Introduction to Robotics

Felipe P. Vista IV

Class Admin Matters

Grading

➢ Attendance 5%

Name (Original Name)	User Email	Join Time	Leave Time	Duration (Minutes)
[redacted]	[redacted]	4/12/2021 9:12	4/12/2021 10:14	62
[redacted]	[redacted]	4/12/2021 9:12	4/12/2021 9:14	3
[redacted]	[redacted]	4/12/2021 9:12	4/12/2021 9:14	3
[redacted]	[redacted]	4/12/2021 9:12	4/12/2021 9:14	3
[redacted]	[redacted]	4/12/2021 9:12	4/12/2021 9:14	3
[redacted]	[redacted]	4/12/2021 9:12	4/12/2021 9:14	3
[redacted]	[redacted]	4/12/2021 9:13	4/12/2021 9:13	1
[redacted]	[redacted]	4/12/2021 9:13	4/12/2021 9:14	2
[redacted]	[redacted]	4/12/2021 9:14	4/12/2021 9:14	1
[redacted]	[redacted]	4/12/2021 9:14	4/12/2021 9:14	1
[redacted]	[redacted]	4/12/2021 9:14	4/12/2021 10:14	60

Bad ZOOM User Name (Absent)

- Iphone → Not your name
- SiAko 202100001 → Wrong order
- SiAko → Name only
- 202100001 → ID Num only

ZOOM User Name (Present)

- University ID Num_Name
- 202100001 SiAko → GOOD (Present)

Name (Original Name)	User Email	Total Duration (Minutes)
[redacted]	[redacted]	62
[redacted]	[redacted]	63
[redacted]	[redacted]	62
[redacted]	[redacted]	62
[redacted]	[redacted]	63
[redacted]	[redacted]	62
[redacted]	[redacted]	63

Student Responsibilities

- Download/Install ZOOM app for online lecture
  - *Zoom profile must be your **OASIS ID+name** similar to OASIS*
  - *Ex.: **202061234 YourName***
  - *If you are asked, but no reply, then you'll be out of zoom & mark **absent***
- Regularly login, check OLD IEILMS for updates, notifications
  - *https://ieilmsold.jbnu.ac.kr*
  - *Presentations & lecture videos will be uploaded after class*
- Regularly check Kakao Group Chat for class
  - *Everybody must have a Kakao talk account*
  - *Search & add account "**botjok**", introduce yourself and name of class ("Robotics"), then you will be added to the group chat*

Intro To Robotics

IMAGE PROCESSING

Intro

- Distance sensor of your self-driving car detect object 100m ahead
  - *Following car in front at a safe distance?*
    Or has pedestrian jumped into the road?
- Algorithms so far studied
  - **Physical properties** : *distance, angle, reflectance*
- More complex tasks need detailed info of surroundings
  - *Especially if robot must work autonomously in unfamiliar environments*
- Vision is obvious way for humans to make sense ~~send~~ environment
- Visual system (eyes + brain)
  - *Is really complex & we just take it for granted*
  - *Easily distinguish between moving car and pedestrian crossing road →*
    react quickly

Intro

- Automatically record images using camera for almost 200 years
  - *But interpreting the tasks remained a job for humans*
- Automatic processing & interpretation of images
  - *Became possible with the use of computers*
- We are familiar with digital images
  - *Weather maps (satellites)*
  - *Medical images (X-ray, CT, MRI, ultrasound)*
  - *Photos (taken using smartphones or even digital cameras)*
- **Digital Image Processing (DIP)**
  - *One of most intensely studied field of computer science & engineering*
  - *But image processing systems not yet level of human visual system*

Intro

- We will learn **DIP**
  - *Algorithms for DIP*
  - *Their use in robotic systems*
  - **Enhancement** : *digital filters & histogram manipulation*
  - **Segmentation** : *edge detection*
  - **Feature recognition** : *detection of corners & blobs, identification of multi-features*
- Few educational robots use cameras
  - *Due to cost and computing power*
- Therefore we can implement image processing algorithms
  - *On a personal computer*
  - *Using images captured from digital cameras/ smartphones*

- Obtaining Images
- Overview of Digital Image Processing
- Image Enhancement
- Edge Detection
- Corner Detection
- Recognizing Blobs

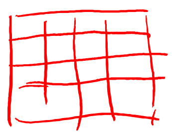
Obtaining Images (Optics)

Overview of **design considerations** for imaging systems

- Camera optical system consists of lens focus light on a sensor
- Wider the lens → more light that can be collected
  - *Important for systems that need to work in dark environments*
- Longer focal length → greater the magnification
  - **Focal length** : *distance between lens and sensor*
  - *Why professional photog's use heavy cameras w/ long lenses*
- Smartphone manufacturers facing a dilemma
  - *Users want thin & elegant phone → but it limits focal length of camera*
- For most robotic applications
  - *Magnification not worth required size & weight to achieve long focal length*

Resolution

- Images used to be captured on film by chemical reaction
  - *Light hitting plastic covered with emulsion of tiny silver particles*
  - *(In principle) Each particle react independently* → *extremely high resolution*
- In digital devices
  - *Light captured by semiconductor devices i.e. charged coupled devices(CCD)*
- Digital cameras
  - *There is a chip w/ fixed number of elements in rectangular array*
  - *Each element measure light intensity independently*
    - These measurements are called **pixels**
  - *More pixels captured* → *higher resolution*
  - *Cheap cameras in smartphones can capture millions of pixels (single image)*



Resolution

- Problem with high resolution
  - *Large amount of memory needed to store it*
- **Ex. : High resolution display**
  - *w/ 1920 x 1080 pixels and 8 bits/pixel to store intensity (0 – 255)*
    - Single image about 2MB memory
  - *Embedded computer can analyze a single image*
  - *But mobile robot need to store several images per second*
- Computing power to analyze images
  - *More important than amount of memory required*
- Image processing algo need computers to compute each pixel
  - *Not a problem for astronomers analyzing images from space telescope*
  - *Problem for self-driving car that make decisions in fraction of a second*

