X-ray data: The Chandra Source Catalog (Evans+ 2010, 2019)

A catalog of all sources detected in observations by the Chandra X-ray satellite. It is not an all-sky survey, it is simply a list of X-ray sources detected in various Chandra observations over the years.

The catalog contains lots of information, we will only be interested in the following:

- Sky position (RA, Dec): `RAIRCS, DECIRCS`
- Positional uncertainty (in arcsec): `r0`
- Signal to noise of the detection: `S/N`
- X-ray Flux and upper/lower limits, measure by the Chandra ACIS detector: `F90b, b_F90b, B_F90b`
- X-ray hardness: `HRhs`

We will get this info only for objects that are

- X-ray point sources
- detected in the Chandra ACIS imaging
- with S/N > 2
- within a 0.5 degree radius around the center of each cluster

We will do this by doing using astropy's `astroquery` function to pull the catalog from the Vizier data portal.

Using astroquery.vizier

```
try:

    # if the datafile is already stored locally on your computer, read it from there
    CHANDRA=ascii.read(cluster_name+'_CHANDRA.csv')

except:

    # if there is no local datafile on your computer, go get the data from Vizier
    from astroquery.vizier import Vizier

    # get this data: RA, Dec, position accuracy, S/N, ACIS X-ray flux, fluxLL, fluxUL, hardness
    get_cols=['RAICRS','DEICRS','r0','S/N','F90b','b_F90b','B_F90b','HRhs’]
    # select these sources: X-ray point source, detected ACIS flux, S/N > 2
    use_filters={"fe":"=0","F90b":">0","S/N":">2"}

    # set up a Vizier query function
    v=Vizier(columns=get_cols,column_filters=use_filters)
    v.ROW_LIMIT=-1 # -1 means no row limit

    # query Vizier for objects in a 0.5 degree radius around the position
    # of the cluster, getting the data from catalog "IX/57" (which is the Chandra
    # source catalog). The [0] thing just means get the first data table, which
    # in our case is the only data table.
    CHANDRA=v.query_region(cluster_pos,width="0.5d",catalog="IX/57")[0]

    # write the downloaded data to a local datafile for subsequent use
    CHANDRA.write(cluster_name+'_CHANDRA.csv')
    print('Wrote {}_CHANDRA.csv'.format(cluster_name))
```

Cross-matching catalogs

Two catalogs: A and B. For each object in catalog A, is there a match in catalog B?

Match on unique property (name or ID #): this is what we did with the SDSS data with the Photometric catalog and the Spectroscopic catalog.

Match on angular separation: For each object in catalog A, what is the closest object in catalog B?

- Catalog A = Chandra Point Source Catalog (PSC)
- Catalog B = Our SDSS dataset of galaxies in the field of each cluster

