A Tutorial on Computed Tomography

Sarah Vallélian

samsi
NSF • Duke • NCSU • UNC • NISS

October 19, 2015

Outline

What is Computed Tomography?

Computed tomography is a medical imaging technique which produces cross-sectional images or "slices" of the body, by detecting either:

- Gamma rays– both PET and SPECT
- X-rays– called X-ray CT or CAT scans

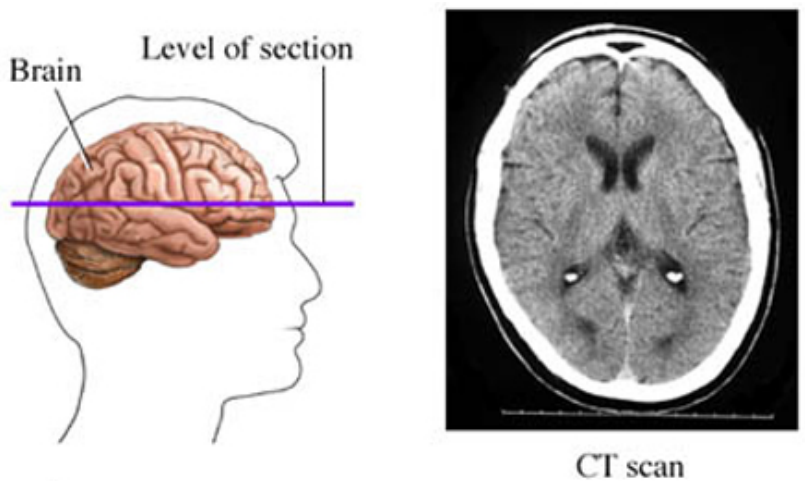


Figure: An example of a brain CT slice.[1]

[1] www.bidmc.org

What is it good for?

- X-ray CT has high contrast: easy to tell different tissues apart!
- Fast scanning times: around 1s! Whole process can take as little as 10 min!
- Often used as a first diagnostic test for brain tumors and in emergency situations.
- Non-invasive . . . although

Involves moderate or high levels of radiation.

HOW DOES IT WORK?