B + W

pixels

1 byte

8 bits

1 byte

bits

R+G = Yellow ?
(255, 255, 0) ↗
(0, 0, 0) → Black
(255, 255, 255) → white
1920 x 1080 px
1 byte = 2MB
RGB = 6MB
B/W = 1 byte/pixel
RGB = 3 byte/px
RGB(255, 255, 0)

Color

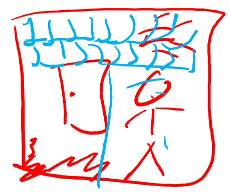
- Our visual system can distinguish visible light
  - **Visible light** : *range of wavelengths*
  - *See different wavelengths as different colors*
    - **Red** : longer wavelengths; **Violet** : shorter wavelengths
  - *Human eye can distinguish millions of different colors but name only a few*
  - *Color is one of primary tools we use to identify objects*
- **Sensors** can measure outside range outside "visual light"
  - **Infrared** : *longer wavelengths*; **Ultraviolet** : *shorter wavelengths*
  - *Infrared images important in robotics (people, cars as "hot objects")*
- Problem with color is it triples storage & processing req'ts
  - 3 **primary colors** *for all colors : red, green, blue* → **RGB** *(3 bytes/pixel)*
  - *Single color 1920 x 1080 image* → *6MB storage & 3x as long processing*

- Obtaining Images
- Overview of Digital Image Processing
- Image Enhancement
- Edge Detection
- Corner Detection
- Recognizing Blobs

Overview of Digital Image Processing

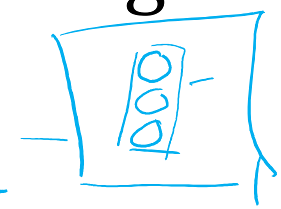
- Optical system of robot
  - *Capture images as rectangular arrays of pixels*
  - *But robot tasks expressed in terms of objects in the environment*
    - enter room using door, pick item off shelf, stop if pedestrian front of car
  - *How to go from pixels to objects?*

- **Image enhancement** is the **first stage**
  - *Images has noise due to optics & electronics*
  - *Lighting in environment can make image too dark or washed out*
  - *Image may be accidentally rotated or out of focus*
  - *All these problems independent of the content*



- Image enhancement algorithms typically work by
  - *Modifying values of individual pixels w/o regard to its meaning (the values)*

Overview of Digital Image Processing



- Image enhancement is difficult
  - *Since no formal definition what it means to enhance image*
  - *Blurred blob might be dirt on camera lens or unknown galaxy*
  - *Later see two approach*
    - **Filtering** : remove noise by *replacing* pixel w/ neighboring pixels
    - **Histogram manipulation** : *modify* contrast & brightness
- Objects distinguished by lines, curves & areas
  - *A door has 3 straight edges of rectangle w/ one short side missing*
  - *A traffic light consists of three bright disks one above the other*
- Before a door or traffic light can be identified
  - *Image processing algo must determine w/c pixels determine lines, edges, etc*
  - *This process is called* **segmentation** *or* **feature extraction** **(2nd phase)**

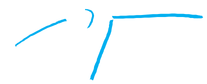
Overview of Digital Image Processing

- Process is called segmentation or feature extraction
  - *Since algorithm must determine w/c pixels are part of a segment of image*
- Segmentation easy if edges, lines, curves are uniform
  - *But not the case in real images*
  - *Edge slanted in arbitrary angle; missing pixels or obscured by shadow*
  - *Familiar with* **captchas**
    - Letters are intentionally distorted so auto recognition is very difficult
    - But humans can still easily identify
- Enhancement algorithms make segmentation easier
  - *Can be by filling in missing pixels but may introduce artificial segments*
- Study later a segmentation technique
  - *Which is a filter that detects edges in an image*

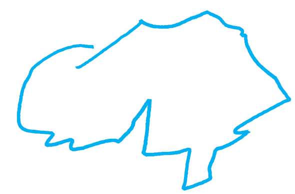
Overview of Digital Image Processing

DIP:
- Enhancement → Segmentation / Feature Ext → Object Recognition

- **Object recognition**
  - *The final phase of image processing*
- Two algorithms for detecting corners to be studied later
  - *Locating intersection of two edges, and*
  - *Counting neighbours with similar intensities*



- Algorithm for recognizing **blobs**
  - *Areas whose pixels have similar intensities*
  - *But bounded by regular features like lines and curves*



- Recognition of object defined by more than one feature
  - *Such as a door defined by two edges*
    - That are at an arbitrary distance from each other

Image Processing

- Obtaining Images
- Overview of Digital Image Processing
- Image Enhancement
- Edge Detection
- Corner Detection
- Recognizing Blobs

Image Enhancement

- (1.a) image of a rectangle
  - *Intensity is uniform horizontally*
  - *Shaded dark to light from top to bottom*
- (2.a) representation of (1.a)
  - *As a 6 x 10 array of pixels*
  - *Each pixel's light intensity in range 0 - 100*

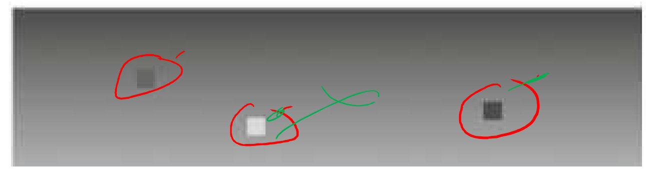
(1.a) Image w/o noise

	0	1	2	3	4	5	6	7	8	9
0	10	10	10	10	10	10	10	10	10	10
1	20	20	20	20	20	20	20	20	20	20
2	30	30	30	30	30	30	30	30	30	30
3	40	40	40	40	40	40	40	40	40	40
4	50	50	50	50	50	50	50	50	50	50
5	60	60	60	60	60	60	60	60	60	60

(2.a) Pixel array w/o noise

Image Enhancement

- (1.b) image that is not smooth
  - *Three points whose intensity different to its neighbor*
- (2.b) representation of (1.b)
  - *Intensity of pixels (2, 3), (3, 6) & (4, 4) different*
  - *Probably due to noise & not actual feature of photographed object*
- Doesn't matter where noise comes from
  - *Object itself, dust on lens, non-uniformity of sensor, noise in electronics*
  - *Impossible entirely get rid of noise (not sure if actual object feature or noise)*
    - *We do want to enhance image so noise not noticeable*



