Day 1
Transforming Pictures

Pennsylvania State University

Look into a digital image

red channel

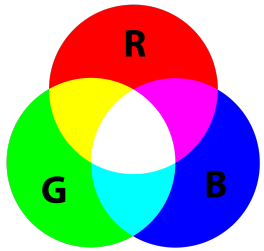
238	238	237	235	236	236	235	235	236	237
237	237	238	237	238	238	236	237	236	237
237	237	238	238	239	239	237	238	236	237
239	238	238	238	237	238	237	239	236	238
238	239	235	238	240	238	244	238	237	238
241	243	243	237	237	247	241	236	236	245
244	242	242	242	238	233	248	244	242	236
239	236	233	233	239	240	236	236	236	237
246	250	251	255	253	250	248	243	245	245
253	253	255	254	247	253	255	255	246	245

green channel

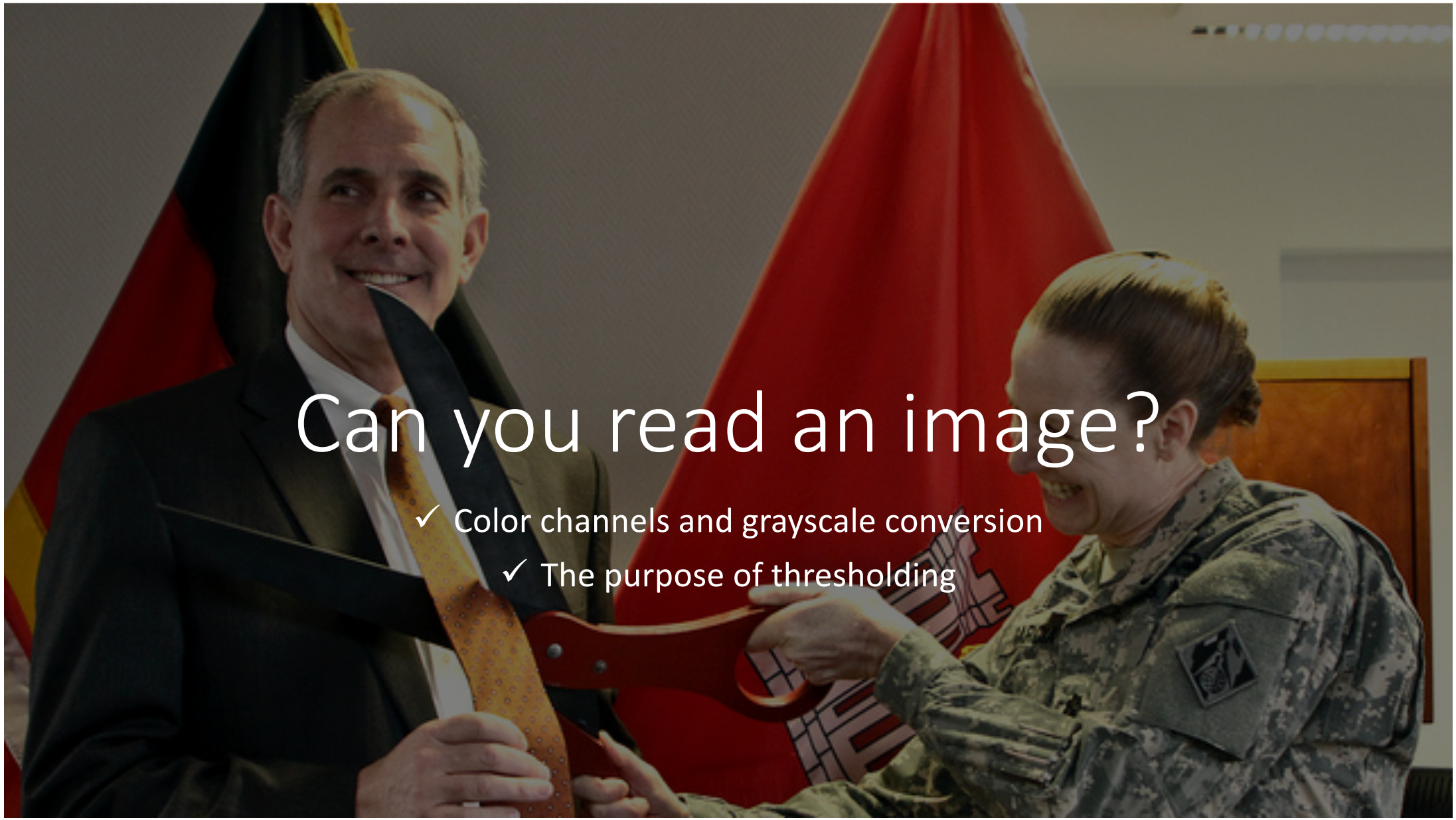
232	232	231	229	230	230	229	229	230	231
231	231	232	231	232	232	230	231	230	231
231	231	232	232	233	233	231	232	230	231
233	232	232	232	232	232	231	233	230	232
239	235	229	233	234	229	235	229	231	232
232	234	234	232	233	244	240	234	231	236
236	233	234	234	239	230	243	237	234	230
235	232	232	234	231	232	225	227	231	234
242	242	240	246	240	242	244	239	241	239
252	251	255	255	255	255	254	245	240	240

