

Lecture 4: Finishing with MATLAB



◆ Part solution to Lab 1:-

```
function [t, sinewave] = sinegen(fsig, fsamp, ncycle)
% Sinewave Generation
%   fsig = signal frequency
%   fsamp = sampling frequency
%   ncycle = number of cycles to generate
%
%   Peter Cheung
%   15th October 1998.

% calculate angular increment per sample
delta_angle = 2*pi*fsig/fsamp;

% create angle vector for 4 cycles
t = 0:delta_angle:4*(2*pi);

% create sine wave
sinewave = sin(t);
% convert angle to time: time = angle/(2*pi*f)
t = t/(2*pi*fsig);
```

Solution to Lab 1 (con't)



```
% Model answer to Lab Session 1  
% Exercise 2 - file: lab1_2.m
```

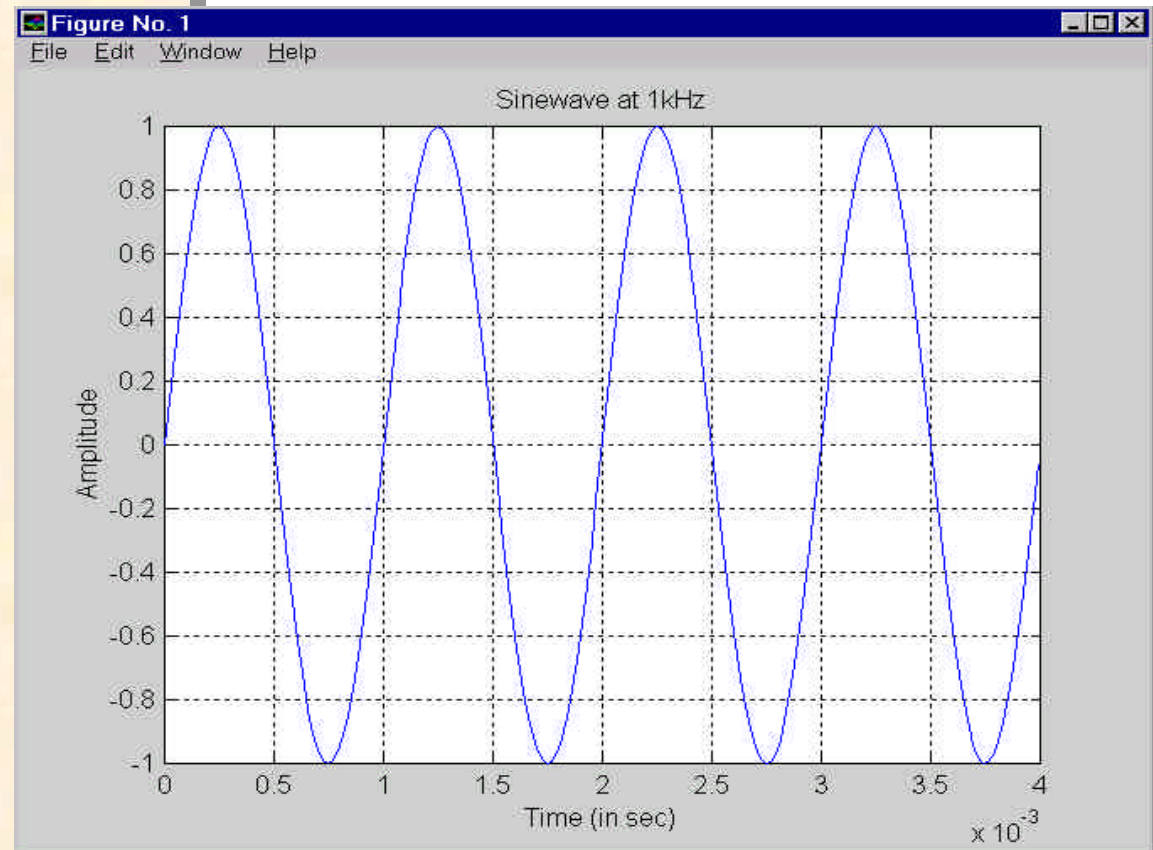
```
% define sampling frequency  
fs = 44100;
```

```
% define signal frequency  
f = 1000;
```

