NANYANG TECHNOLOGICAL UNIVERSITY

School of Mechanical & Aerospace Engineering

Design, Machine, Control, Intelligence

MA4829

Machine Intelligence

(AI 1.0 <><> AI 2.0 <><> AI 3.0)

XIE Ming, PhD (France)

http://personal.ntu.edu.sg/mmxie

"We are living inside an ocean of signals"

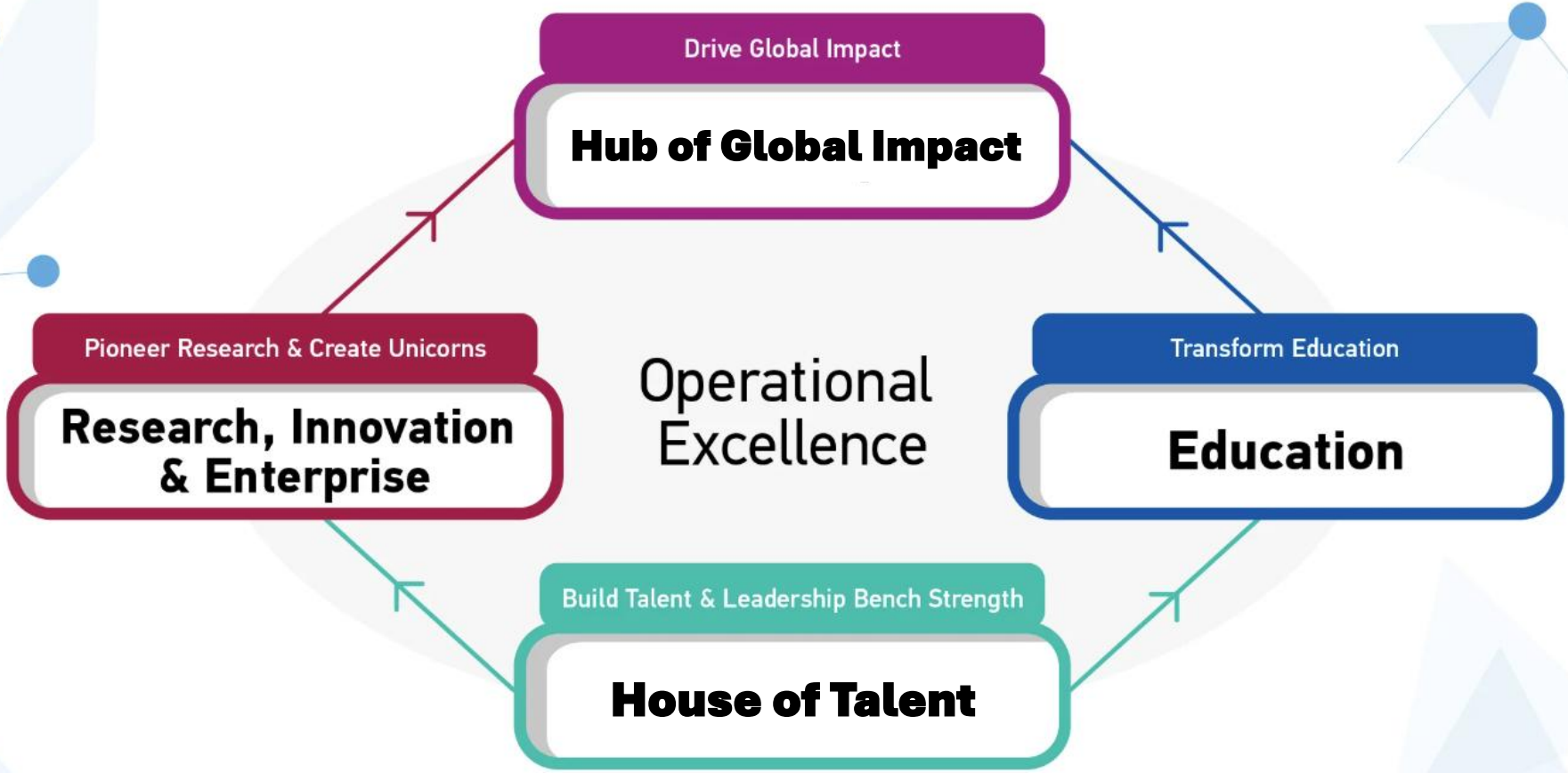
What is a system with intelligence inside?

Outline of AI 3.0

- Module 1: True Foundation of Visual Intelligence
- Module 2: Visual Knowledge Representation
- Module 3: Visual Knowledge Perception
- Module 4: Visual Knowledge Computation
- Module 5: Visual Knowledge Applications

ABOUT NTU

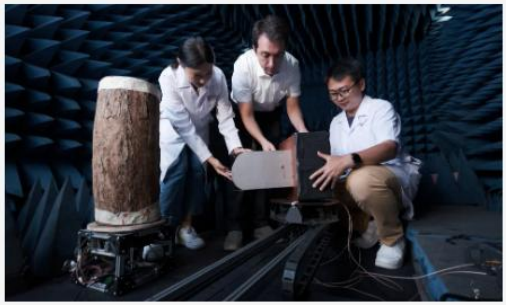
Remember NTU's Vision ...



Remember NTU's Mission ...

Education

We deliver transformative educational experiences that make our students both future- and AI-ready, so they are sought after by employers.



House of Talent

We attract, develop, and retain the very best people to drive excellence across the University.