(1.b) Image w/ noise

	0	1	2	3	4	5	6	7	8	9
0	10	10	10	10	10	10	10	10	10	10
1	20	20	20	20	20	20	20	20	20	20
2	30	30	30	**20**	30	30	30	30	30	30
3	40	40	40	40	40	40	**10**	40	40	40
4	50	50	50	50	**90**	50	50	50	50	50
5	60	60	60	60	60	60	60	60	60	60

(2.b) Pixel array w/ noise

Spatial Filters

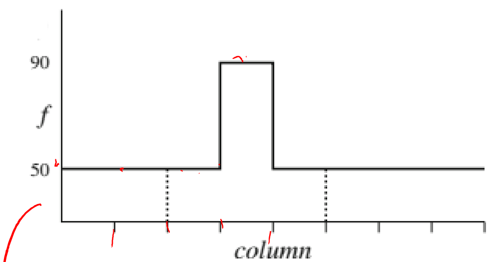
- Consider row 4 in (2.b)

  $$50, 50, 50, 50, 90, 50, 50, 50, 50, 50$$

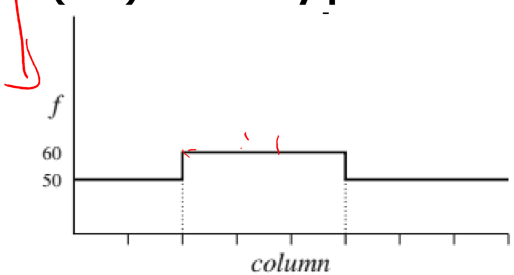
- (3.a) plot for intensity ***f*** of given row
  – *Clear that one pixel unlikely value since so different from neighbors*
  – *Program can make each pixel more like neighbors assigning w/ average of its & neighbors intensity*
- (3.b) for most pixels in the row
  – *Doesn't change values*: $(50+50+50)/3 = 50$
  – *But noise pixel & neighbors*: $(50+90+50)/3 \approx 60$
- Averaging gave two pixels "wrong" values
  – *Overall, image will be visually enhanced*

	0	1	2	3	4	5	6	7	8	9
0	10	10	10	10	10	10	10	10	10	10
1	20	20	20	20	20	20	20	20	20	20
2	30	30	30	**20**	30	30	30	30	30	30
3	40	40	40	40	40	40	**10**	40	40	40
4	50	50	50	50	**90**	50	50	50	50	50
5	60	60	60	60	60	60	60	60	60	60

(2.b) Pixel array w/ noise

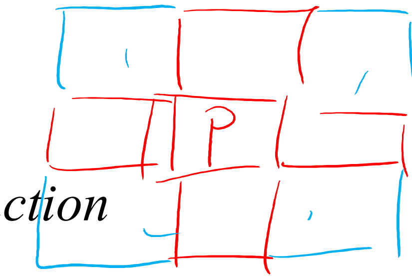


(3.a) Intensity plot before

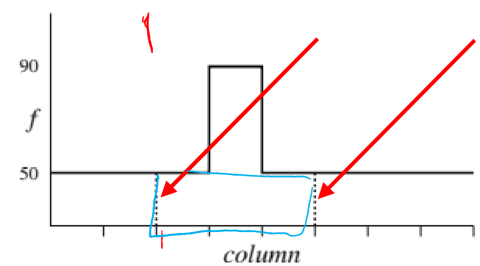


(3.b) Intensity plot after averaging

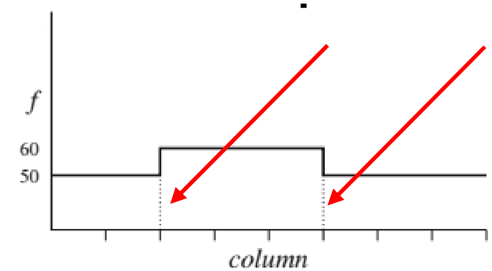
Spatial Filters



- Average of sequence of pixels
  - *Discrete version of integrating continuous intensity function*
  - *Integration smooths out local variation of function*
  - *Dotted lines (3.a) & (3.b) indicate 3-pixel sequence (bounded to same area)*
- Averaging operation for each pixel
  - Done through **spatial filter**
- For 2D array of pixels
  - *Filter represented by a 3 x 3 array*
    - Each element specify factor by which pixel and its neighbors will be multiplied
  - *Each pixel has 4 or 8 neighbors*
    - Depending if we include diagonal ones



(3.a) Intensity plot before



(3.b) Intensity plot after averaging

Spatial Filters

- The **box filter** is: $\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$
  - *Results of multiplications are added & then divided by 9 to scale back to intensity value*
- Application of filter to each pixel (r, c)

$$g(r,c) = (\\ f(r-1,c-1) \quad +f(r-1,c) \quad +f(r-1,c+1)+ \\ f(r,c-1) \quad +f(r,c) \quad +f(r,c+1)+ \\ f(r+1,c-1) \quad +f(r+1,c) \quad +f(r+1,c+1) \\ )/9$$

  - *Applying filter to (2.b) is shown in (4.a)*

	0	1	2	3	4	5	6	7	8	9
0	10	10	10	10	10	10	10	10	10	10
1	20	20	20	20	20	20	20	20	20	20
2	30	30	30	**20**	30	30	30	30	30	30
3	40	40	40	40	40	40	**10**	40	40	40
4	50	50	50	50	**90**	50	50	50	50	50
5	60	60	60	60	60	60	60	60	60	60

(2.b) Pixel array w/ noise

	0	1	2	3	4	5	6	7	8	9
0	10	10	10	10	10	10	10	10	10	10
1	20	20	18	18	18	20	20	20	20	20
2	30	30	28	28	28	26	26	26	30	30
3	40	40	38	43	43	41	36	36	40	40
4	50	50	50	54	54	51	46	46	50	50
5	60	60	60	60	60	60	60	60	60	60