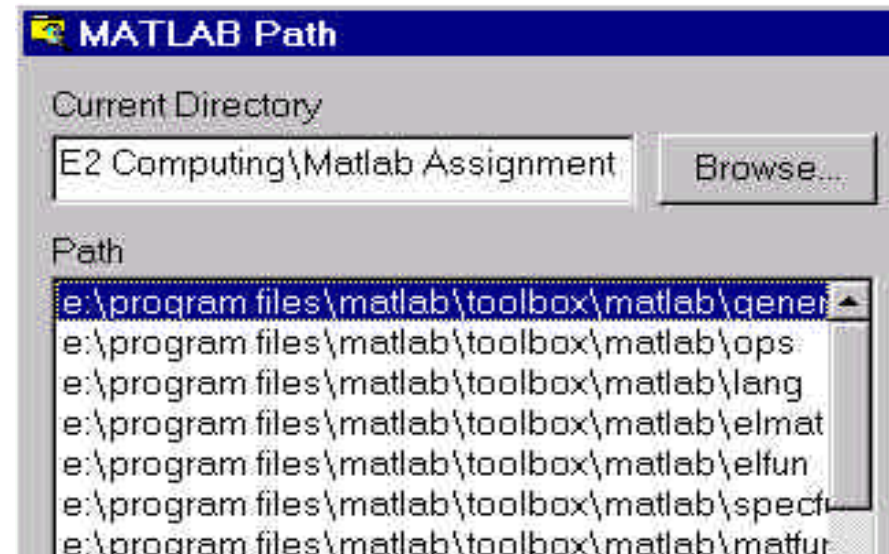
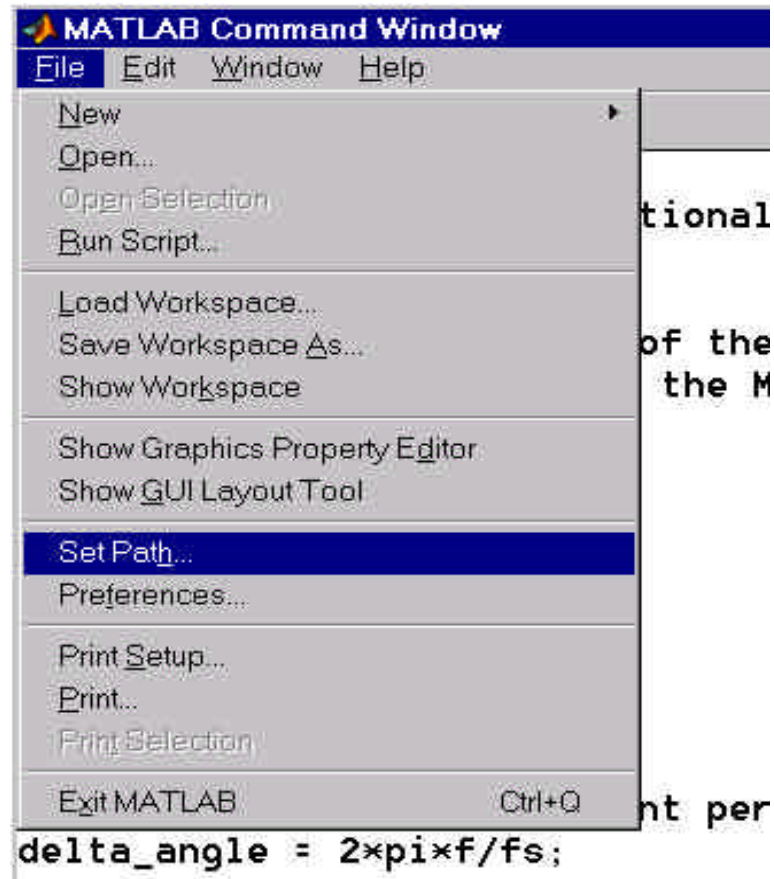
```
% create sine wave  
[t,sinewave]=sinegen(f,fs,4);
```

```
% plot it  
plot(t,sinewave);  
grid
```

```
% label axes  
xlabel('Time (in sec)');  
ylabel('Amplitude');  
title('Sinewave at 1kHz');
```