Research, Innovation and Enterprise

We pursue breakthrough discoveries. We integrate technology and the humanities to address global challenges. We accelerate cutting edge innovation and create promising new enterprises.

Hub of Global Impact

We drive global impact in all that we do. We pursue long-lasting global partnerships with like-minded institutions across the world.

Education is to help citizens to fulfill their missions on Earth, which include: to understand the world and to improve the world …

0月18日 Friday
农历癸巳年九月十四
版／售价80￠
obao | zaobao.com
加坡报业控股

联合早报

南洋 ○ 星洲

部分时间托管残疾者
明年也可获80%津贴
3 焦点

拒载德士增加
今年210人受罚 4 新加坡

华乐团招募4000人
明年在国家体育场办音乐会
7 新加坡

李总理：排名不应是办大学终极考量

黄伟曼 报道
ngwaimun@sph.com.sg

本地大专学府的主要服务对象是新加坡学生，大学除了追求学术表现之外，也扮演重要的国家与社会角色，李显龙总理提醒本地大学不应为追求名次而忽略所肩负的重任，必须持续改善本地教育，为新加坡人提供精益求精的教育。

李总理昨晚主持新加坡国立大学大学城（University Town，简称UTown）开幕仪式时指出，本地大学的世界排名一直受到关注，但在衡量大学的表现时，不应只看名次先后，排名也不应是办大学的终极考量。

他向现场近1000名国大学生、教师与嘉宾解释说，许多在世界大专教育排行榜上名列前茅的大国顶尖大学收生广泛，例如美国哈佛大学、斯坦福大学、英国牛津大学、剑桥大学、中国北京大学和清华大学等每年录取的学生中，只有一小部分是当地学生，而本地的情况却不同，大专学府主要收本地生，必须履行更广泛的国家任务。

他说："本地大学的责任在于发展学生的品格与社会良知、鼓励他们在求学时期建立长久的友谊与默契。大学要以灌输新加坡价值观和精神特质为出发点，教导学生取之社会用之社会的精神，让他们知道自己有责任推动新加坡前进。"

"要达到这个宗旨，每一所大学都应该建立自己独特的教育模式，不要一味复制其他大学。"

占地19公顷的大学城是国大搬到肯特岗校园现址后，最大型的一次扩展计划。设于大学城的四所寄宿型学院和为研究生所设的大学城宿舍，将为4000多名学生提供住宿，将教学融入住宿环境里，让来自不同背景和科系的学生一起生活、一起学习。

在大学城学院课程中，学生需在所住的学院里修读五个单元，包括两个写作单元和三个研讨课单元。他们也有机会聆听外国学者发表专题演讲，并参加各种不同的课外活动。

李总理昨天肯定国大为提升学生学习体验所做的努力，他认为大学城为学生营造了一个有利于相互学习的空间。他指出，大学城是国大的"新聚集点"，它将国大社群缩小，进一步拉近学生与同学和教授的关系。

他也以陈爱丽丝与彼德学院（College of Alice & Peter Tan)学生之前在"整合式课程"（Capstone Programme）下为乐龄人士制定健康饮食计划为例，表示希望看到大学城继续推行类似社会计划，鼓励学生积极参与这些关怀社会的活动。

国大校长陈祝全教授致辞时指出，除了设施与空间设计吸引人之外，大学城为国大社群以及国大的运作和成长带来了深刻且积极的影响。"大学城可说已成为国大校园的核心，在一些方面补足了之前肯特岗校园所缺少的生气。"