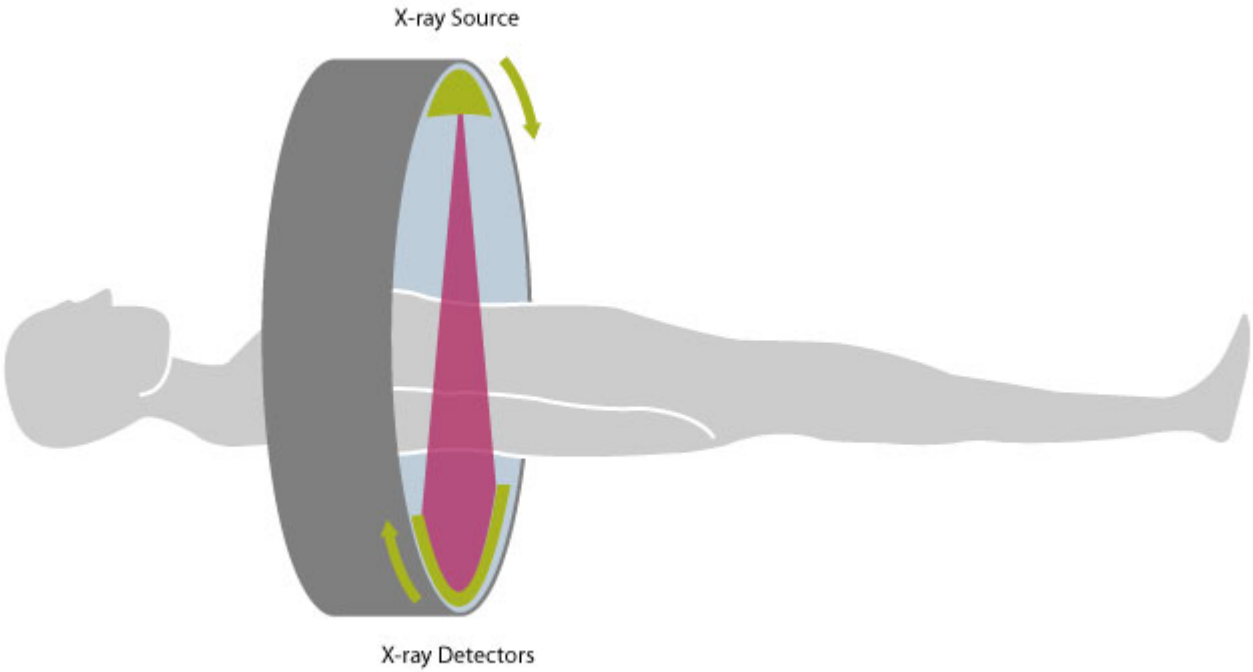
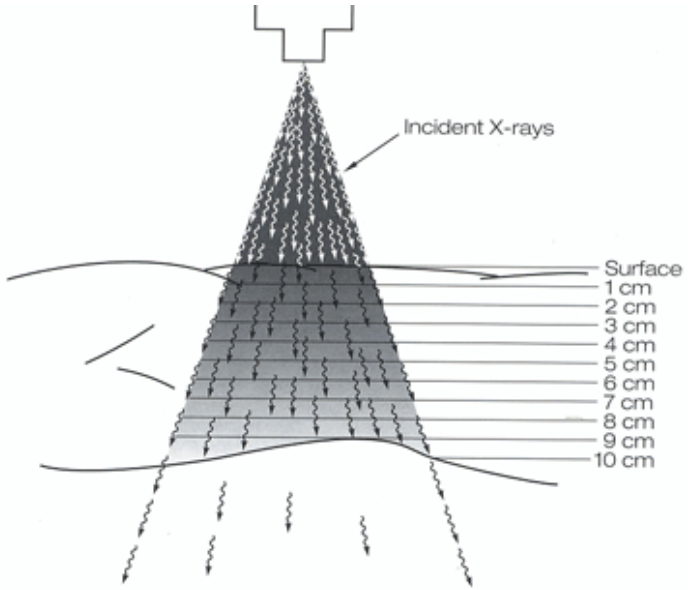


Figure: Gathering data in a CT scan procedure.[2]

[2] www.medicalradiation.com

Principles of CT

- X-rays are partially absorbed/scattered, aka attenuated, while traveling through tissues as thin beams or lines.



- The amount of attenuation differs between different tissues!

Knowing how the X-rays attenuated at different points in the body = knowing the body's composition = image of the body!

Where's the math?

Goal: Recover the amount of attenuation at each point in the body. Do this repeatedly for many 2-D slices. For a single slice, we want the attenuation coefficient $a(\mathbf{x})$, $x \in \mathbb{R}^2$.

- Each detector measures the X-ray intensity

$$I_1 = I_0 \exp\left\{ -\int_L a(\mathbf{x})\, \mathrm{d}s \right\},$$

  where L is the line between the X-ray source and detector. (Beer-Lambert law of physics)
- This can be rewritten as

$$\int_L a(\mathbf{x})\, \mathrm{d}s = \log \frac{I_0}{I_1}, \tag{1}$$

  where the RHS is a known quantity.

THE CT PROBLEM

Question: Using the detector's measured data (1), for many different lines L, how can we recover $a(\mathbf{x})$?

The data and the attenuation coefficient are related by the 2-D Radon transform,

$$a(\mathbf{x}) \mapsto Ra(L) := \int_L a(\mathbf{x})\, \mathrm{d}s, \tag{2}$$

also called the 2-D X-ray transform.[3]

- X-ray CT relies on being able to invert this transform R!

[3] In 3-D, however, these two transforms are not the same!