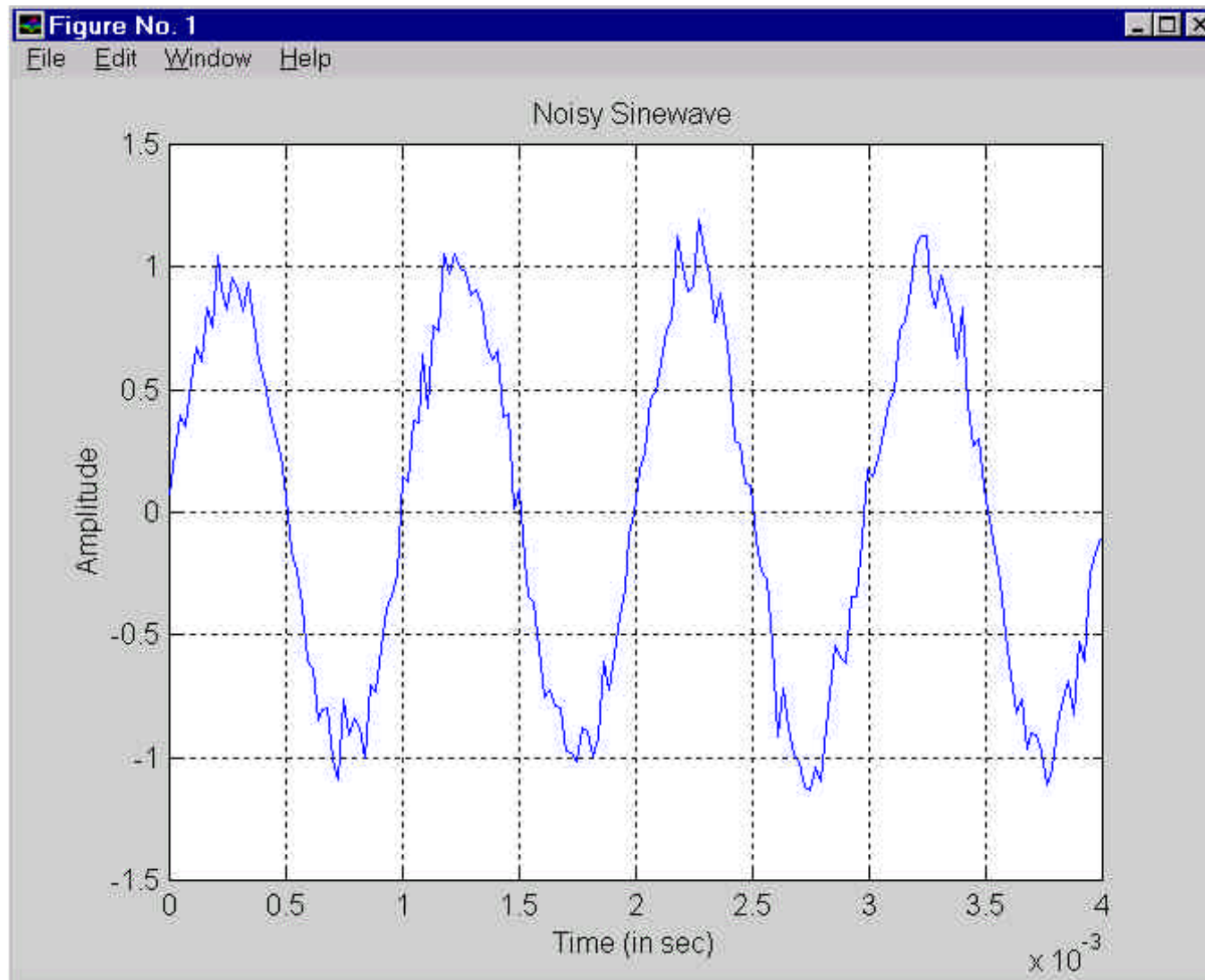


Must use Add Path (or Set Path)



- ◆ Must use Set Path manual or addpath command to make new .m files visible!