相关新闻刊第6页

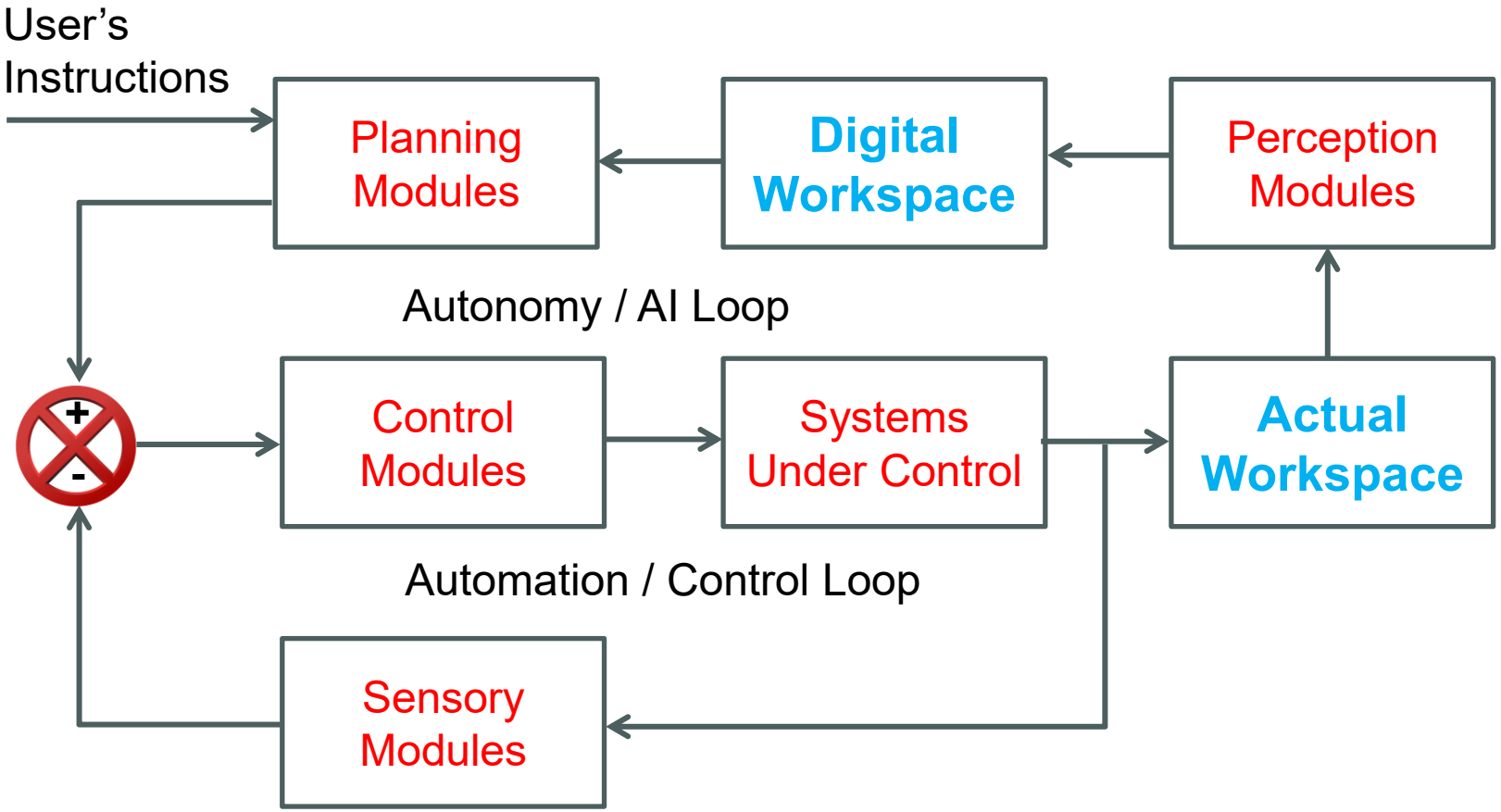
ABOUT YOU

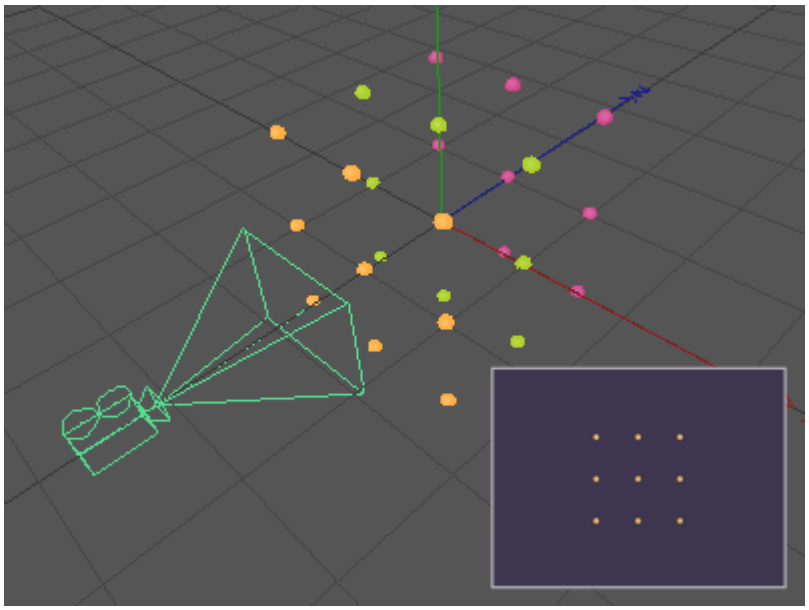
Remember your mission as MAE undergraduates …

- You are here to grow your knowledge and skills so as to be able to design machines with controllable behaviors and hopefully in some intelligent ways.

How to fulfill your mission?

- To apply learnt knowledge and skills into the implementation of the following universal blueprint underlying all the intelligent machines or systems.