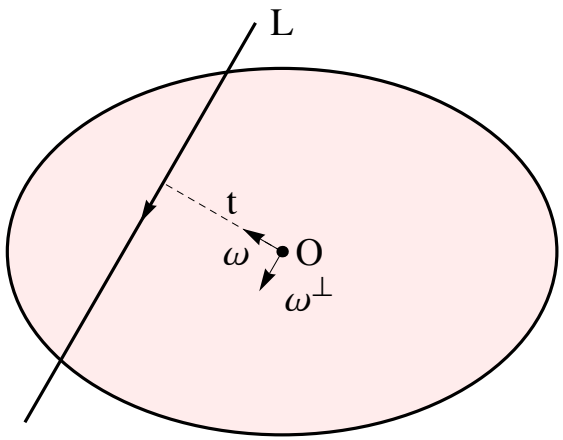
Parameterizing Lines in 2-D

A standard way to represent lines in 2-D for computing the Radon transform:

$$L = \left\{ \mathbf{x} \in \mathbb{R}^2 : \mathbf{x} \cdot \theta = t \right\}$$

The line L is parameterized by:

- A unit vector $\theta \in \mathbb{S}$ perpendicular to the line, and
- A distance t from the origin.



Properties of Radon Transform

With the line parameterization, the 2-D Radon transform is expressed as

$$Ra(t,\theta) = \int_{-\infty}^{\infty} a(t\theta + s\theta^{\perp})\,\mathrm{d}s, \tag{3}$$

where $\theta^{\perp}$ is the unit vector parallel to the line L.

- Well-defined on the Schwartz space, which is the space of smooth, rapidly decaying functions.
- $Ra(t,\theta)$ is an even function, also smooth and rapidly decaying.

Adjoint Radon Transform

Related to the Radon transform, and an important part of the inverse Radon transform, is the adjoint Radon transform, defined as

$$g(t,\theta) \mapsto R^{\#}g(\mathbf{x}) := \int_{\mathbb{S}} g(\mathbf{x}\cdot\theta,\theta)\,\mathrm{d}\theta \tag{4}$$

- If $g(t,\theta)$ is the Radon transform of a, then $g(\mathbf{x}\cdot\theta,\theta)$ is the integral of a along a line L passing through $\mathbf{x}$.
- Varying θ gives us all lines passing through $\mathbf{x}$.
- So $R^{\#}g(\mathbf{x})$ is an "average" of the Radon data over all points (t,θ) parameterizing a line passing through $\mathbf{x}$.

$R^{\#}$ is also known as back-projection.

Inverse Radon Transform

Deriving the inverse Radon transform involves use of the Fourier transform and the Hilbert transform, denoted by H.[4]

$$a(\mathbf{x}) = \frac{1}{4\pi} R^{\#} \Lambda R a(\mathbf{x}), \tag{5}$$

where $\Lambda = H \frac{\mathrm{d}}{\mathrm{d}t}$. The code to invert the Radon transform on the computer illustrates how this inverse procedure works:

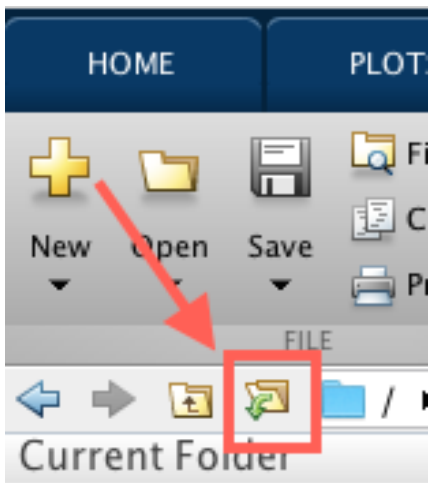
- The operator Λ acts as a "filter": the Radon data is Fourier transformed, multiplied by a filter function in the frequency space, and inverse Fourier transformed back.
- Then the back-projection operator $R^{\#}$ is applied to the filtered Radon data.

This algorithm is called filtered back-projection.

[4] Notes with the mathematical details for this section will be available online for those interested.

Setting up in Matlab or Octave

1. Download or copy `demo.zip` to an easy-to-find folder and unzip the files.
2. Open Matlab or Octave.
3. Change your current folder to the location containing the demo files.