Lab 1 (con't) - Noisy Sinewave



Logical Subscripting



- ◆ The logical vectors created from logical and relational operations can be used to reference subarrays.
- ◆ Suppose X is an ordinary matrix and L is a matrix of the same size that is the result of some logical operation. Then $X(L)$ specifies the elements of X where the elements of L are nonzero.
- ◆ Suppose:

```
x = 2.1 1.7 1.6 1.5 NaN 1.9 1.8 1.5 5.1 1.8 1.4 2.2 1.6 1.8
```

```
» x = x(finite(x))
```

```
x = 2.1 1.7 1.6 1.5 1.9 1.8 1.5 5.1 1.8 1.4 2.2 1.6 1.8
```

Logical Subscripting in action



- ◆ Now there is one observation, 5.1, which seems to be very different from the others. It is an **outlier**. The following statement removes **outliers**, in this case those elements **more than three standard deviations** from the mean.

```
x = x(abs(x-mean(x)) <= 3*std(x))
```

```
x = 2.1 1.7 1.6 1.5 1.9 1.8 1.5 1.8 1.4 2.2 1.6 1.8
```

Structures in MATLAB



- ◆ Structures are multidimensional MATLAB arrays with elements accessed by textual *field designators*. For example,

```
S.name = 'Ed Plum';  
S.score = 83;  
S.grade = 'B+'
```

- ◆ creates a scalar structure with three fields.

```
S =  
    name: 'Ed Plum'  
    score: 83  
    grade: 'B+'
```

- ◆ an entire element can be added with a single statement.

```
S(3) = struct('name','Jerry Garcia',...  
              'score',70,'grade','C')
```

Assignment: Image Warping



- ◆ Four Tasks:
 - ❖ Image rotation
 - ❖ Image shearing
 - ❖ Edge detection
 - ❖ Image blurring
- ◆ Deadline
 - ❖ Monday 9th November, noon (Level 6 office)
- ◆ Deliverables:-
 - ❖ Well commented listing of your MATLAB files
 - ❖ Evidence that it works (i.e. hardcopy for each of the special effects)
 - ❖ Floppy disk containing a ready-to-try copy of your programmes

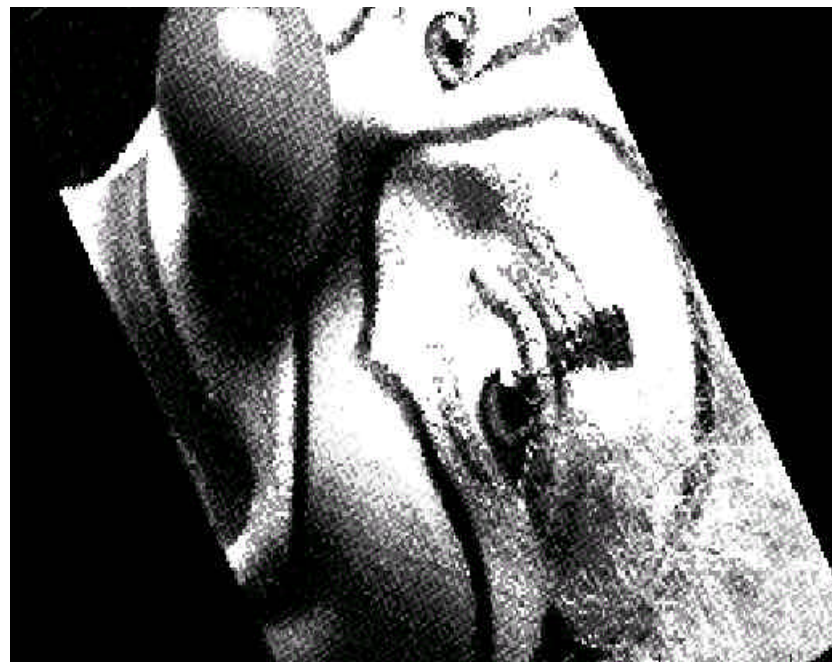
Problem 1: Rotation (1)