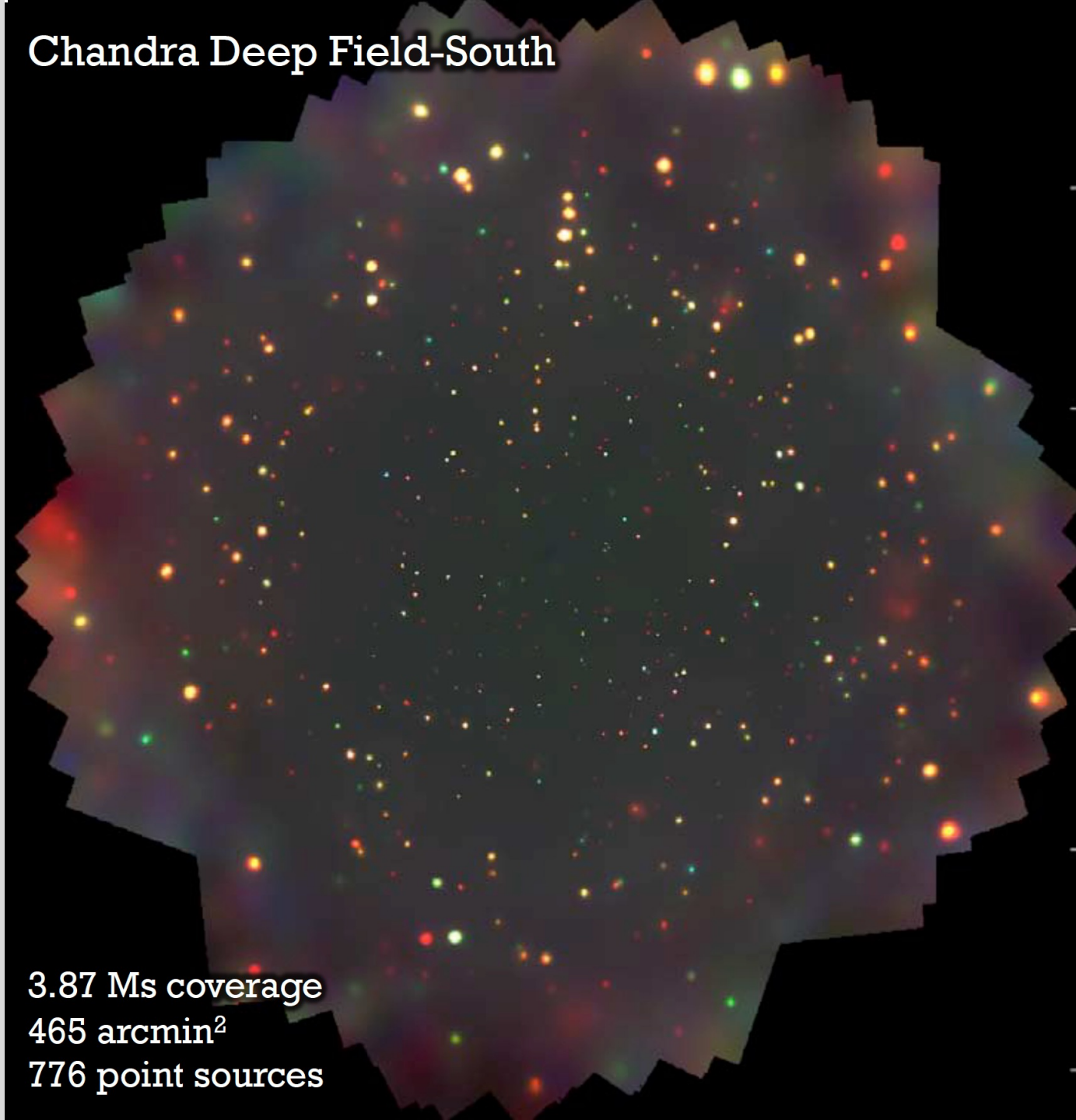
But remember:

- *By definition, there will **always** be a closest object B!*
- *The closest match may not be very close!*
- *The closest match may not be the correct object!*

So we have to think about the how good the matches are, and not just use them blindly…

How well determined are the coordinates?

- **SDSS**: ≈ 1 arcsec accuracy
- **Chandra**: typically ≈ few arcsec accuracy but can exceed 10 arcsec depending on S/N and location in X-ray image



Matching catalogs by coordinates

We have two datasets: SDSS (optical) and CHANDRA (X-ray). We want to cross-match them into one dataset that contains the optical and X-ray data for all X-ray sources. We do this by matching on position.

```
# make a sky coordinate object for all the objects in the SDSS catalog....
sdss_coord=SkyCoord(SDSS['ra'],SDSS['dec'], unit='deg', frame='icrs')

# ....and for all the objects in the Chandra catalog
chandra_coord=SkyCoord(CHANDRA['RAICRS'],CHANDRA['DEICRS'], unit='deg', frame='icrs')

# now for each chandra_coord, match it to the nearest sdss coord using sky coords (ra,dec)
# idx lists (for each chandra source) the row number in the SDSS table that contains the
#      closest match.
# d2d is the 2D separation between the sources (angle on the sky)
# d3d is the 3D separation, which is meaningless for us, since we dont have distances
idx, d2d, d3d = chandra_coord.match_to_catalog_sky(sdss_coord)
```

SDSS[idx] will be a data table where each row of data is the optical properties of the object ***closest to*** the X-ray sources listed in the CHANDRA table, in the proper row order to match the CHANDRA table.

Matching catalogs by coordinates

SDSS[idx] will be a data table where each row of data is the optical properties of the object **closest to** the X-ray sources listed in the CHANDRA table, in the proper row order to match the CHANDRA table.

So we can now do a horizontal stack of the CHANDRA and SDSS[idx] to make our final data table.