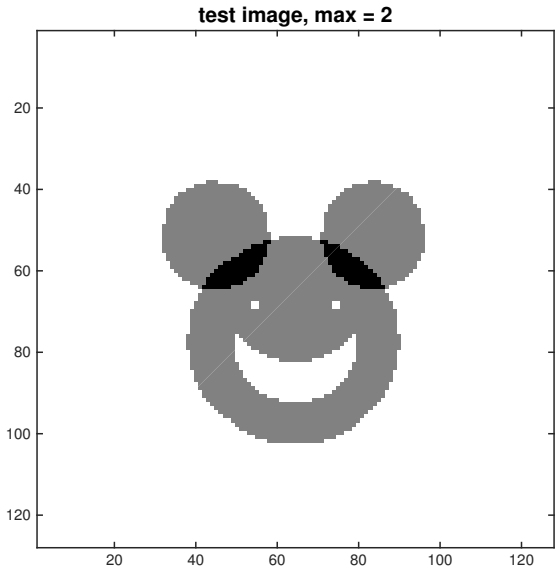


RUNNING PARTS 1 AND 2

Double-click `xrayCTdemo_1.m` to open the demo code for Parts 1 and 2 in your Editor window.
Type `xrayCTdemo_1` in your Command Window and hit enter to run the demo Parts 1 and 2.

Part 1: Generating test images

Here we are specifying the test images for our Radon and inverse Radon experiments.



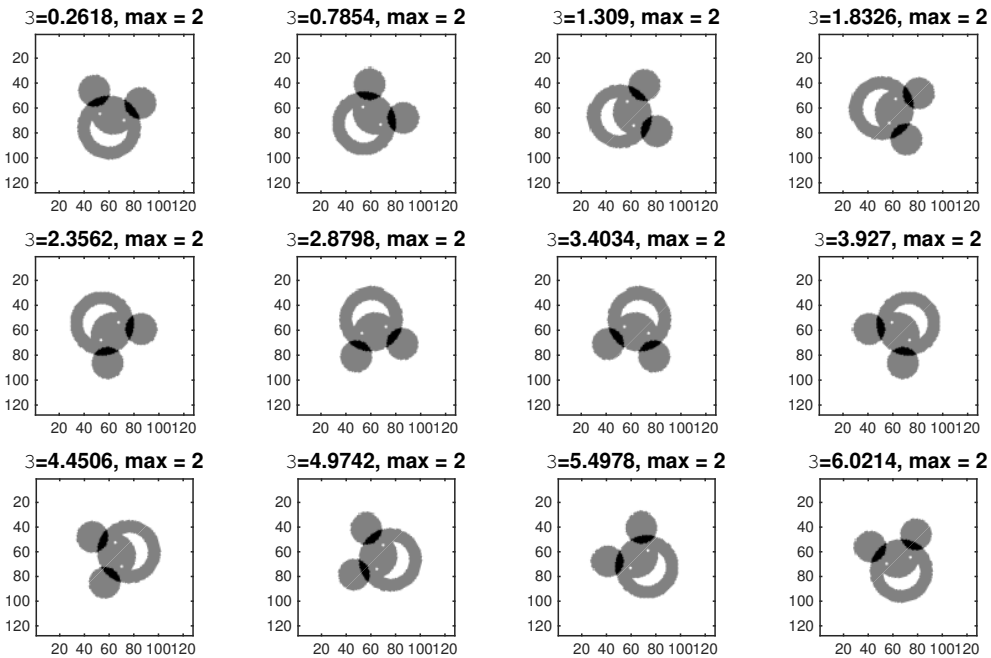
You can change the parameters:

- `'n'`: number of pixels along one side of the square image
- `'im'`: a matrix representing the test image values per pixel

by adding and removing the comment symbol % in front of a line specifying the possible values.