(4.a) Smoothing with box filter

Spatial Filters

- Intensity values no longer uniform
  - *But quite close to original values, except where noise pixels existed*
  - *4th row show no more noise of 90, instead, all values are in the range 46 – 54*
  - **Box filter**
    - *Equal importance* to pixel & its neighbors
- **Weighted filter**
  - *Different factor for different pixels*
  - *This filter give more weight to pixel itself than its neighbors:*

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

	0	1	2	3	4	5	6	7	8	9
0	10	10	10	10	10	10	10	10	10	10
1	20	20	20	20	20	20	20	20	20	20
2	30	30	30	**20**	30	30	30	30	30	30
3	40	40	40	40	40	40	**10**	40	40	40
4	50	50	50	50	**90**	50	50	50	50	50
5	60	60	60	60	60	60	60	60	60	60

(2.b) Pixel array w/ noise

	0	1	2	3	4	5	6	7	8	9
0	10	10	10	10	10	10	10	10	10	10
1	20	20	18	18	18	20	20	20	20	20
2	30	30	28	28	28	26	26	26	30	30
3	40	40	38	43	43	41	36	36	40	40
4	50	50	50	54	54	51	46	46	50	50
5	60	60	60	60	60	60	60	60	60	60

(4.a) Smoothing with box filter

Spatial Filters

- **Weighted filter** $\begin{bmatrix} 1 & 1 & 1 \\ 1 & 8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$
  - *Appropriate to use this filter if we think the pixel almost certainly has its correct value*
  - *But still want its neighbours to influence it*
  - *After applying filter, divide by 16 to scale sum to an intensity value*
- Applying filter to (2.b) is shown in (4.b)
  - *Checking again 4^{th} row*
  - *Original value of 90 reduced to 70 only*
  - *Since greater weight given to pixel relative to its neighbours*

	0	1	2	3	4	5	6	7	8	9
0	10	10	10	10	10	10	10	10	10	10
1	20	20	20	20	20	20	20	20	20	20
2	30	30	30	**20**	30	30	30	30	30	30
3	40	40	40	40	40	40	**10**	40	40	40
4	50	50	50	50	**90**	50	50	50	50	50
5	60	60	60	60	60	60	60	60	60	60

(2.b) Pixel array w/ noise

	0	1	2	3	4	5	6	7	8	9
0	10	10	10	10	10	10	10	10	10	10
1	20	20	19	19	19	20	20	20	20	20
2	30	30	29	25	29	28	28	28	30	30
3	40	40	39	41	41	40	25	38	40	40
4	50	50	50	52	70	50	48	48	50	50
5	60	60	60	60	60	60	60	60	60	60

(4.b) Smoothing with weighted filter

Histogram Manipulation

- (6.a) shows pixel of a binary image
  - *Each pixel either black or white*
  - *A 3 x 5 white rectangle on black background*
  - *Clearly see the image*
- (6.b) the same image as (6.a)
  - *But with lots of added random noise*
  - *Very difficult to identify rectangle*
  - *Smoothing image won't help*

	0	1	2	3	4	5	6	7	8	9
0	10	10	10	10	10	10	10	10	10	10
1	10	10	10	10	10	10	10	10	10	10
2	10	10	10	**90**	**90**	**90**	**90**	**90**	10	10
3	10	10	10	**90**	**90**	**90**	**90**	**90**	10	10
4	10	10	10	**90**	**90**	**90**	**90**	**90**	10	10
5	10	10	10	10	10	10	10	10	10	10

(6.a) Binary image w/o noise

	0	1	2	3	4	5	6	7	8	9
0	19	17	37	19	26	11	46	27	37	10
1	11	24	17	30	14	43	29	22	34	46
2	31	37	38	63	72	86	65	64	27	47
3	33	38	49	73	63	66	59	76	40	10
4	47	13	44	90	86	56	63	65	18	44
5	10	34	29	14	35	31	26	42	15	25

(6.b) Binary image w/ noise added

Ave -

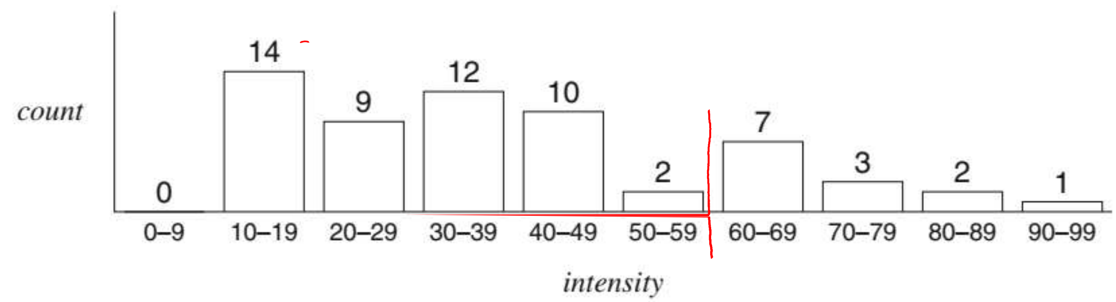
Box F -

W F -

- The values 10 → black and 90 → white are used, instead of the usual : 0 → black and 100 → white.
- This is for clarity in printing the array.