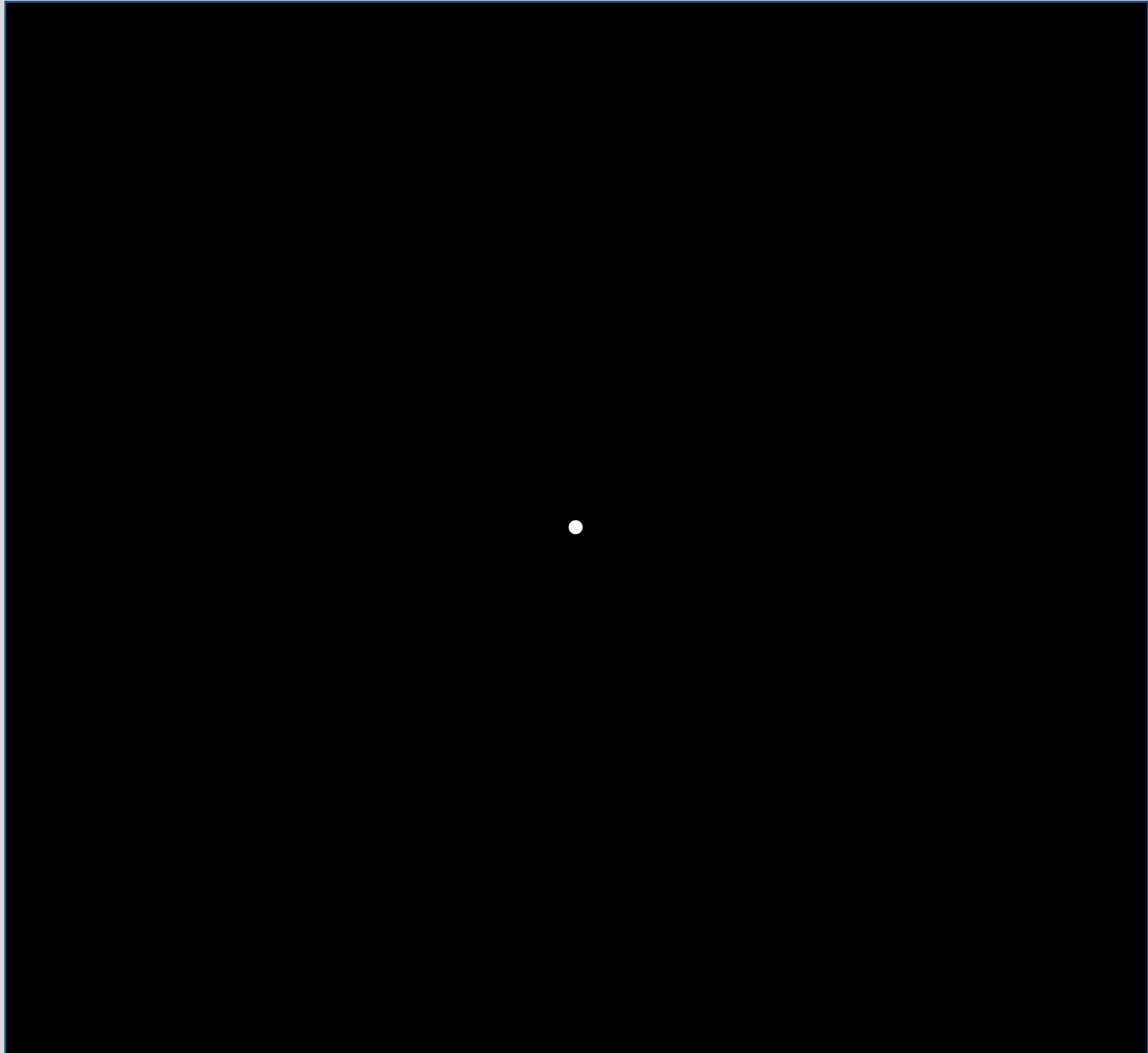
```
# now make a merged list by "horizontally stacking" the two tables: CHANDRA and SDSS[idx]
from astropy.table import hstack
CROSSMATCHED=hstack([CHANDRA,SDSS[idx]])
```

CROSSMATCHED is a new data table containing our final combined/matched X-ray and optical data for all X-ray sources.