blue channel

196	196	195	193	194	194	193	193	194	195
195	195	196	195	196	196	194	195	194	195
195	195	196	196	197	197	195	196	194	195
197	196	196	196	194	196	195	197	194	196
195	197	195	195	200	196	202	196	197	198
193	195	201	200	186	199	196	193	191	197
197	192	195	198	199	189	201	195	197	196
197	197	201	200	194	196	193	194	193	193
195	203	208	205	206	205	206	201	204	203
195	202	216	215	197	205	215	207	192	185



RGB color model
(value: 0 ~ 255)



Can you read an image?

- ✓ Color channels and grayscale conversion
- ✓ The purpose of thresholding

Component of colors: Red, Green, Blue

red channel

Check code
line 34 ~ 38
in
A_read_img.m

green channel

Check code
line 40 ~ 44
in
A_read_img.m

blue channel

Check code
line 46 ~ 50
in
A_read_img.m

grayscale image

Check code
line 52 ~ 55
in
A_read_img.m

= Mix of 40% Red + 40% Green + 20% Blue

Thresholding: simplification of an image

image thresholding

Check <u>code lines 58 ~ 64</u>
in **A_read_img.m**

What does it happen to the image if you change value of *threshold* (line 59) ?

Set to 255 if value $> threshold$
Set to 0 if value $\leq threshold$

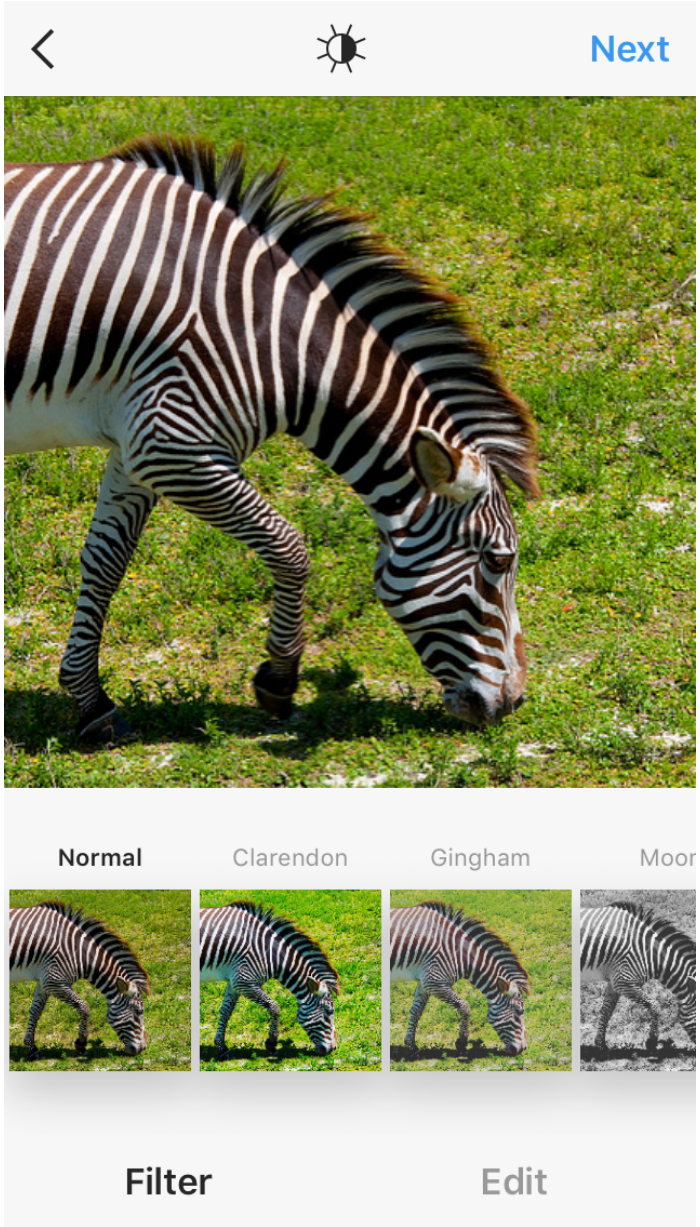
✓ Understand Blur and Median filter

Need to attenuate image noise

Noise Reduction – Blur and Median

Try to compare output of
Blur (line 34 ~ 37) and Median (line 39 ~ 42)
in **B_noise_reduction.m**

What does it happen to the image
if you change filter size (line 32) of
Blur and Median filters
in **B_noise_reduction.m**?