Histogram Manipulation

- (7.a) shows **histogram** of the intensities
  - **Histogram** : constructed of **bins**
  - *Each bin store count of pixels having range of intensities*
  - *Histogram in the figure has ten bins for intensities*
  - *In the ranges 0-9, 10-19, …, 91-99*
- If assume that white rectangle smaller relative to background
  - *Easy to see from histogram there are two groups of pixels*
    - **Relatively black**
    - **Relatively white**



(7.a) Histogram of noisy image

Histogram Manipulation

- If assume that white rectangle smaller relative to background
  - *Easy to see from histogram there are two groups of pixels*
    - **Relatively black**
    - **Relatively white**
- Threshold of 50 or 60
  - *Allow to distinguish between rectangle and background*
  - *Even with presence of noise*
  - *In fact, threshold of 50* → *restore original image*
  - *While threshold of 60* → *restores 13 of the 15 pixels of the triangle*

Histogram Manipulation

- **Histogram manipulation**
  - *Very efficient to compute even on large images*
  - *For each pixel :*
    divide intensity by number of bins and increment the bin number

```
for each pixel p
    bin_number ← intensity(p) / number_of_bins
    bins[bin_number] ← bins[bin_number] + 1
```

- Comparing w/ application of 3 x 3 filter
  - *9 multiplications, 8 additions, and 1 division* **per pixel**
  - *More memory required*
- 10 bins chosen so figure (7.a) can display entire histogram
  - *Full 8-bit grayscale histogram requires only 256 bins*

Histogram Manipulation

- Choosing threshold
  - *Can be easily done through examining plot of the histogram*
- Selection of the threshold can be done automatically
  - *If we know roughly the fraction of the background covered by the objects*
- Algorithms for histogram manipulation
  - *Can perform more complex enhancement*
  - *Compared to simple binary threshold described here*
- Specifically:
  - *Algorithms for enhancing images*
    - By modifying the brightness and contrast of an image

Image Processing

- Obtaining Images
- Overview of Digital Image Processing
- Image Enhancement
- **Edge Detection**
- Corner Detection
- Recognizing Blobs

Edge Detection

- Medical image processing systems
  - *Require sophisticated image enhancement algorithms*
  - *To modify brightness & contrast, removing noise, etc…*
- Medical specialists
  - *They interpret image after it has been enhanced*
  - *Know which lines/shadows correspond to which organs*
    - And if organs are normal or not
- Autonomous robot
  - *Doesn't have human to perform the interpretation*
  - *It must identify objects by itself*
    - Doors in a building, boxes in a warehouse, cars on the road
  - *First step is extract features or segments such as lines, edges, and areas*

Edge Detection

- (8.a) a 6 x 6 array of pixels
  - *Intensity across each row is uniform*
  - *But sharp discontinuity between rows 2 and 3*
  - *Clearly representing edge between*
    - Top → Dark area; Bottom → Light area
  - *Averaging makes intensity change smoother and lose sharp change at edge*

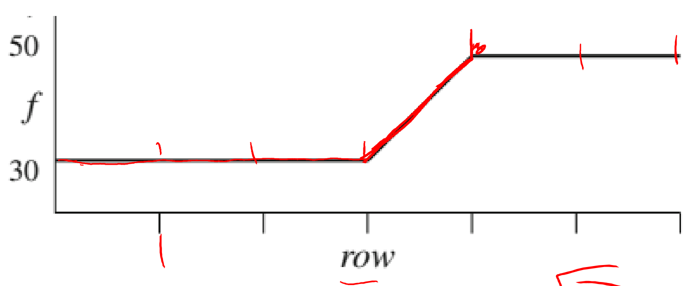
	0	1	2	3	4	5
0	30	30	30	30	30	30
1	30	30	30	30	30	30
2	30	30	30	30	30	30
3	**50**	**50**	**50**	**50**	**50**	**50**
4	**50**	**50**	**50**	**50**	**50**	**50**
5	**50**	**50**	**50**	**50**	**50**	**50**

(8.a) Image w/ an edge

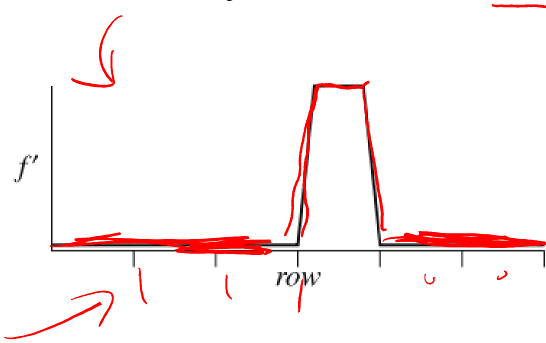
- Averaging
  - *It is an integrating operator that* **remove** *abrupt changes in intensities*
  - *Not surprising that differential operator can be used to* **detect** *abrupt changes that represent edges*

Edge Detection

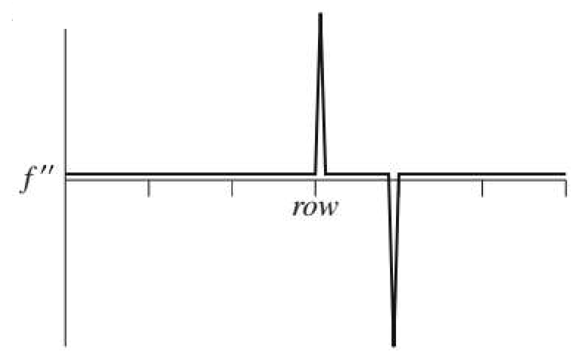
- (8.b) plot
  - *Intensity vs row number along single column of (a)*
  - *Intensities shown as lines instead of discrete pts*
  - *Constant for 1st three pixels then rapidly increase & continue at that level*
- (9.a) 1st derivative *f'* of function *f* is:
  - *"**0**" when f is constant, "**+**" when f increases, "**-**" when f decreases*



