CSE185
Introduction to Computer Vision
Lab 02: For Loop Operation

Instructor: Daniel Leung
TA: Mohammadkazem Ebrahimpour
Xueqing Deng

If statement

- If statement

```
if EXPRESSION
    ...
end
```

- If-else statement

```
if EXPRESSION
    ...
else
    ...
end
```

```
if EXPRESSION
    ...
elseif EXPRESSION
    ...
else
    ...
end
```

Loop

- For loop

```
for i = 1:10
    ...
end
```

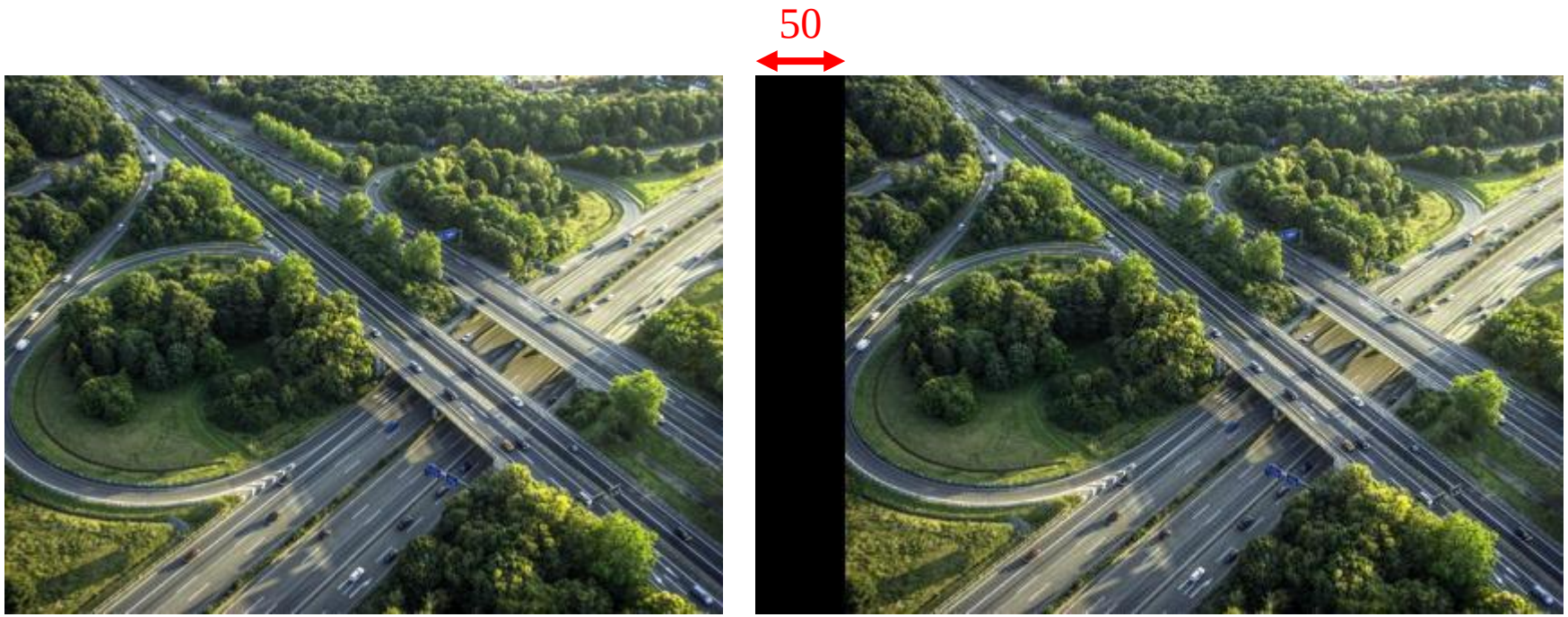
- While loop

```
while EXPRESSION
    ...
end
```

Translation

- Shift image by 50 pixels:

$$I_2(y,x) = 0 \qquad if\ x \leq 50$$
$$I_2(y,x) = I_1(y,x-50) \quad if\ x > 50$$