One important thing: we matched on the closest objects. The closest object may still be far away, and may not really be a good match! So we will add a column to our crossmatched dataset that says how far away the closest optical object was:

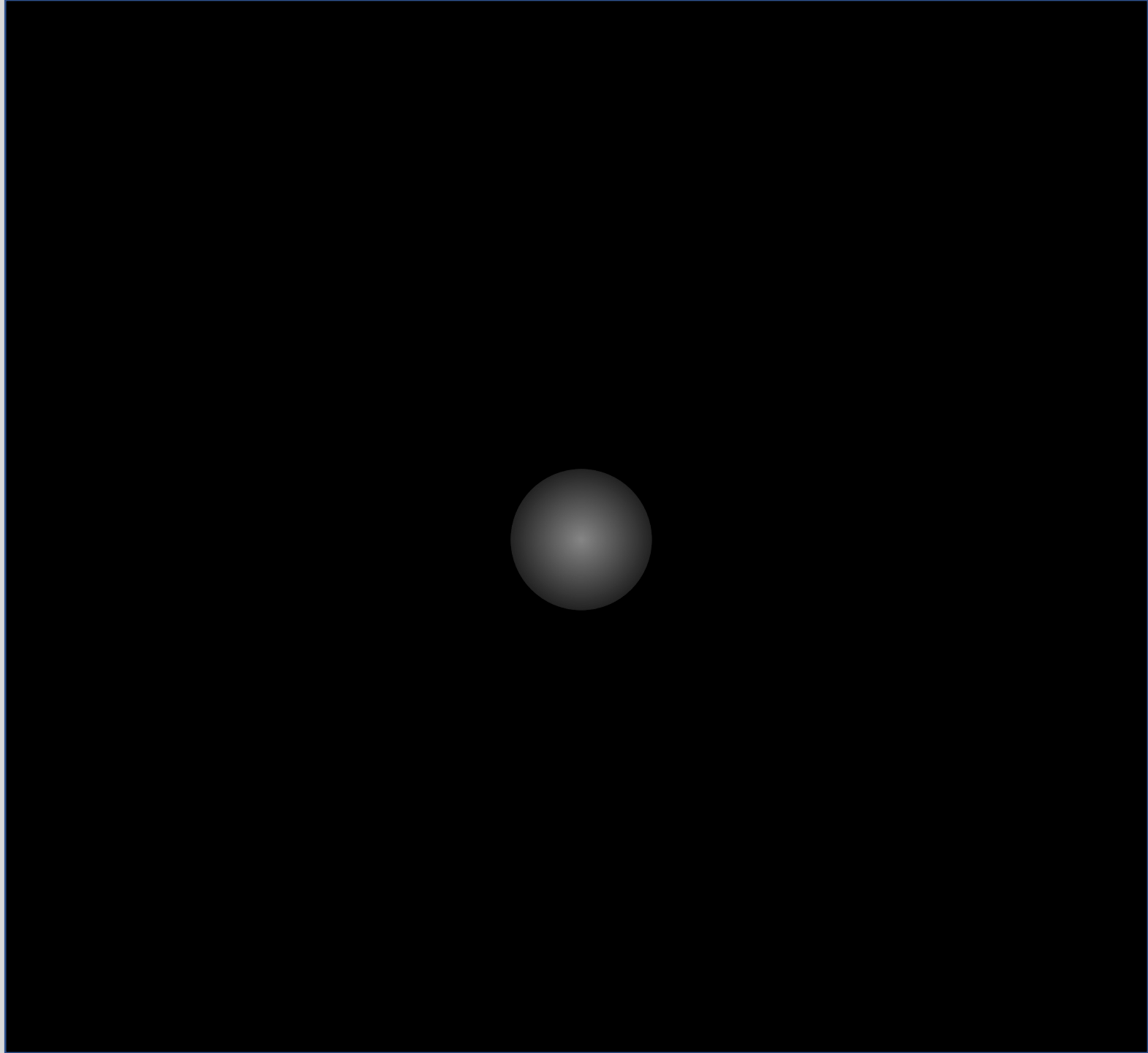
```
# add a column that shows the separation in arcsec
CROSSMATCHED['match_sep']=d2d.arcsec
```

Small X-ray positional error, well determined position.



Larger X-ray positional error, less well determined position.