(8.b) Intensity of edge



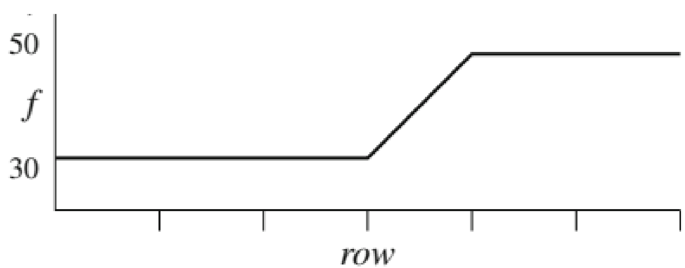
(9.a) 1st derivative of edge intensity



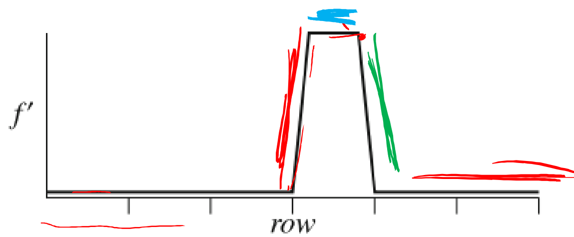
(9.b) 2nd derivative of edge intensity

Edge Detection

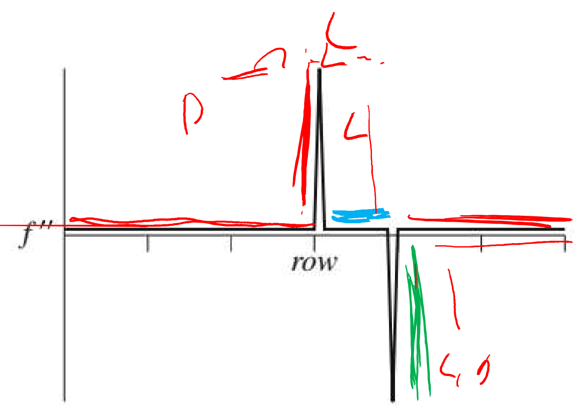
- Edge can be detected
  - *By searching for rapid increase/decrease of 1st derivative of image intensity*
  - *In practice, it is better to use 2nd derivative*
- The plot of *f''* is given in (9.b), w/c is the derivative of *f'* (9.a)
  - **Dark to light transition** : *Positive spike followed by negative spike*
  - **Light to dark transition** : *Negative spike followed by positive spike*



(8.b) Intensity of edge



(9.a) 1st derivative of edge intensity



(9.b) 2nd derivative of edge intensity

Edge Detection

- **Sobel filter**
  - *One of many digital derivative operators; Simple but effective*
  - *There are two filters*

    left : for detecting *horizontal edges*,

    right : for detecting *vertical edges*

$$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \quad \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

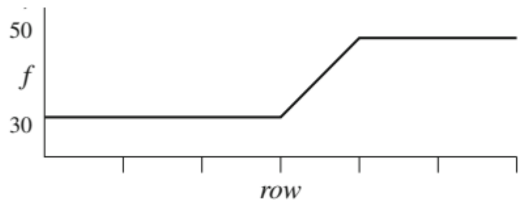
60 40 60
40 40 60
60 60 60

-60 -120 -60
0 0 0
+60 120 +60
= 0

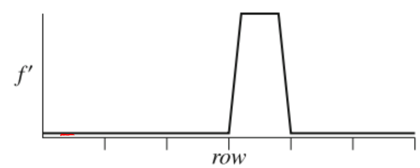
- Characteristic of a derivative filter
  - *Sum of its elements must equal to zero*
  - **Reason** : *If operator is applied to a pixel whose intensity is same as all its neighbors → result must be zero*

Edge Detection

- Observe (8.b) & (9.a)
  - *Derivative is zero when intensity is constant*



(8.b) Intensity of edge



(9.a) 1st derivative of edge intensity

- Applying Sobel filter to (8.a)
  - *We observe one horizontal edge (red rectangle)*

	0	1	2	3	4	5
0	30	30	30	30	30	30
1	30	30	30	30	30	30
2	30	30	30	30	30	30
3	**50**	**50**	**50**	**50**	**50**	**50**
4	**50**	**50**	**50**	**50**	**50**	**50**
5	**50**	**50**	**50**	**50**	**50**	**50**

(8.a) Image w/ an edge

$$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \qquad \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

+ Sobel Filter =

	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	0	0	0	0
2	0	80	80	80	80	0
3	0	80	80	80	80	0
4	0	0	0	0	0	0
5	0	0	0	0	0	0

(10.a) Sobel horizontal edge

	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	0	0	0	0	0	0
4	0	0	0	0	0	0
5	0	0	0	0	0	0

(10.b) Sobel vertical edge

Edge Detection

- Sobel filters very powerful since not only detect an edge
  - *But also compute angle of edge within the image**
  - *Applying Sobel filter to (11.a)*

	0	1	2	3	4	5
0	30	30	30	30	30	30
1	**50**	30	30	30	30	30
2	**50**	50	30	30	30	30
3	**50**	**50**	**50**	30	30	30
4	**50**	**50**	**50**	**50**	30	30
5	**50**	**50**	**50**	**50**	**50**	30

(11.a) Image w/ diagonal edge

+ Sobel Filter =

	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	**60**	20	0	0	0
2	0	**60**	**60**	20	0	0
3	0	20	**60**	**60**	20	0
4	0	0	20	**60**	**60**	0
5	0	0	0	0	0	0

(12.a) Sobel horizontal edge

	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	**−60**	−20	0	0	0
2	0	**−60**	**−60**	−20	0	0
3	0	−20	**−60**	**−60**	−20	0
4	0	0	−20	**−60**	**−60**	0
5	0	0	0	0	0	0

(12.b) Sobel vertical edge

* The angle of the edge can be computed from the given magnitudes and signs of the elements of the arrays (12.a & b) as described in:
-- Siegwart, R., Nourbakhsh, I.R., Scaramuzza, D., "*Introduction to Autonomous Mobile Robots*", 2nd Ed., MIT Press, Cambridge (2011)

Image Processing

- Obtaining Images
- Overview of Digital Image Processing
- Image Enhancement
- Edge Detection
- Corner Detection
- Recognizing Blobs

Corner Detection

- Black rectangle in gray background (14.a)
  - *Is more than just set of edges*
  - *Vertical edges form 2 corners with horizontal edge*
- Two algo for identifying corners in image
  - *For simplicity, assume corners are aligned with rectangular image*
  - *We now know how to detect edges*
- What is a corner?
  - *Defined by intersection of a vertical edge and a horizontal edge*
- (14.b) is pixel array of the image in (14.a)
  - *6 x 10 pixel array*

(14.a) Image of a corner

	0	1	2	3	4	5	6	7	8	9
0	30	30	30	30	30	30	30	30	30	30
1	30	30	30	30	30	30	30	30	30	30
2	30	30	30	**50**	**50**	**50**	**50**	**50**	30	30
3	30	30	30	**50**	**50**	**50**	**50**	**50**	30	30
4	30	30	30	**50**	**50**	**50**	**50**	**50**	30	30
5	30	30	30	**50**	**50**	**50**	**50**	**50**	30	30

