

GEORGIA INSTITUTE OF TECHNOLOGY
SCHOOL OF ELECTRICAL ENGINEERING
ECE 6258 Fall 2005
CLASS PROJECT

Assigned: 6 October 2005
Due Date: 23 November 2005

This project is to be done *individually*. Do NOT discuss the details of your design with other students. Questions or clarifications should be directed to Prof. Mersereau. Errata, revisions and hints (if any) will be posted on the course web site.

Your write-up for this project should include:

1. An overview description of how the processing program works. This should include a *block diagram* of the major system components, but need not include minute details of the implementation.
2. A brief description of the theory that is relevant to the reconstruction problem and to your solution of it.
3. A list of references consulted in the preparation of your algorithm.
4. A listing of the MATLAB program in an Appendix. This is sufficient to convey most details, so write modular programs, with separate functions for each major signal processing operation. Your code should be sufficiently commented so that the graders can understand it.
5. Results showing how the program performs on the test cases. Some of the plots should show the intermediate results, e.g., the filtered projection, one back projection, etc. Give hardcopy of plots, especially when you must make measurements on the graphs.

Start working early. If you encounter problems, please ask questions to clarify anything that is vague. **DEADLINE:** Beginning of class on Wednesday, 23 Nov 05.

The projection data is in a file called (`projdata.mat`), which contains three variables: the projections are in a 61×180 array called `PP` in which each column is a single projection. Projections were measured every 1° so the total number of angles is 180. Use the `load` command in MATLAB to access this data. The file also contains vectors called `theta` which gives the order of the angles and `uu` which gives the sample points on the u -axis.

This project requires that you develop a processing implementation for the method of “reconstruction from projections.” An unknown object is to be reconstructed from projection data that can be loaded from the file `projdata.mat`. The choice of reconstruction algorithm is up to you. The unknown contains one or more of the following objects:

1. A rectangle: you must try to determine the locations and dimensions of the rectangle and its amplitude
2. An ellipse: you must try to determine the equation for the ellipse including its center.

3. A Gaussian with ellipsoidal level curves. You must try to deduce an equation for the ellipsoidal level curves.

If the 2-D objects are considered to lie in the (x, y) plane, these ellipsoidal equations can be written in the form:

$$\alpha(x - x_o)^2 + \beta(y - y_o)^2 + \gamma(x - x_o)(y - y_o) = \text{constant},$$

where (x_o, y_o) is the center of the ellipse, and α , β and γ determine the length and orientation of major and minor axes of the ellipse.

A reconstruction on a grid that is 64×64 or greater should be attempted. Then the resolution should be adequate to find the various objects in the image. The original object was confined to lie in the (x, y) -plane in a square defined by

$$-1 \leq x \leq +1 \quad \text{and} \quad -1 \leq y \leq +1.$$

The amplitudes of the component objects is different so they would appear as different gray levels in an image. **Note:** the objects were added together if they overlapped in space. A small amount of white noise was added to the final image.

PROJECTIONS

The projections were measured at an angular spacing of 1° . Along the projection axis itself, the sampling was uniform. If this axis is called the \hat{u} -axis and the sampling interval $\Delta\hat{u}$, then $\Delta\hat{u} = 0.05$. The total extent of the projection axis was:

$$-\frac{3}{2} \leq x \leq +\frac{3}{2}$$

so there are a total of 61 samples along each projection.

Each projection can be considered as a line integral through the image $x(u, v)$:

$$p_\theta(\hat{u}) = \int f(\hat{u} \cos \theta - \hat{v} \sin \theta, \hat{u} \sin \theta + \hat{v} \cos \theta) d\hat{v}$$

The (\hat{u}, \hat{v}) coordinate system is rotated from the original (u, v) coordinates by the angle θ . Thus projections taken at angles in the range $-90^\circ < \theta \leq +90^\circ$ are sufficient to reconstruct the image.

RECONSTRUCTION METHODS

There are at least three possible methods for performing the reconstruction:

1. Filtered Backprojections,
2. Iterative Approach,
3. Fourier Reconstruction.

All three of these methods are described (to some extent) in the course notes.

Your implementation can be done by any of these methods, or by some combination of them, but experience suggests that the filtered back-projection method gives better results with a simple program. On the other hand, a successful Fourier reconstruction would be most impressive!!! You should also feel free to apply some post-processing to the result. A portion of the grade will be based on the novelty of your implementation. It is recommended that you pick one approach and work with it in depth.

TESTING

In the process of developing your reconstruction algorithm, you should experiment with some trade-offs:

- (a) Try the reconstruction with different numbers of projections. For example, do it with $\Delta\theta = 10^\circ$, and $\Delta\theta = 15^\circ$, as well as $\Delta\theta = 5^\circ$ and $\Delta\theta = 1^\circ$.
- (b) Experiment with different filtering methods if you choose the filter and back-projection method. The $|\Omega|$ filter in the Radon inversion formula must be approximated with either an FIR filter or a frequency domain operation.
- (c) Experiment with applying filters and/or constraints on the unknown density if you use an iterative approach. There are also different possibilities for the back-projection operation.
- (d) Experiment with different interpolation methods if you choose the Fourier reconstruction method.