ABOUT COURSE

Two Big Questions …

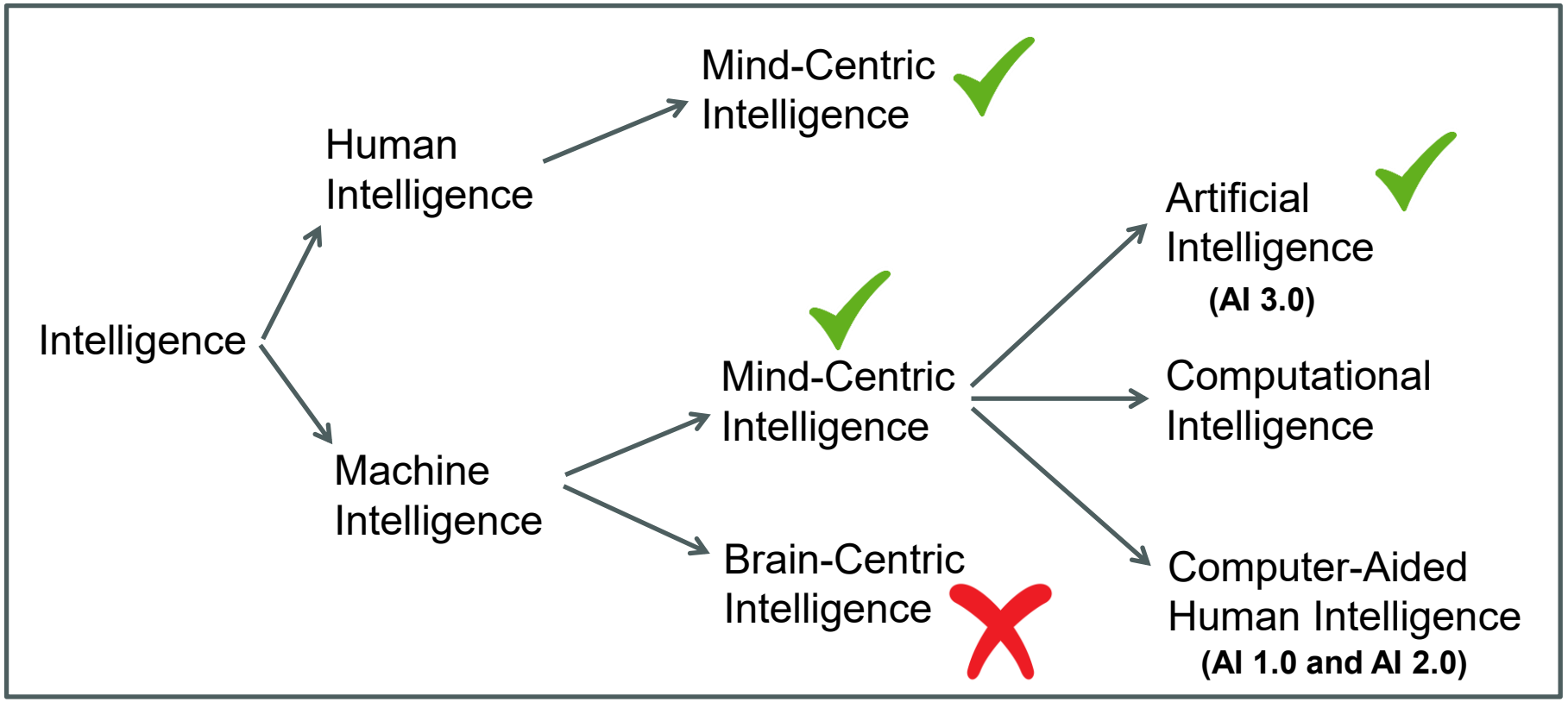
- How to achieve automation?
  - To apply the theory of control
- How to achieve autonomy?
  - To apply the theory of mind

Two Big Dilemmas …

- Top-Down Design versus Bottom-Up Optimization
  - Should the creatures in the universe be the outcomes of designs by Designers or the outcomes of natural evolution by Optimizers?

- Mind-Centric Approach versus Brain-Centric Approach
  - Does intelligence arise from Mind or from Brain?

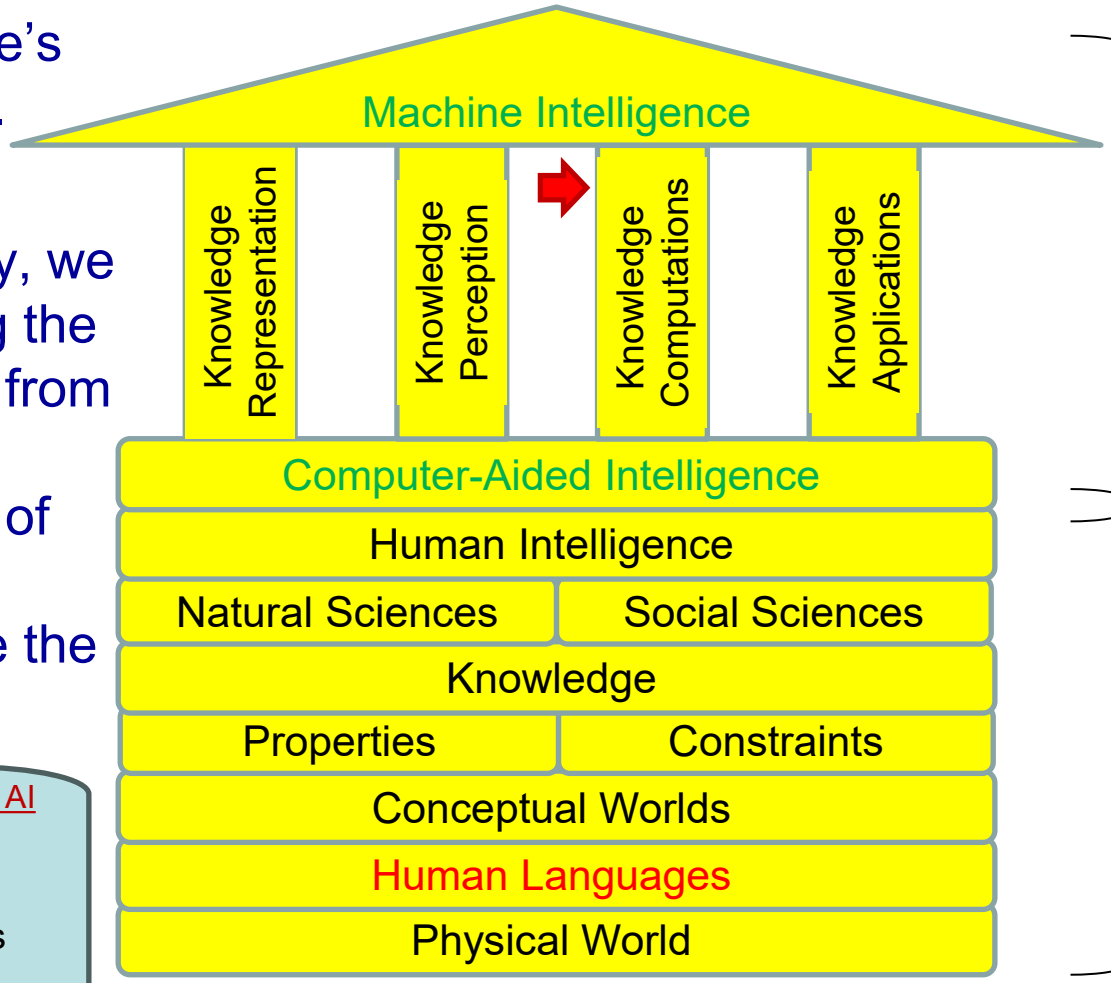
Dilemma versus Common Sense ...