Make filters

Understand an impact of addition and multiplication on images

Linear Filters: Identity

filter result

Try to assign *identity* (line 33)
to *filter* (line 88)
and display an image
in **C_linear_filter.m**

0	0	0
0	1	0
0	0	0

Take a center pixel
among neighbors

Linear Filters: Blur (averaging)

filter result

Try to assign
average (line 38)
or *average2* (line 42)
to *filter* (line 88)
and display an image
in **C_linear_filter.m**

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Take an average of
neighbor pixels

Linear Filters: Blur (weighted averaging)

filter result

Try to assign
weighted_average (line 48)
or *weighted_average2* (line 52)
to *filter* (line 88) and
display an image
in **C_linear_filter.m**

$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Take an average of neighbor pixels (some are important than others)

Linear Filters: outline (Identity – averaging)

filter result

Try to assign *outline* (line 58)
to *filter* (line 88) and
display an image
in **C_linear_filter.m**

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Remove the impact
of neighbor pixels

Linear Filters: sharpen (Identity + outline)

filter result

Try to assign *sharpen* (line 62)
to *filter* (line 88) and
display an image
in **C_linear_filter.m**

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

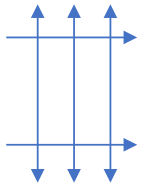
Fortify outline in an image

Linear Filters: Edge operator (Prewitt)

filter result

Try to assign
prewitt_horizontal (line 70)
to *filter* (line 88)
in **C_linear_filter.m**

-1	-1	-1
0	0	0
1	1	1

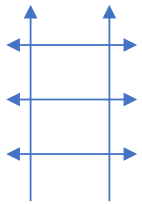


Find horizontal differences
and averaging them equally

filter result

Try to assign
prewitt_vertical (line 66)
to *filter* (line 88)
in **C_linear_filter.m**

-1	0	1
-1	0	1
-1	0	1



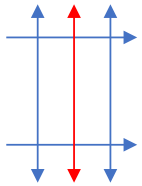
Find vertical differences
and averaging them equally

Linear Filters: Edge operator (Sobel)

filter result

Try to assign
sobel_horizontal (line 78)
to *filter* (line 88)
in **C_linear_filter.m**

-1	-2	-1
0	0	0
1	2	1

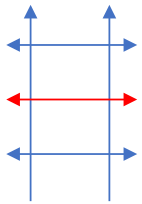


Find horizontal differences
and averaging them
(center is more important)

filter result

Try to assign
sobel_vertical (line 74)
to *filter* (line 88)
in **C_linear_filter.m**

-1	0	1
-2	0	2
-1	0	1



Find vertical differences
and averaging them
(center is more important)

Linear Filters: Your own filter!

Try to create *your_own_filter* (line 83)
and assign it to *filter* (line 88),
then display an image
in **C_linear_filter.m**

Can you find edges in an image?

- ✓ Smoothing, Differentiation, Suppression, and Thresholding

Canny Edge Detection
– 1. Smoothing and Differentiation

smoothing + sobel operator

Check <u>code lines 39 ~ 79</u>
in **D_edge_detection.m**

What does it happen to the image
if you change value of
sigma (line 33) ?

- Blur smooths complex edges in an image
- Edge operator presents the intensity of differences in edges among neighbor pixels

Canny Edge Detection – 2. Suppression

Non-Maximal Suppression + Hysteresis thresholding

Check <u>code lines 81 ~ 189</u>
in **D_edge_detection.m**

What does it happen to the image if you change value of *threshold_alpha* (line 34) and *threshold_beta* (line 35) ?