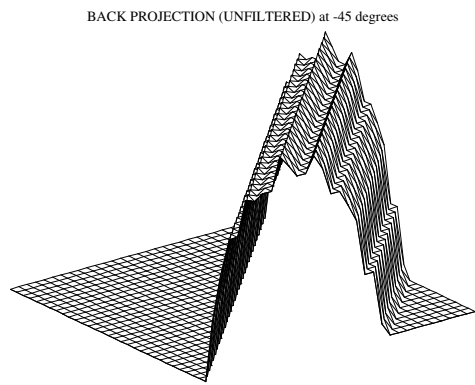
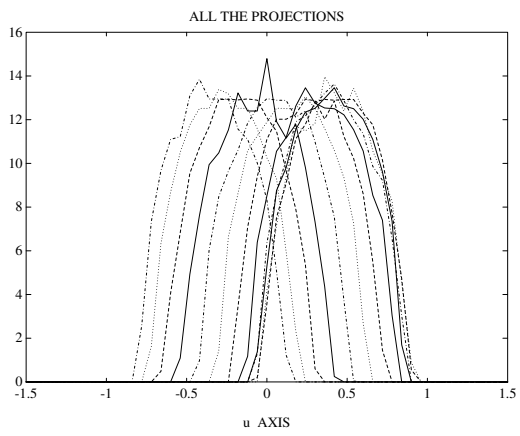
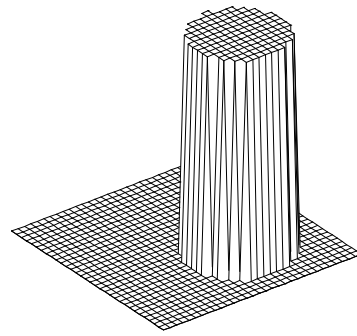
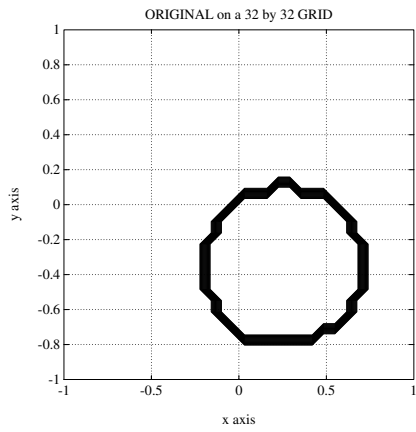
PROGRAMMING PROBLEMS

This project will be rather difficult to program in MATLAB because the basic operation of slicing through an array which is defined on a discrete grid is not easy. Past experience with similar algorithms have revealed student solutions whose run times varied over three orders of magnitude. It does not vectorize in any obvious manner. The following programs are offered as samples of one way to map the indices along the projections to the indices along a line in an array. Since there is a discrete grid for the array this mapping is only *approximate*.

You should write a brief description of how to create the mapping between the 1-D projection and the 2-D image. The sample programs provide one method for generating projections, so it should be possible to write this description by “deciphering” the MATLAB code in those files. This is not the only method for generating the projections, however, and you may prefer to work with a different one. (If you have done Problem 3.3, you should have already done this.)

The M-file `sample.m` will generate an object that is a circle (to within the resolution of a 32×32 array). Then the projections of that image are taken at an angular spacing of 15° . See the figures on the next page. Unfortunately, this program runs VERY slowly, so give it a few minutes to complete the job of computing the 12 different projections. In contrast to this program, it is possible to generate the projections from analytic formulas for simple objects such as those that make up the data in this project.

In theory, all these projections should be identical in shape, because the object is a circle. (You can derive a mathematical formula for the projection., e.g. Problem 3.1.) The projections will be shifted relative to one another, however, because the circular object is not centered at the origin in the (x, y) -plane. In addition, due to the finite grid effect, the projections look rough—because there is no way to draw a perfect circle on rectangular grid. (You should be able to do a better job!) Also the projections are being done by summing up samples taken from the finite grid, and each (x, y) sample is spread out linearly to two points on the u -axis of the projection.



SAMPLE.M

```
%--- SAMPLE for making a projection
%--- ECE 6258 5-October 2005
%--- Originally prepared for EE 4171, Mar. 1992
%
L = 32;   smpls = linspace(-1,1,L);
[x,y] = meshgrid( smpls, smpls );
%-----
jkl = find( (x(:)-0.25).^2 + (y(:)+0.35).^2 < 0.45.^2 );
z = zeros(L);
z(jkl) = ones(length(jkl),1);
axis('square')
contour(z,20,smpls,smpls)
title('ORIGINAL on a 32 by 32 GRID')
xlabel('x axis'), ylabel('y axis')
pause, mesh(z), pause
%----- PROJECTION -----
Lu = 51;
u = linspace(-1.5,1.5,Lu);
del_theta = 15;   %--- angular spacing
theta = (-90+del_theta):del_theta:90;
Nth = length(theta);
PP = zeros( Lu, Nth );
for i=1:Nth
    PP(:,i) = projekt( z, x, y, u, theta(i) );
end
axis('normal')
plot(u,PP)
title('ALL THE PROJECTIONS'), xlabel('u AXIS')
```

PROJEKT.M

```
function pp = projekt( zz, xx, yy, uu, theta)
%PROJEKT
% make a projection at the angle theta (in degrees)
%
uu = uu(:);
theta_deg = theta/180*pi;
utt = cos(theta_deg)*xx(:) + sin(theta_deg)*yy(:);
L = length(uu);
delu = uu(2) - uu(1);
jkl = floor( ( utt - uu(1) )/delu ) + 1;
if( min(jkl) < 1 | max(jkl) >= L )
    error('projection axis TOO SHORT');
end
pp1 = zz(:) .* (( uu(jkl+1) - utt )/delu);
pp2 = zz(:) .* (( utt - uu(jkl) )/delu);
%%%keyboard
pp = zeros(uu);
for i = 1:(L-1)
    ii = find( jkl == i );
    pp(i) = pp(i) + sum( pp1(ii) );
    pp(i+1) = sum( pp2(ii) );
end
```