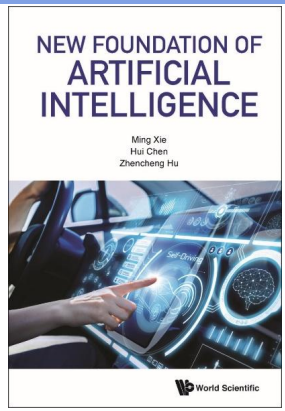


Why to study visual intelligence?

- The motivation is to achieve machine's self-intelligence.
- More specifically, we aim at achieving the transformations from visual signals to cognitive states of knowing the meanings inside the visual signals.

"We are living inside an ocean of signals"





Artificial Intelligence

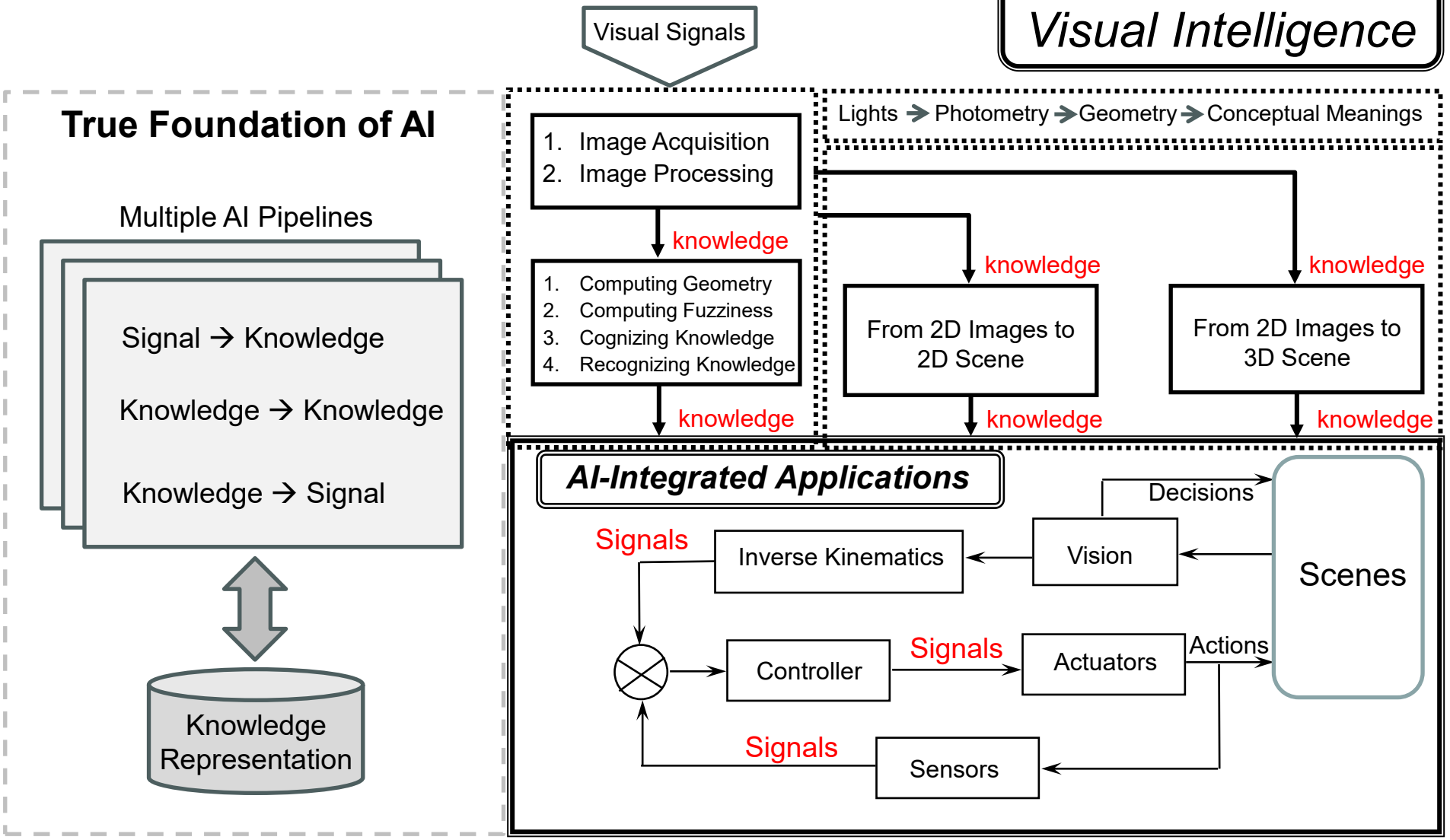
Human Intelligence

Keys of AI

- One Tool
- Two Worlds
- Three Intelligences
- Four Pillars

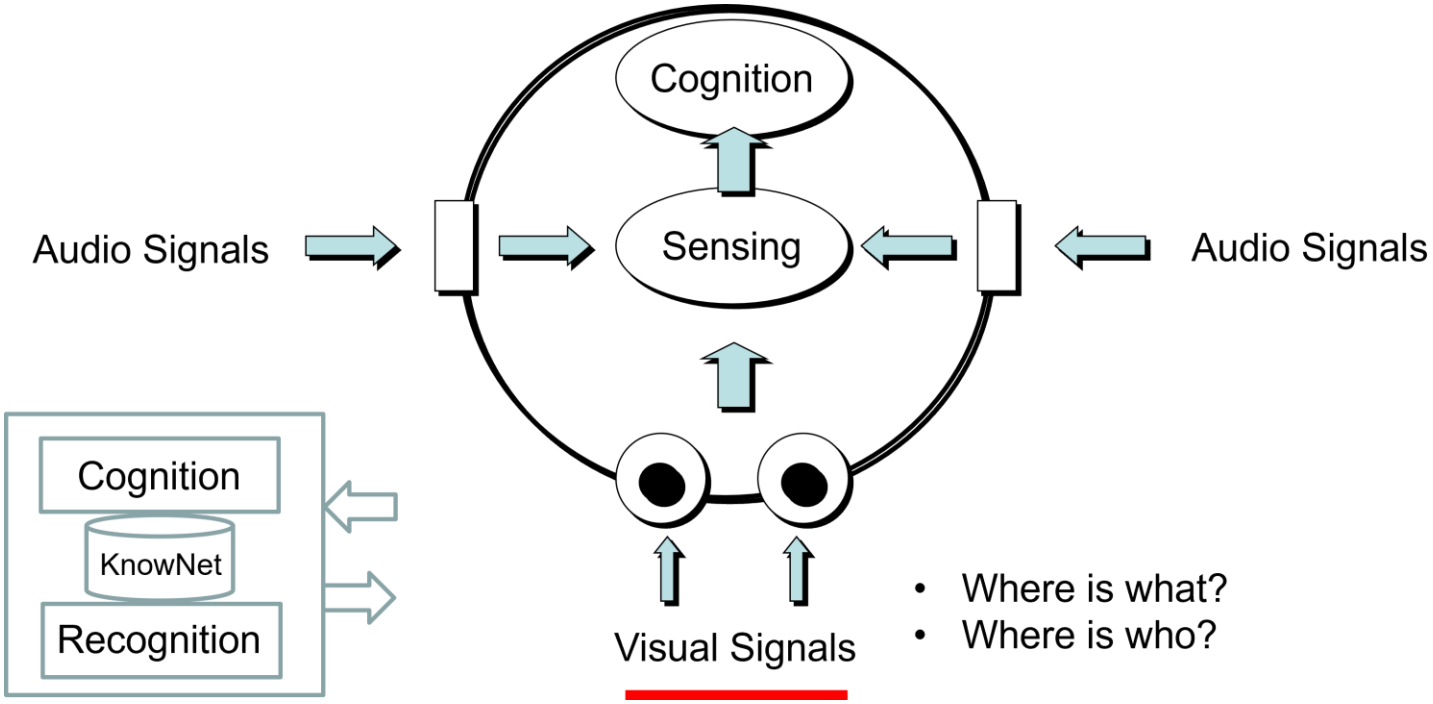
What to learn?

Roadmap of Visual Intelligence



How to study visual intelligence?

- To put yourselves into the mindset of designers of intelligent systems:
  - Who are the users? (any machine with intelligence inside)
  - What are the needs of users? (to transform visual signals into knowledge)
  - What are your intelligent systems? (to spell out the functionality)
  - What are the solutions behind the design of your intelligent systems?



How to apply?

- You could use MATLAB to quickly apply:
  - What you have read
  - What you have heard
  - What you have learnt

Practice
Practice
Practice

MATLAB = No.1 AI Software

(Rapid Prototyping Software)

Should Seeing be Believing?

We are more affected by Beliefs than by Truths.

Should Seeing Be Believing?

Terminology Alert …

1. Image properties at the lowest level include continuities and discontinuities. Hence, there are two types of pixels which are color pixels and edge pixels. These are the identities of pixels at the lowest level.
2. Therefore, "how to transform pixels into their identities?" is the so-called problem of symbol grounding in visual intelligence.
3. As a result, symbol groundings include symbol grounding of color pixels and symbol grounding of edge pixels.
4. Symbol grounding of color pixels is a process of sense-making which transforms pixels into common-sense or inner-sense of colors.
5. Symbol grounding of edge pixels is also a process of sense-making which transforms pixels into common-sense or inner sense of curves.
6. The combinations of colors and curves create meaningful patterns.