Part 1: Rotations

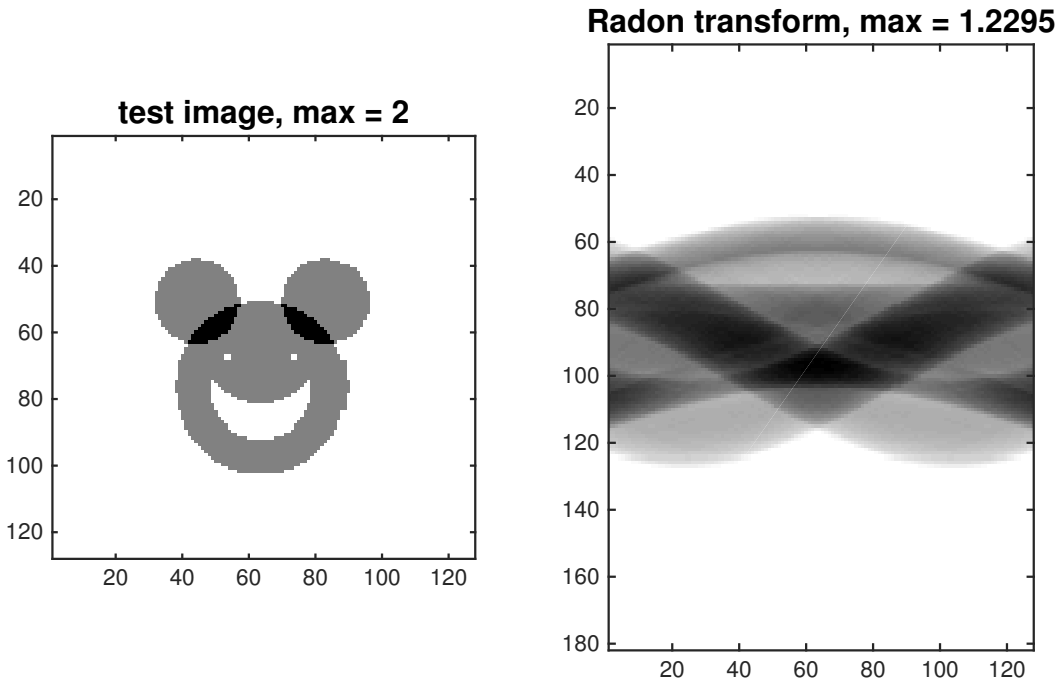
The Radon data are collected along certain lines, specified by angles in the parameter `'thetas'`



Rotating the whole image rather than the lines of integration makes calculating the Radon data easier.

PART 2: GENERATING THE RADON DATA

In Figure 3 we can see the Radon transform of the test image, computed along many lines (`n × nth`). This type of image is often called a sinogram.



To see in detail how the Radon transform was computed, open `myRadon.m` in the Editor. [5]

[5] Further documentation for the codes will be made available for those interested.

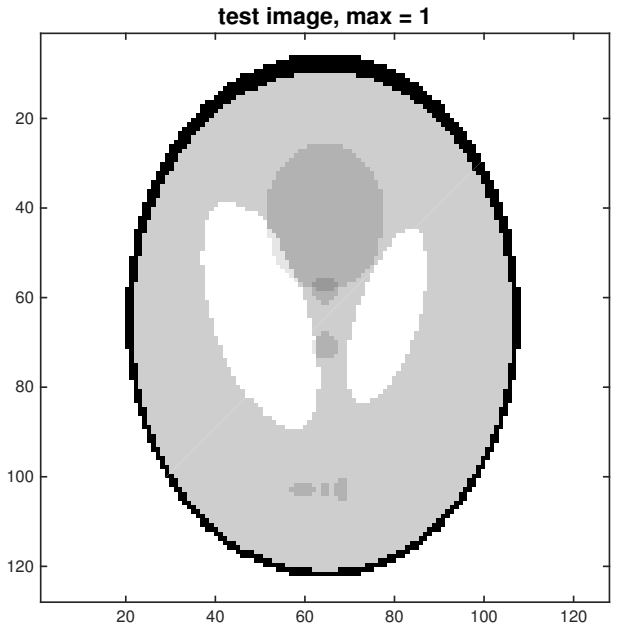
PART 2: DISCUSSION

```
% Discuss: Try changing the values of the resolution 'n', the test image
% 'im', and the number of rotation angles 'nth'. What changes in the
% pictures of the Radon transforms as each of these variables changes?
```

Can you pick out features of your test image in your Radon transform? If not, try the test image `stripes`!