(14.b) Pixel image of a corner

Corner Detection

- Applying Sobel filter to (14.a)
  - *Obtain two vertical edges*
  - *And one horizontal edge*

	0	1	2	3	4	5	6	7	8	9
0	30	30	30	30	30	30	30	30	30	30
1	30	30	30	30	30	30	30	30	30	30
2	30	30	30	**50**	**50**	**50**	**50**	**50**	30	30
3	30	30	30	**50**	**50**	**50**	**50**	**50**	30	30
4	30	30	30	**50**	**50**	**50**	**50**	**50**	30	30
5	30	30	30	**50**	**50**	**50**	**50**	**50**	30	30

(14.a) Image of a corner

Sobel Filter

(2,3)

(2,7)

	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	0	**20**	**20**	0	0	0	**−20**	**−20**	0
2	0	0	**60**	**60**	0	0	0	**−60**	**−60**	0
3	0	0	**80**	**80**	0	0	0	**−80**	**−80**	0
4	0	0	**80**	**80**	0	0	0	**−80**	**−80**	0
5	0	0	0	0	0	0	0	0	0	0

(15.a) Sobel ~~horizontal~~ edge

vertical

	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	0	**20**	**60**	**80**	**80**	**80**	**60**	**20**	0
2	0	0	**20**	**60**	**80**	**80**	**80**	**60**	**20**	0
3	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0

(15.b) Sobel ~~vertical~~ edge

horizontal

Corner Detection

- Intersection is defined as
  - *Sum of absolute values in the two Sobel edge arrays is over a threshold*

$$(1,2)=40,(1,3)=80,\ldots,(1,7)=80,(1,8)=40$$

$$(2,2)=80,(2,3)=120,\ldots,(2,7)=120,(2,8)=80$$

  - *With threshold of 30, edges intersect in the pixels (2, 3) and (2,7) which are the corners*

	0	1	2	3	4	5	6	7	8	9
0	30	30	30	30	30	30	30	30	30	30
1	30	30	30	30	30	30	30	30	30	30
2	30	30	30	**50**	**50**	**50**	**50**	**50**	30	30
3	30	30	30	**50**	**50**	**50**	**50**	**50**	30	30
4	30	30	30	**50**	**50**	**50**	**50**	**50**	30	30
5	30	30	30	**50**	**50**	**50**	**50**	**50**	30	30

(14.b) Pixel image of a corner

	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	0	**20**	**20**	0	0	0	**−20**	**−20**	0
2	0	0	**60**	**60**	0	0	0	**−60**	**−60**	0
3	0	0	**80**	**80**	0	0	0	**−80**	**−80**	0
4	0	0	**80**	**80**	0	0	0	**−80**	**−80**	0
5	0	0	0	0	0	0	0	0	0	0

(15.a) Sobel horizontal edge

	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	0	**20**	**60**	**80**	**80**	**80**	**60**	**20**	0
2	0	0	**20**	**60**	**80**	**80**	**80**	**60**	**20**	0
3	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0

(15.b) Sobel vertical edge

Corner Detection

- Uniform area, an edge and a corner
  - *Distinguished by analyzing neighbours of pixel*
  - *In a* **uniform area**
    - *All neighbours* approximately *same* intensity
  - *At an* **edge**
    - Intensities of *neighbours* are *very different* in *one* direction but *similar* in the *other* direction
  - *At a* **corner**
    - Intensities of *neighbours* show *little* similarity
- (16.a) to detect corner of (14.b)
  - *Count num of similar neighbours for each pixel → find ones w/ min value*
  - *As expected, corner pixels (2, 3) and (2, 7) have min similar neighbours*

	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	8	7	6	5	5	5	6	7	0
2	0	8	6	**3**	5	5	5	**3**	6	0
3	0	8	5	5	8	8	8	5	5	0
4	0	8	5	5	8	8	8	5	5	0
5	0	0	0	0	0	0	0	0	0	0

(16.a) Similar neighbours

	0	1	2	3	4	5	6	7	8	9
0	30	30	30	30	30	30	30	30	30	30
1	30	30	30	30	30	30	30	30	30	30
2	30	30	30	**50**	**50**	**50**	**50**	**50**	30	30
3	30	30	30	**50**	**50**	**50**	**50**	**50**	30	30
4	30	30	30	**50**	**50**	**50**	**50**	**50**	30	30
5	30	30	30	**50**	**50**	**50**	**50**	**50**	30	30

- Obtaining Images
- Overview of Digital Image Processing
- Image Enhancement
- Edge Detection
- Corner Detection
- Recognizing Blobs

Recognizing Blobs

- (17.a) shows a **blob**
  - *Roughly circular area of 12 pixels*
  - *High-intensity on a low-intensity background*
  - *No well-defined boundary like rectangle*
  - *Row 4 show two artifacts of high-intensity, not part of the blob, they may represent distinct blobs*
  - *(17.b) pixels after adding random noise*
- Task is to identify blob
  - *Without depending on predefined intensity threshold & ignoring the artifacts*
  - *Independence of the identification from overall intensity is important*
    - So robot can *fulfill task* regardless of *environmental* lighting *conditions*

	0	1	2	3	4	5	6	7	8	9
0	30	30	30	30	30	30	30	30	30	30
1	30	30	30	30	**80**	**80**	30	30	30	30
2	30	30	30	**80**	**80**	**80**	**80**	30	30	30
3	30	30	30	**80**	**80**	**80**	**80**	30	30	30
4	**80**	30	30	30	**80**	**80**	30	30	30	**80**
5	30	30	30	30	30	30	30	30	30	30

(17.a) Blob

	0	1	2	3	4	5	6	7	8	9
0	46	42	40	50	46	44	40	33	30	34
1	32	46	46	46	**67**	**73**	39	47	39	30
2	33	40	40	**73**	**68**	**63**	**73**	44	42	31
3	35	41	50	**67**	**60**	**71**	**60**	37	30	49
4	**68**	46	32	44	**61**	**77**	48	42	45	**62**
5	39	37	38	34	33	40	35	37	34	32