Today's Lectures …

- Module 1: True Foundation of Visual Intelligence
- Module 2: Visual Knowledge Representation
- Module 3: Visual Knowledge Perception
- Module 4: Visual Knowledge Computation
- Module 5: Visual Knowledge Applications

NANYANG TECHNOLOGICAL UNIVERSITY

School of Mechanical & Aerospace Engineering

Design, Machine, Control, Intelligence

Module 4

AI 3.0

MA4829 Machine Intelligence

Visual Knowledge Computation

XIE Ming, PhD (France)

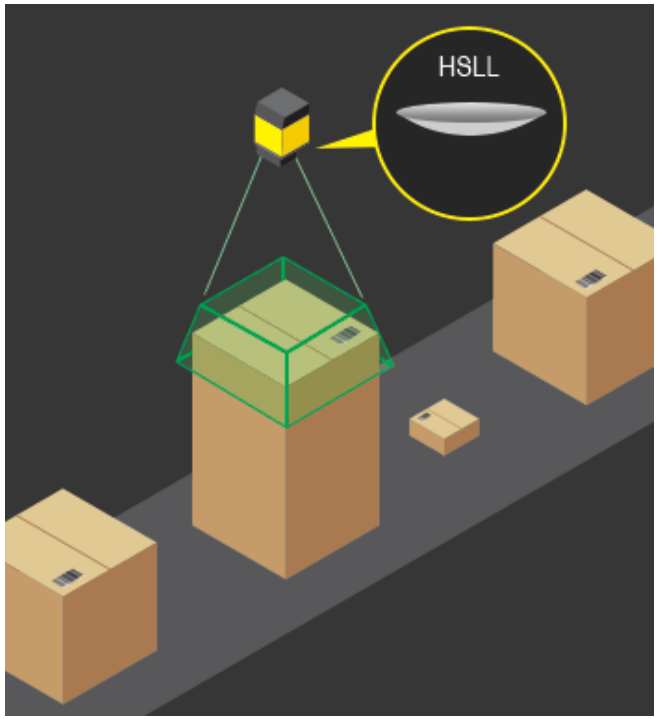
http://personal.ntu.edu.sg/mmxie

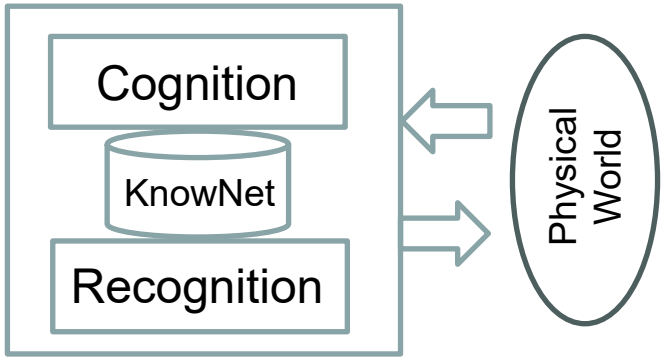
"We are living inside an ocean of signals"

What is a system with intelligence inside?

Outline of Module 4

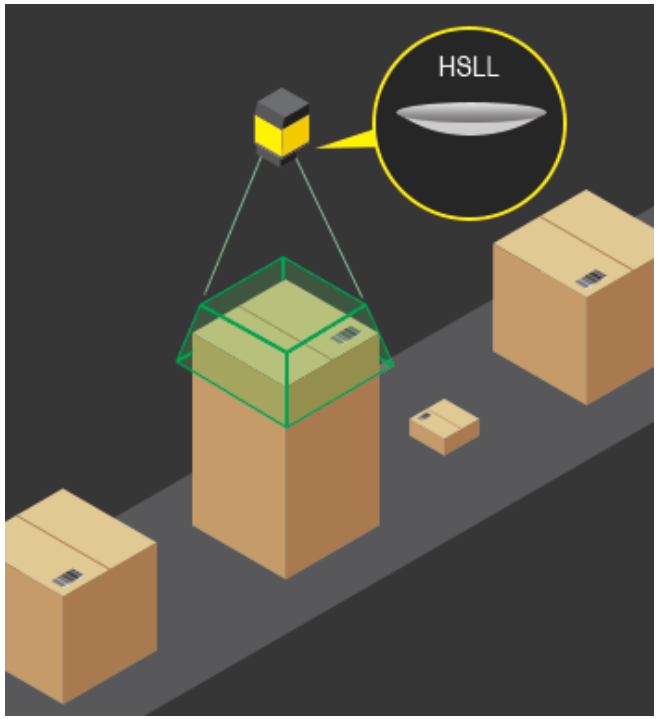
- Lecture 1:
  - Geometry-Driven Computation
- Lecture 2:
  - Fuzziness-Driven Computation
- Lecture 3:
  - Cognition-Driven Computation
- Lecture 4:
  - Recognition-Driven Computation
- Lecture 5:
  - Computation Using Monocular Vision
- Lecture 6:
  - Computation Using Binocular Vision

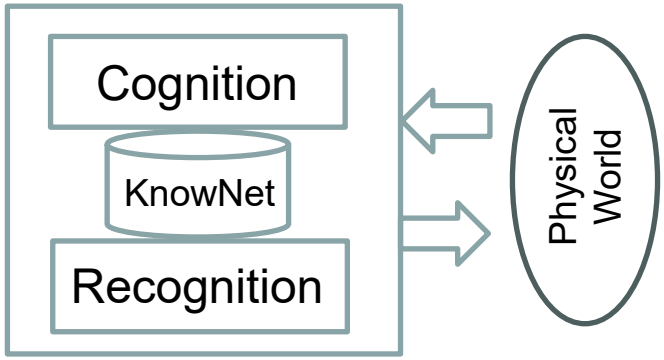




Outline of Module 4

- Lecture 1:
  - Geometry-Driven Computation
- Lecture 2:
  - Fuzziness-Driven Computation
- Lecture 3:
  - Cognition-Driven Computation
- Lecture 4:
  - Recognition-Driven Computation
- Lecture 5:
  - Computation Using Monocular Vision
- Lecture 6:
  - Computation Using Binocular Vision