RUNNING PARTS 3 AND 4

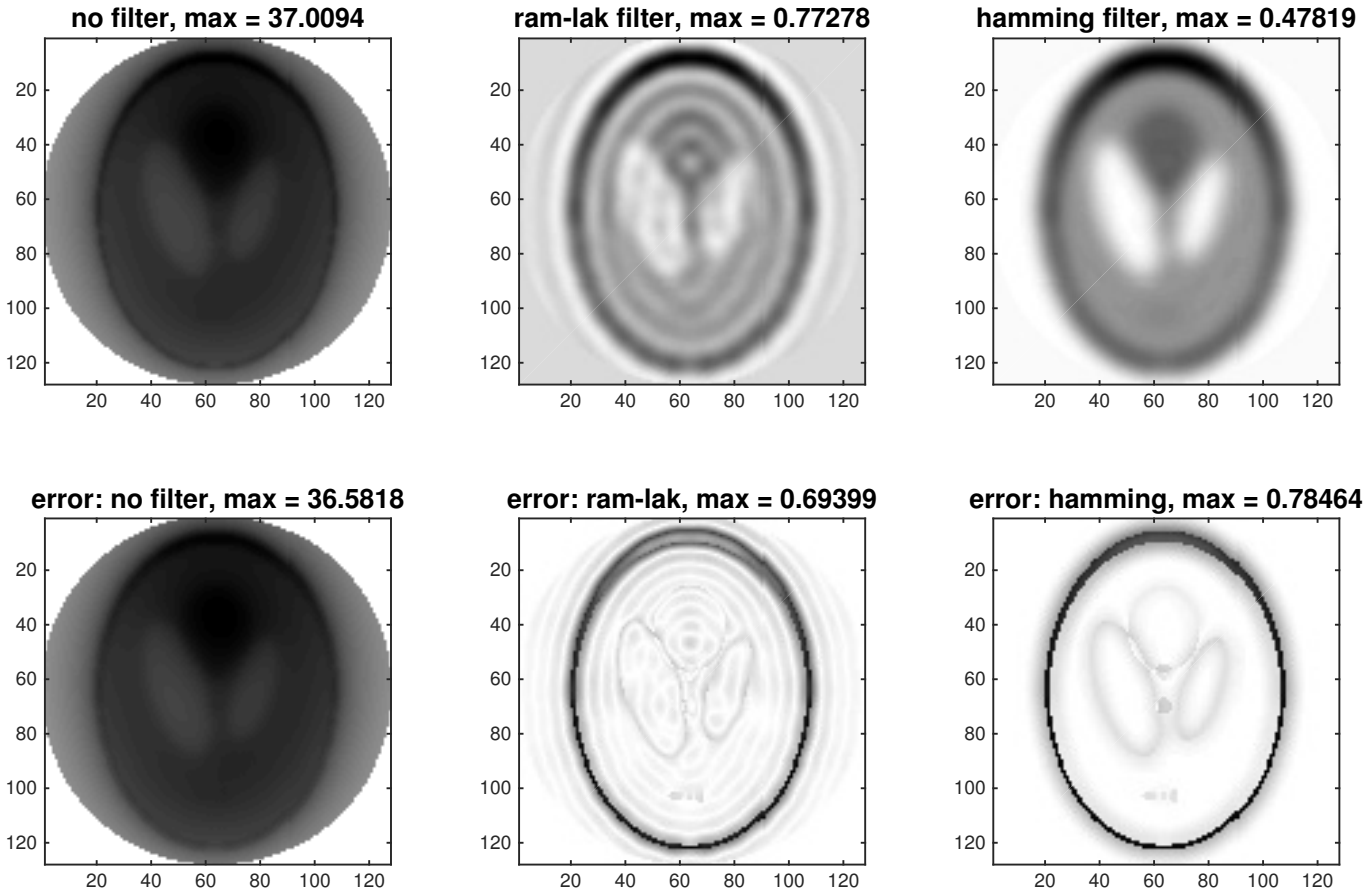
Double-click `xrayCTdemo_2.m` to open the demo code for Parts 3 and 4 in your Editor window.
Type `xrayCTdemo_2` in your Command Window and hit enter to run the demo Parts 3 and 4.



The test image for these parts is the well-known Shepp-Logan phantom.

Part 3: Filtering and the inverse Radon transform

In this part, we see the effect that the filtering step has on the filtered back-projection algorithm for X-ray CT image reconstruction.