- Edge pixels should have the highest intensity among neighbor pixels
- Otherwise, pixels have to be connected with edge pixels if they are part of edges

Step-by-step Output comparison in Canny Edge Detection

smoothing + sobel operator

Non-Maximal Suppression + Hysteresis thresholding

A way to find circles in an image

- ✓ Edge detection, Transformation, and Voting

Circle Detection – Edge detection

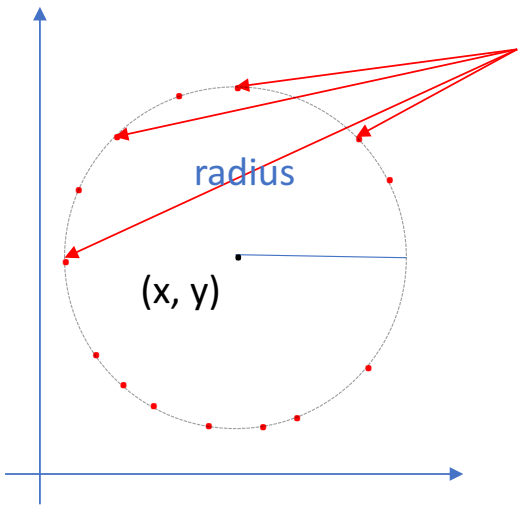
Edges

Check code lines 38 ~ 45
in **E_circle_detection.m**

- Determine edges at first
- By comparing edge pixels to pre-defined template, we can figure out shapes
- Try to change *sigma* (line 33)

Let's find out circles!

Circle Detection – Voting Hough Transform



Count the number of edge pixels for a specific circle

Circles in edges

Check <u>code lines 47 ~ 106</u>
in **E_circle_detection.m**

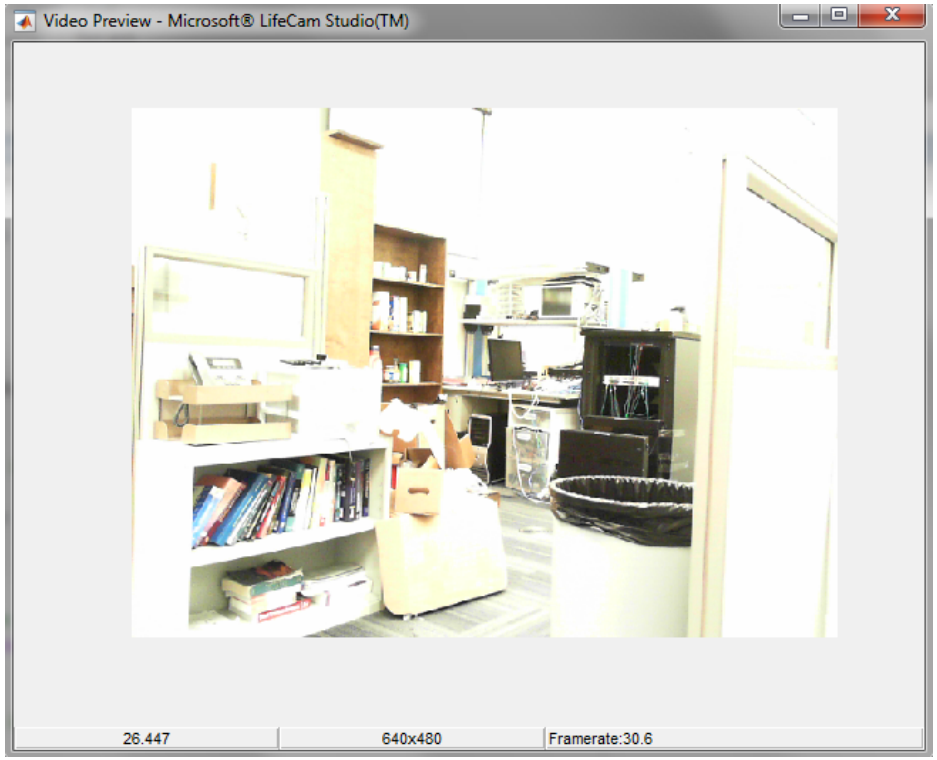
- If voting count exceeds a defined threshold, then you will find a circle at specific position!

- Try to change *min_radius* (line 34), *max_radius* (line 35), and alpha (line 36)

Let’s connect computer to real world

- ✓ Use USB Camera to take an image
- ✓ Practice image processing which you have learnt

Try image processing with your own image



Try to copy & paste

- Line 34 ~ 64 in **A_read_img.m**
- Line 31 ~ 42 in **B_noise_reduction.m**
- Line 32 ~ 91 in **C_linear_filter.m**
- Line 32 ~ 189 in **D_edge_detection.m**
- Line 32 ~ 106 in **E_circle_detection.m**

to line 32 in **F_test_camera.m**

Original Image will be stored at
imgs/f_test_camera/original.jpg

Question & Answers

Please save your preferred practices as images ☺

Thank you!