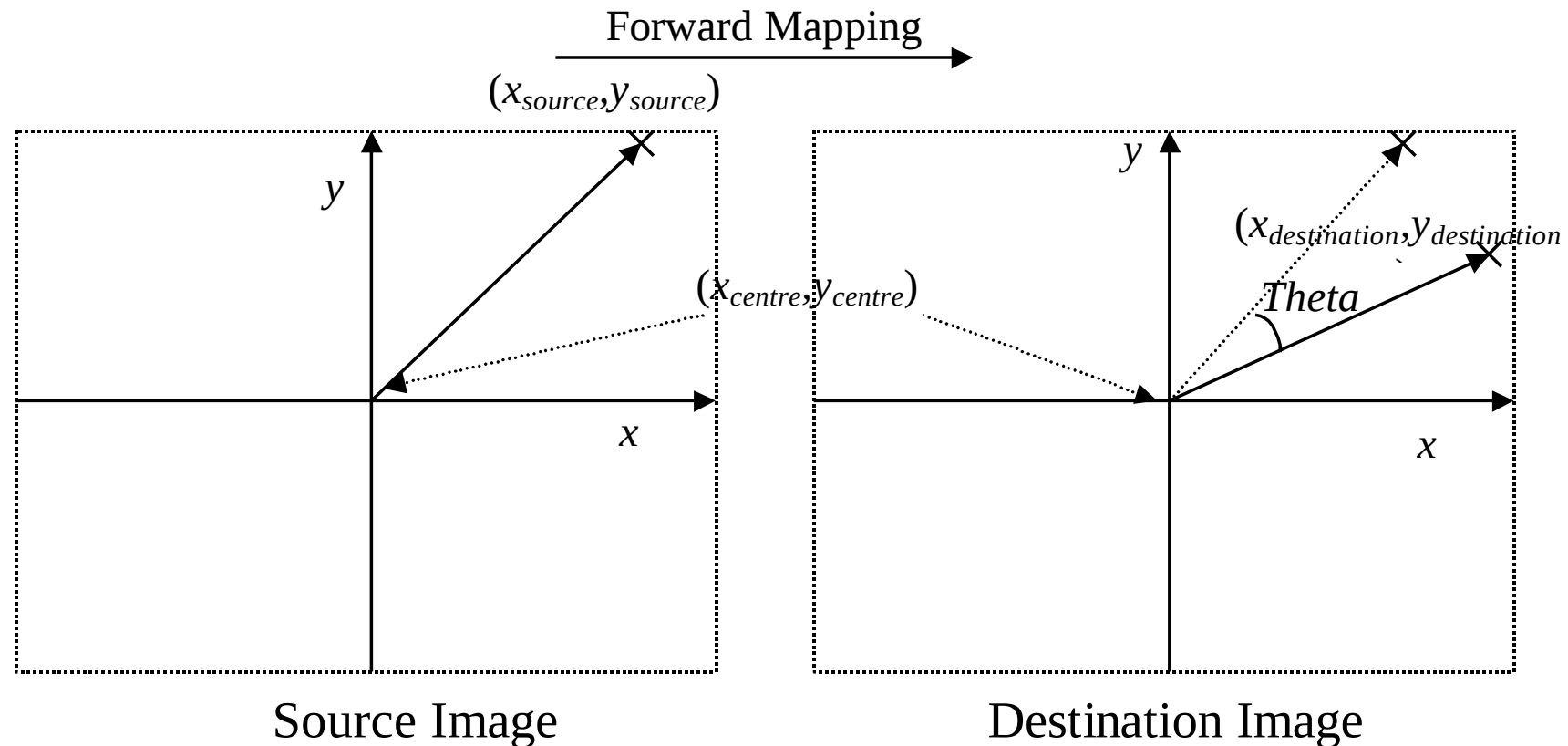
`Show(clown)`



`Show(rotate(clown, pi/3))`



Problem 1: Rotation (2)