Translation

- Shift image by 50 pixels
- Use For loop

```
I2 = zeros(300, 400 + 50, 3, 'uint8');

for y1 = 1 : 300
    for x1 = 1 : 400

        y2 = ???
        x2 = ???

        I2(y2, x2, :) = I1(y1, x1, :);

    end
end
```

Rotation

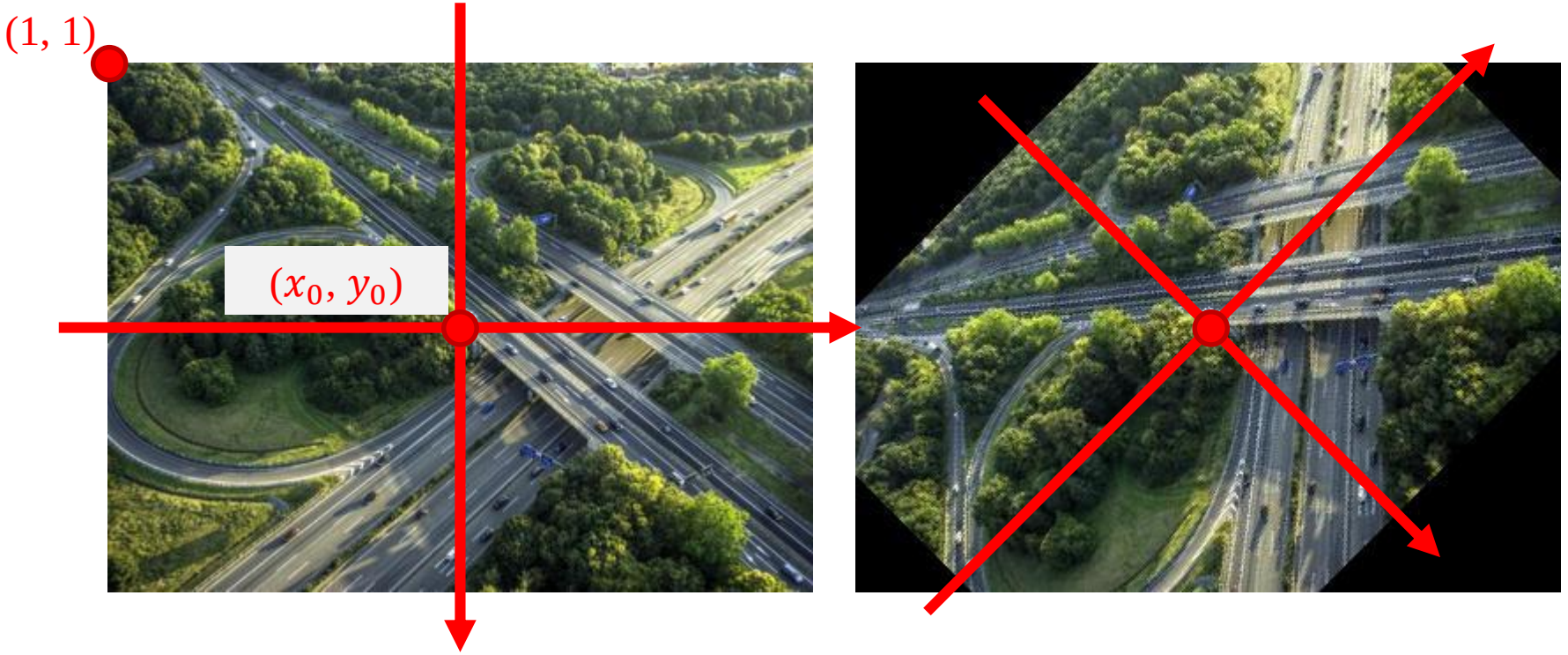
- Rotate image 45 degree: (do NOT use `imrotate()`)

$$\begin{pmatrix} x_2 \\ y_2 \end{pmatrix} = \begin{pmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \end{pmatrix}$$