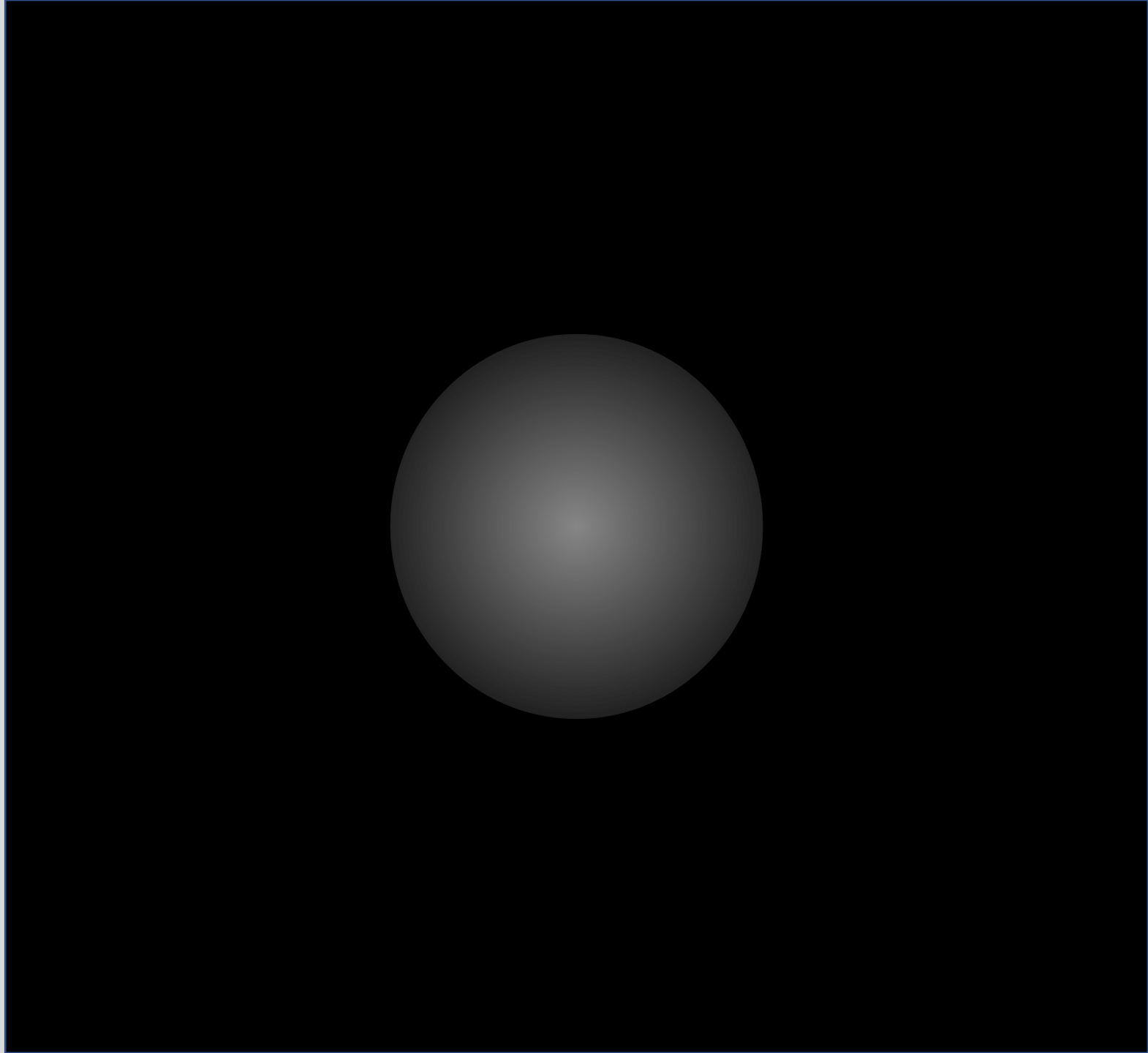
Think of the position and position error as a “location likelihood”.



Very large X-ray positional error, much less well determined position.

“location likelihood” is quite broad.

Happens at low flux levels or when X-ray sources are near the edge of the X-ray field of view.



SDSS image

Good match!