(17.b) Blob w/ noise

Recognizing Blobs

- To ignore noise w/o predefining threshold
  - *Define threshold in terms of average intensity of image*
- To separate blobs from one another
  - *Find pixel whose intensity above threshold →* **grow** *blob by adding neighbouring pixels whose intensity above threshold*
- For (17.b)
  - *Average intensity is 54*
  - *Blob relatively small part of background → set threshold higher than average (60)*
  - *(18.b) show image after assigning 0 to all pixels below threshold*
    - Blob detected but so are the two artifacts

	0	1	2	3	4	5	6	7	8	9
0	46	42	40	50	46	44	40	33	30	34
1	32	46	46	46	**67**	**73**	39	47	39	30
2	33	40	40	**73**	**68**	**63**	**73**	44	42	31
3	35	41	50	**67**	**60**	**71**	**60**	37	30	49
4	**68**	46	32	44	**61**	**77**	48	42	45	**62**
5	39	37	38	34	33	40	35	37	34	32

(17.b) Blob w/ noise

	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	67	73	0	0	0	0
2	0	0	0	73	68	63	73	0	0	0
3	0	0	0	67	60	71	60	0	0	0
4	68	0	0	0	61	77	0	0	0	62
5	0	0	0	0	0	0	0	0	0	0

(18.a) Blob after threshold

Recognizing Blobs

r_{-1}, c_{-1} r_{-1}, c r_{-1}, c_{+1}
r, c_{-1} ($P_{r,c}$) r, c_{+1}
r_{+1}, c_{-1} r_{+1}, c r_{+1}, c_{+1}

- Isolating a single blob

```
integer threshold          # 60
pixel p
set not-explored ← empty-set
set blob ← empty-set
```

```
set threshold to the average intensity
set pixels below threshold to zero
find non-zero pixel & add to not-explored
while not-explored not empty
      p ← some element of not-explored
      add p to blob
      remove p from not-explored
      add non-zero neighbours of p to
       not-explored
```

(Algo 12.1) Detecting a blob

	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	67	73	0	0	0	0
2	0	0	0	73	68	63	73	0	0	0
3	0	0	0	67	60	71	60	0	0	0
4	68	0	0	0	61	77	0	0	0	62
5	0	0	0	0	0	0	0	0	0	0

(18.a) Blob after threshold

- **1st**, search pixel that is non-zero
  - Starting top-left, pixel $p_1(1,4) = \mathbf{67}$
- **2nd**, grow blob
  - Add all neighbours of p_1 whose intensities are non-zeroes
  - $p_1(1,5)$, $p_3(2,3)$, $p_4(2,4)$, $p_5(2,5)$

Recognizing Blobs

- Isolating a single blob

```
integer threshold
pixel p
set not-explored ← empty-set
set blob ← empty-set
```

```
set threshold to the average intensity
set pixels below threshold to zero
find non-zero pixel & add to not-explored
while not-explored not empty
      p ← some element of not-explored
      add p to blob
      remove p from not-explored
      add non-zero neighbours of p to
       not-explored
```

(Algo 12.1) Detecting a blob

	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	67	73	0	0	0	0
2	0	0	0	73	68	63	73	0	0	0
3	0	0	0	67	60	71	60	0	0	0
4	68	0	0	0	61	77	0	0	0	62
5	0	0	0	0	0	0	0	0	0	0

(18.a) Blob after threshold

- **3rd**, continue adding
  - Non-zero neighbours of each p_i
  - Until no more pixels are added
- **Result**
  - Will be 12-pixel blob
  - Without the artifacts at (4, 0) & (4, 9)

Recognizing Blobs

- The algorithm worked for (18.a)
  - *Since the first non-zero pixel found (1,4) belonged to the blob*
- If isolated non-zero found at (1,1)
  - *This artifact will be detected as a blob*
- If we estimate minimum size of the blob
  - *Check that blob is at least minimum size after implementing the algorithm*
- Check algo not sensitive to intensity level
  - *Subtract constant value 20 from all elements of noisy image (17.b) in revised algo*
  - *It should still identify same pixels belonging to the blob*

s=1 blob1 blob2 s=12 blob3 s=1 blob4 s=1

	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	67	73	0	0	0	0
2	0	0	0	73	68	63	73	0	0	0
3	0	0	0	67	60	71	60	0	0	0
4	68	0	0	0	61	77	0	0	0	62
5	0	0	0	0	0	0	0	0	0	0

(18.a) Blob after threshold

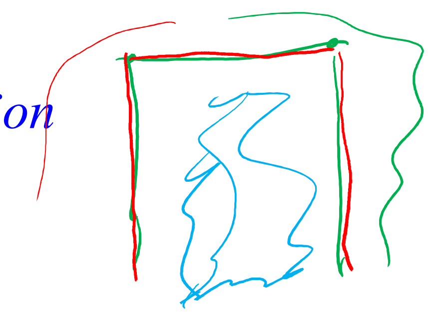
	0	1	2	3	4	5	6	7	8	9
0	46	42	40	50	46	44	40	33	30	34
1	32	46	46	46	**67**	**73**	39	47	39	30
2	33	40	40	**73**	**68**	**63**	**73**	44	42	31
3	35	41	50	**67**	**60**	**71**	**60**	37	30	49
4	**68**	46	32	44	**61**	**77**	48	42	45	**62**
5	39	37	38	34	33	40	35	37	34	32

(17.b) Blob w/ noise

+ 20

Summary

- Vision is most important sensor (human beings & most animals)
  - *Large portion of brain devoted to interpreting visual signals*
  - *Robots use vision w/in environment that is constantly changing*
  - *Algorithms for digital image processing enhance & interpret images*
- Enhancement algorithms
  - *Remove noise, improve contrast, other operations that don't depend on what objects appear in an image.*
  - *We studied spatial filters and histogram modification*



- After image enhancement
  - *Algorithms identify objects in the image*
  - *Start by detecting simple geometric properties like edges & corners → proceed to identify objects that appear in the image*

Thank you.