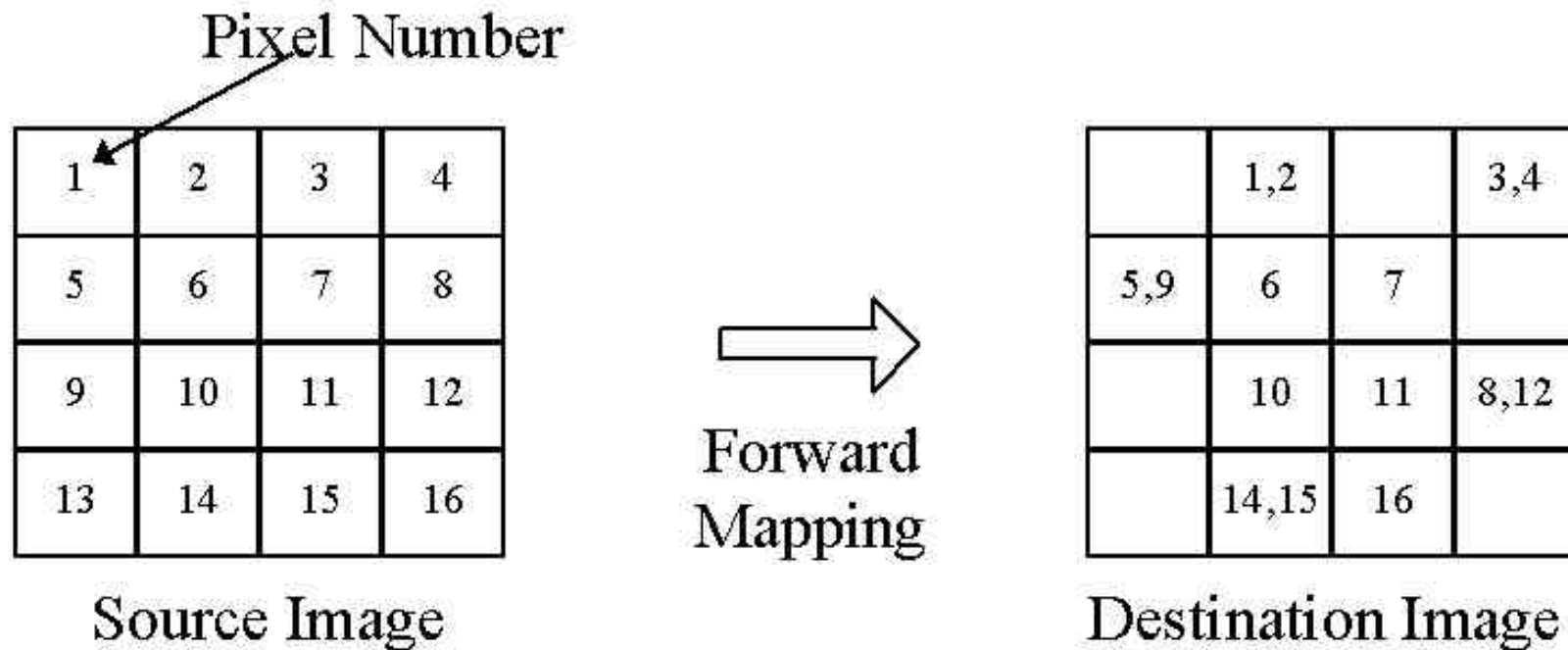


$$\begin{pmatrix} x_{destination} \\ y_{destination} \end{pmatrix} = \begin{pmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{pmatrix} \left(\begin{pmatrix} x_{source} \\ y_{source} \end{pmatrix} - \begin{pmatrix} x_{centre} \\ y_{centre} \end{pmatrix} \right) + \begin{pmatrix} x_{centre} \\ y_{centre} \end{pmatrix}$$