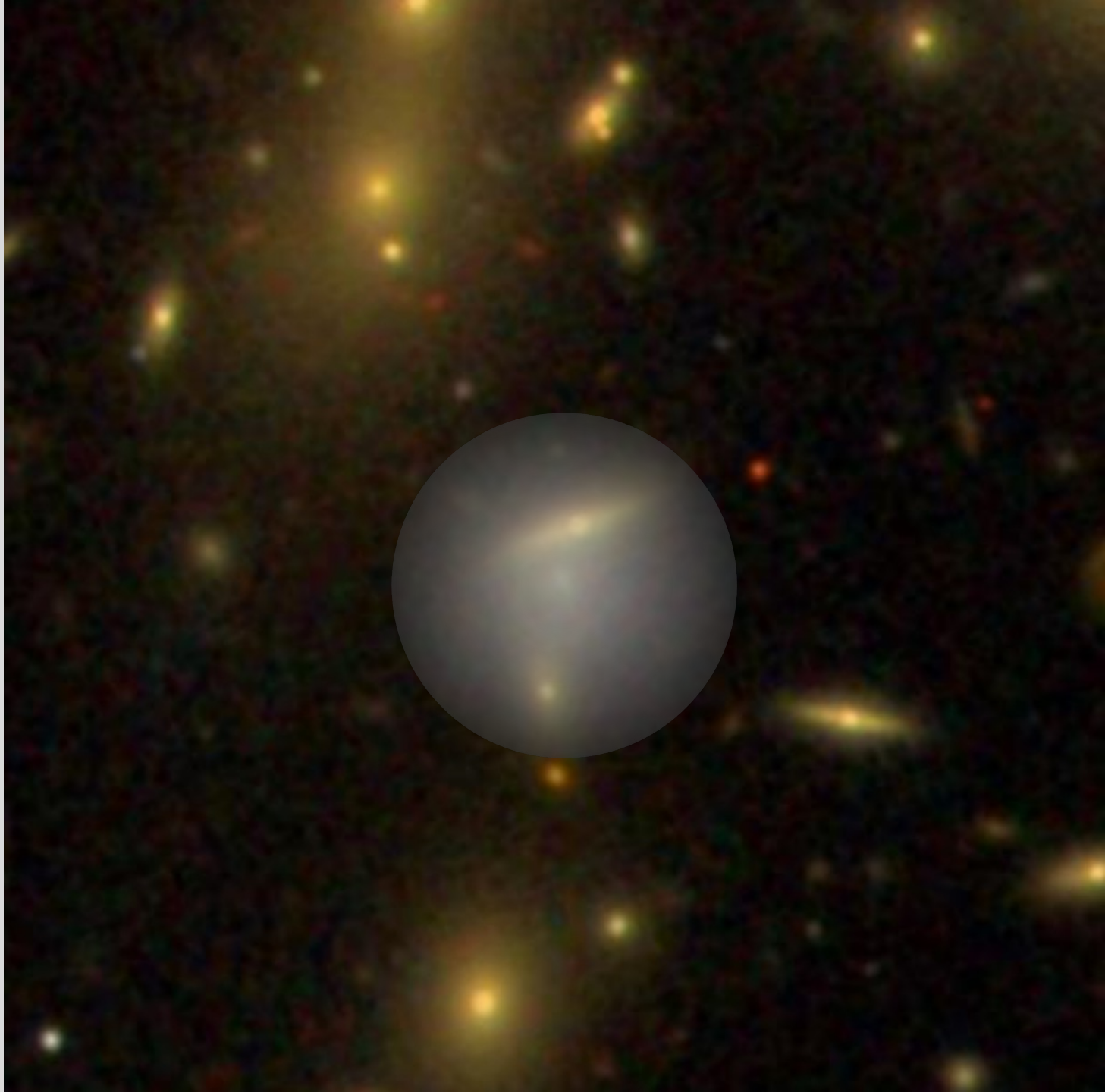
Probably a good match

Hmmm….

Nope

Probably a good match

Uh oh



Oh no

When looking at matched objects in SDSS Navigator, don’t assume the match is correct. Keep these things in mind:

- Know the separation between matched objects (match_sep)
- Know the uncertainty of the Chandra position (r0)
- Know the scale of the SDSS field of view in Navigator (click ’grid’)
- Think about how accurate the match might be!

Is the separation less than the positional uncertainty? (match_sep < r0)
(But how accurate is the positional uncertainty?)

Is the separation small? (match_sep < some_critical_value)

Are there other objects very close nearby?