kat7.cfg data/alma/simmos