PART 3: DISCUSSION

```
% Discuss: What do you notice as the main difference(s) between the
% unfiltered and filtered back-projections? Can you see a difference
% between the two different filters?
```

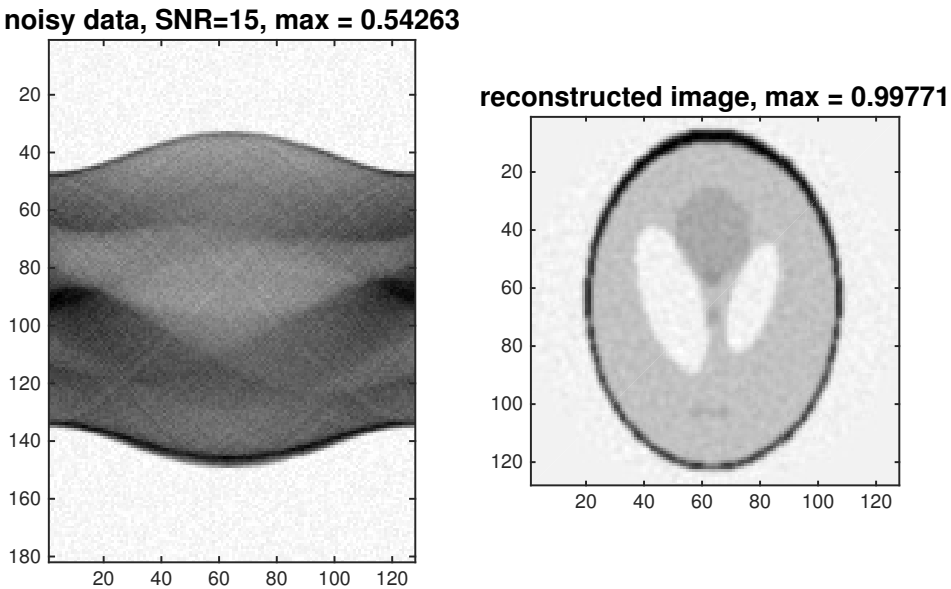
To see in detail how the inverse Radon transform was computed, open `myIRadon.m` in the Editor. [6]

[6] Further documentation for the codes will be made available for those interested.

Part 4: Image reconstruction with noisy data

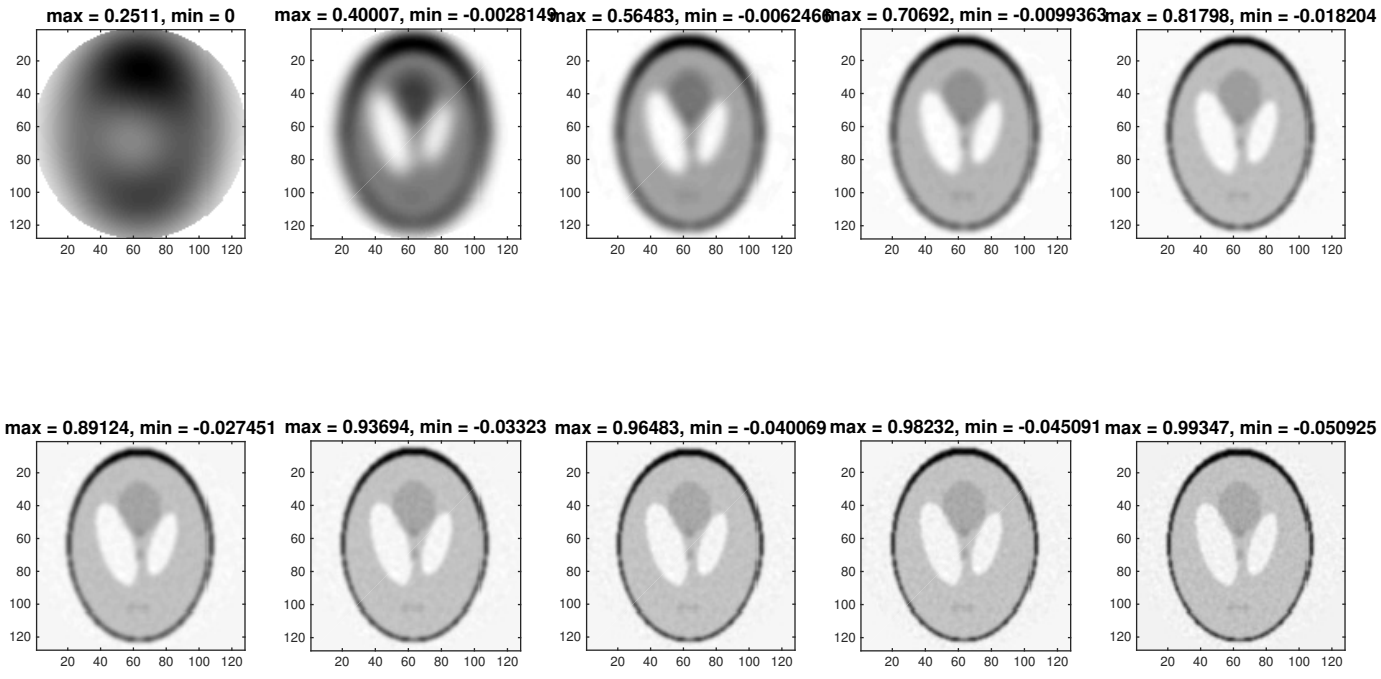
We add random noise to our Radon transform data to test how accurate reconstruction is in realistic settings. The noise level is described by the signal-to-noise ratio, or `'SNR'`.