Rotation

- Rotate image 45 degree:
  - shift origin (x_0, y_0) to the center of the input image

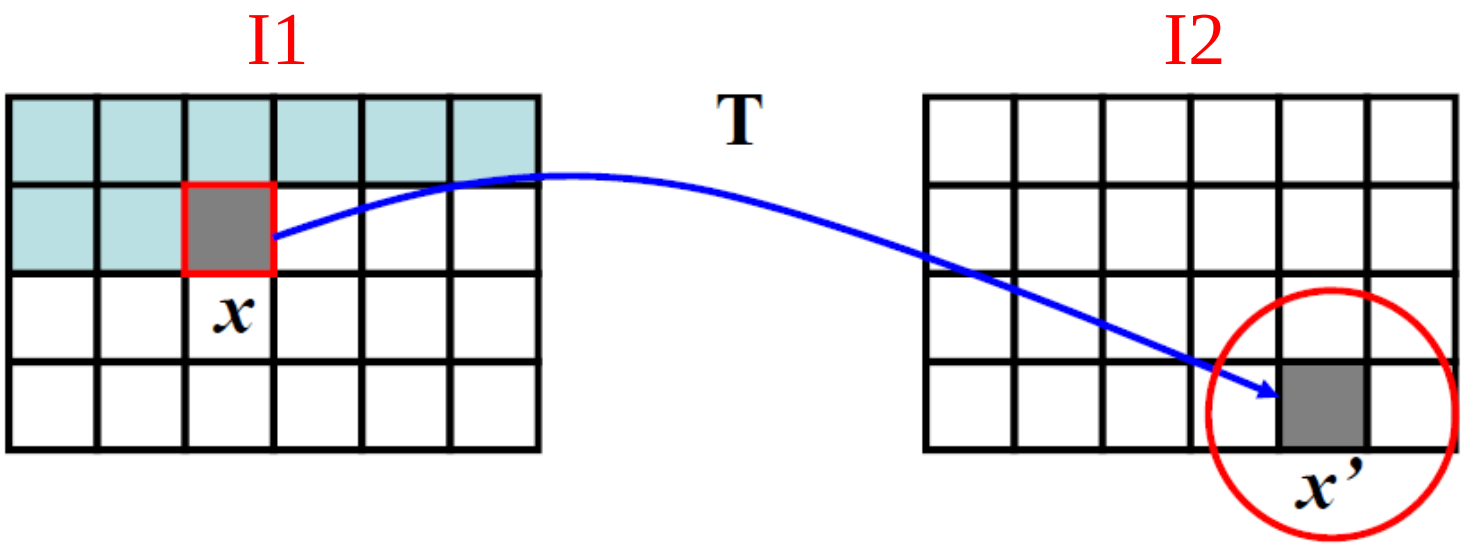
$$x_2 = \cos(\theta) \cdot (x_1 - x_0) + \sin(\theta) \cdot (y_1 - y_0) + x_0$$
$$y_2 = -\sin(\theta) \cdot (x_1 - x_0) + \cos(\theta) \cdot (y_1 - y_0) + y_0$$