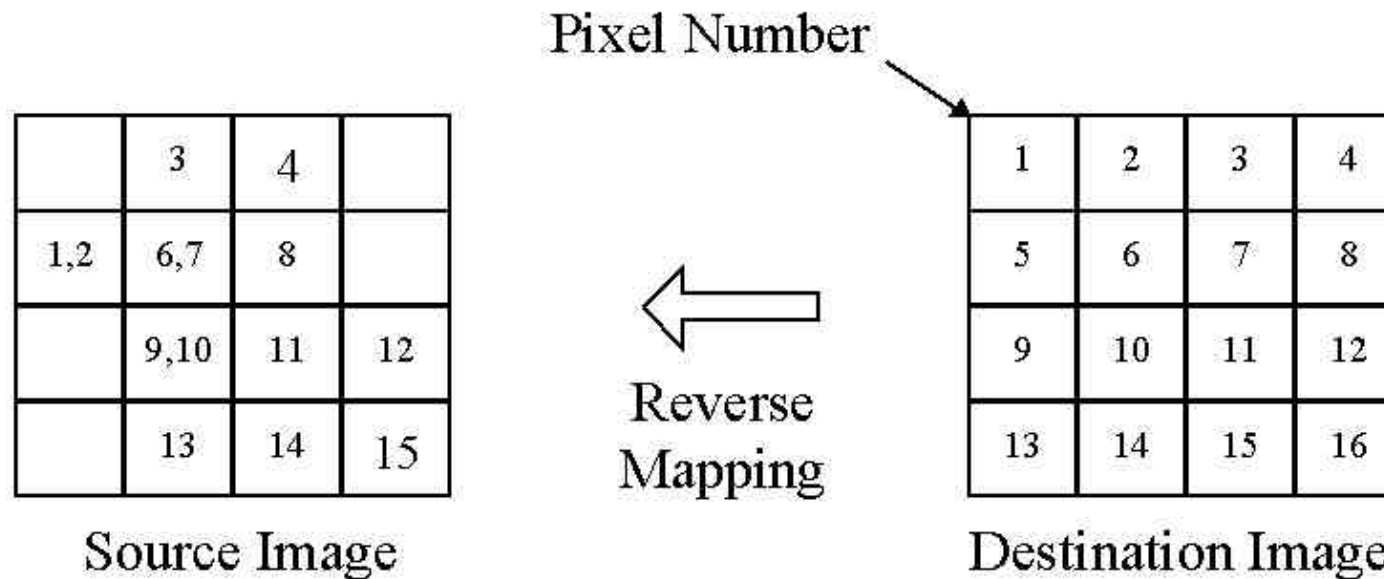
Problem 1: Rotation (2)



For each pixel in the source image {
 Work out the destination pixel location using the forward mapping
 equation.
 Paint that destination pixel with the source image value.
}



Problem 1: Rotation (3)



$$\begin{pmatrix} x_{source} \\ y_{source} \end{pmatrix} = \begin{pmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{pmatrix}^{-1} \left(\begin{pmatrix} x_{destination} \\ y_{destination} \end{pmatrix} - \begin{pmatrix} x_{centre} \\ y_{centre} \end{pmatrix} \right) + \begin{pmatrix} x_{centre} \\ y_{centre} \end{pmatrix}$$

For each pixel in the destination image {
 Work out where the pixel maps to in the source image, using the
 reverse mapping equation
 Paint the destination pixel with that source pixel value.
}

Problem 2 & 3: Shearing & Edge Detection