In Figure 6,



Which details from the original test image are lost or hard to distinguish? Why?

Part 4: Filtering noisy data at different levels

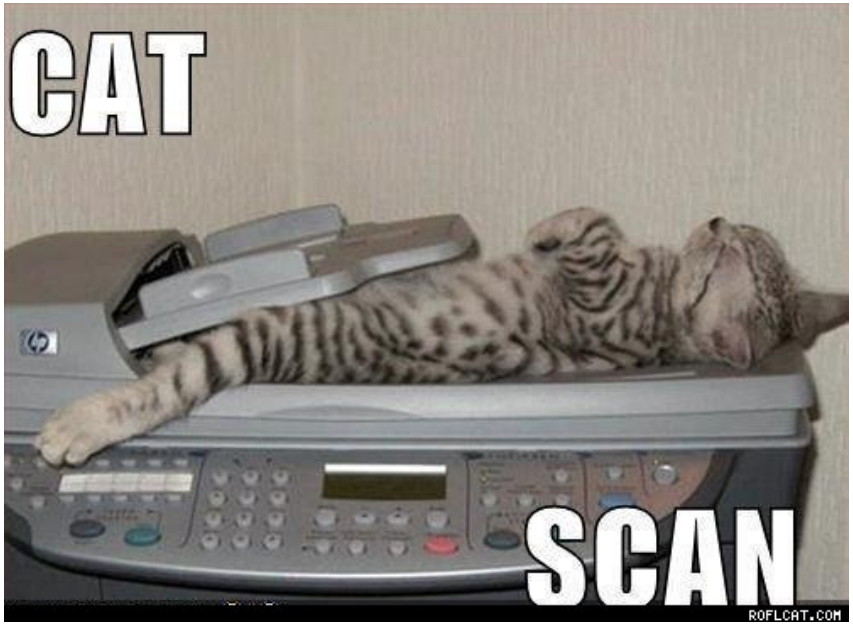


In filtered back-projection, we can control the strength of the filtering, by determining how many frequencies to keep or throw away. The amount of filtering affects the reconstructed image, as we can see in Figure 7.

Part 4: Discussion

```
% Discuss: In Figure 7, we see that the Hamming filter is used with
% different parameters 'd' in the filtered back-projection of the noisy
% signal. The parameter increases from 0 to 1 in the pictures left to right
% and top to bottom. This controls the strength of the filtering effect.
% When 'd' is small, only the low frequencies are used, and the
% reconstructed images are a bit blurry but do not have noisy specks.
% When 'd' is close to 1, all but the highest frequencies are used, and
% the reconstructed images have more detail but also more noise.
%
% Which level of 'd' do you feel is optimal? Does this change if you rerun
% the code with different values of 'SNR'?
```

Feel free to go back to each part of the demo and change the parameters!



Thank you!