Forward Warping

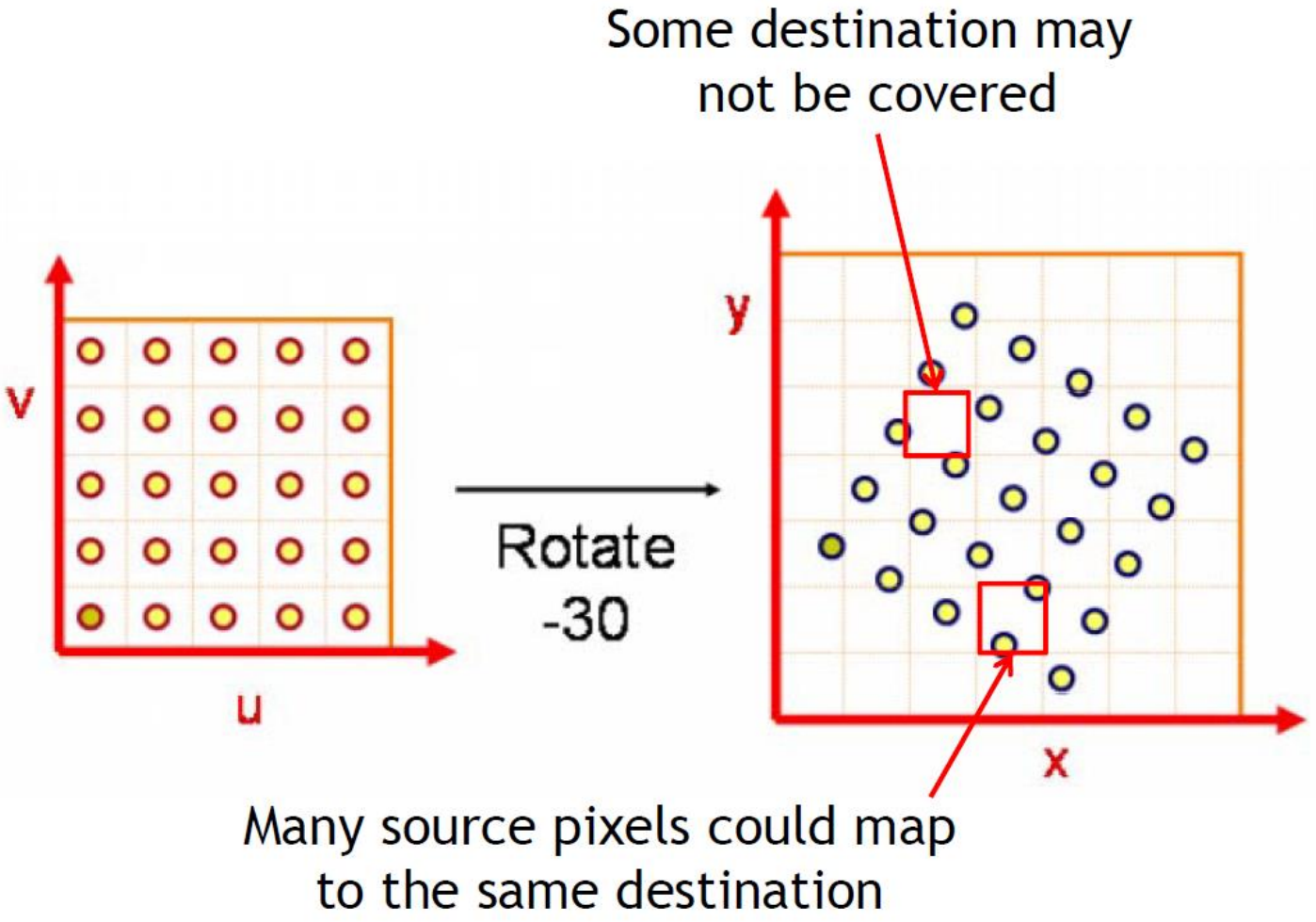
- Suppose I1 is input image, I2 is warped image
- Pseudocode:

```
for each pixel (y1, x1) in I1:
    (y2, x2) = Rotate(y1, x1)
    if (y2, x2) is inside I2:
        I2(y2, x2) = I1(y1, x1)
    end
end
```



Forward Warping

- Forward warping will produce “holes”:



Forward Warping

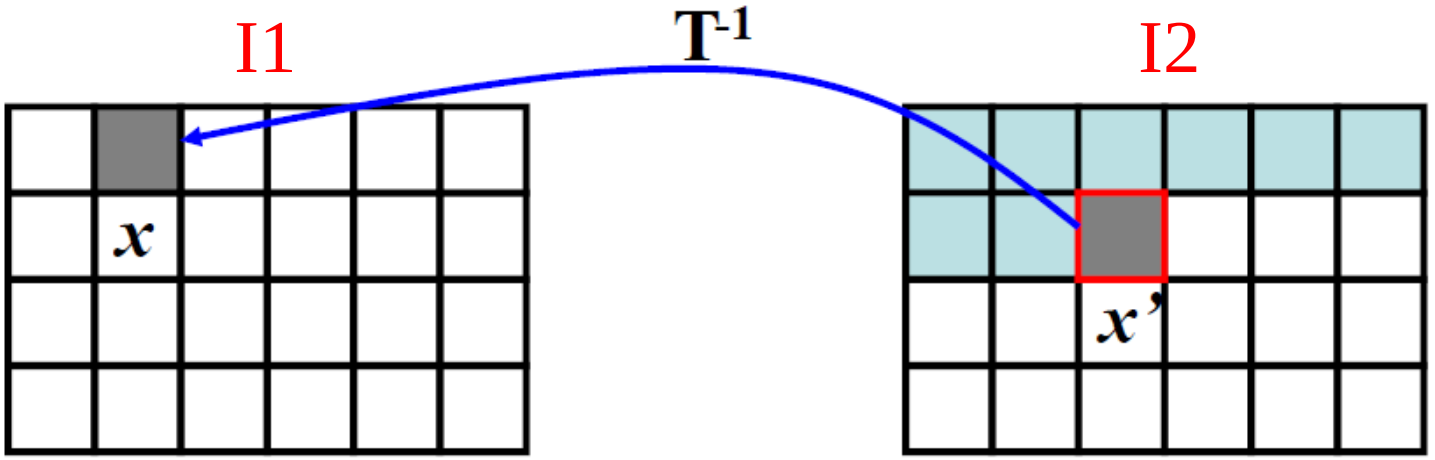
- Forward warping will produce “holes”:

Backward/Inverse Warping

- Suppose I1 is input image, I2 is warped image:

```
for each pixel (y2, x2) in I2:
    (y1, x1) = Rotate^-1(y2, x2)
    if (y1, x1) is inside I1:
        I2(y2, x2) = I1(y1, x1)
    end
end
```

Rotate^-1 = Inverse warping



Backward/Inverse Warping

Hints

- The inverse of a rotation matrix is still a rotation matrix:

$$\begin{pmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{pmatrix}^{-1} = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix}$$

- Use `cosd()` and `sind()` if your angle is in degree, use `cos()` and `sin()` if your angle is in radian.
- Use nearest neighbor sampling:

```
(y1, x1) = Rotate^-1(y2, x2)
y1 = round(y1);
x1 = round(x1);
If( 1 <= y1 && y1 <= H && 1 <= x1 && x1 <= W )
    ...
```

Implemented by sin and cos

- Your rotated image will look similar to:

```
imrotate(I1, 45, 'nearest', 'crop')
```

Image Processing in MATLAB

- Horizontally flip image: do NOT use `flip()`

- Use For loop, or submatrix indexing:

```
>> vec = [10, 20, 30, 40, 50];
>> vec(5:-1:1)
ans =
    50    40    30    20    10
```

Specify the step size to -1

Scaling

- Down-sample image by a factor of 2: do NOT use `imresize()`

- Your result should look like

```
imresize(I1, 0.5, 'nearest')
```

- You should use For loop or submatrix indexing

TODO

1. Shift 01.jpg by 25 pixels in the positive vertical direction, and save as translate.jpg
2. Rotate 01.jpg by 60 degree using **forward warping**, and save as rotateF.jpg
3. Rotate 01.jpg by 60 degree using **backward warping**, and save as rotateB.jpg
4. Vertical flip 01.jpg, and save as flip.jpg
5. Down-sample 01.jpg by scale of 2, and save as scale.jpg
6. Save your codes as lab02.m
7. Upload all output images and your lab02.m (in a zip file)

- Do NOT use any built-in function (e.g., `imrotate`, `imresize`, `flip`)