NANYANG TECHNOLOGICAL UNIVERSITY

School of Mechanical & Aerospace Engineering

Design, Machine, Control, Intelligence

Lecture 1 of Module 4

AI 3.0

MA4829 Machine Intelligence

Geometry-Driven Computation

XIE Ming, PhD (France)

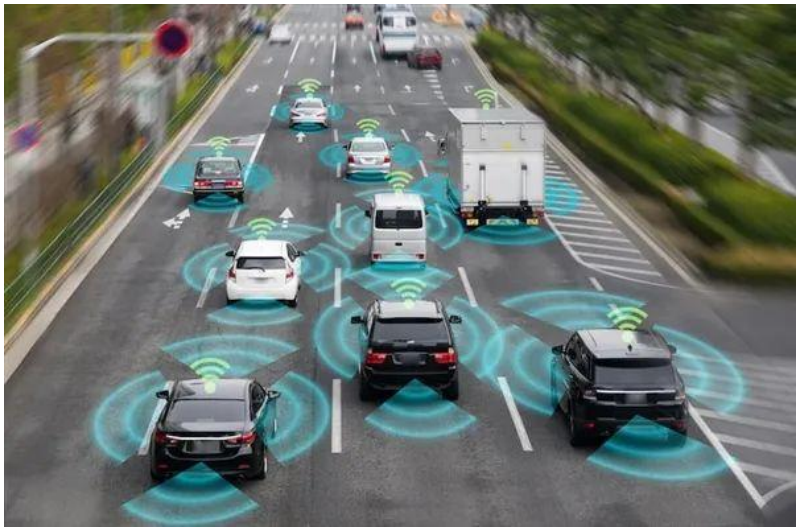
http://personal.ntu.edu.sg/mmxie

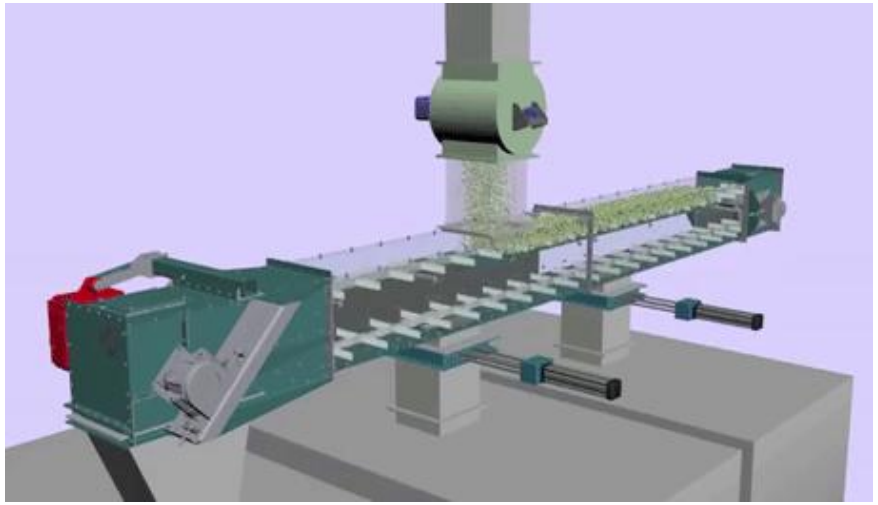
"We are living inside an ocean of signals"

What is a system with intelligence inside?

Outline of Lecture 1

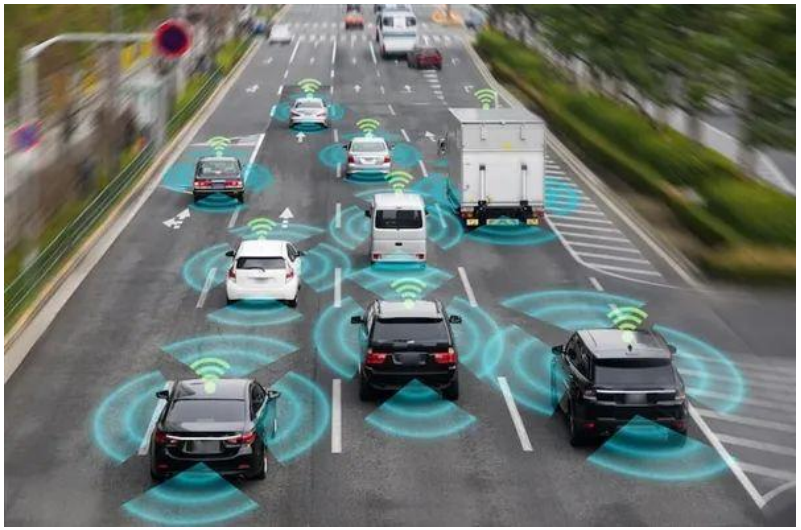
- Concept of Knowledge Space
- Representation of 2D Geometry
- Computation of 2D Geometry
- Representation of 3D Geometry
- Computation of 3D Geometry

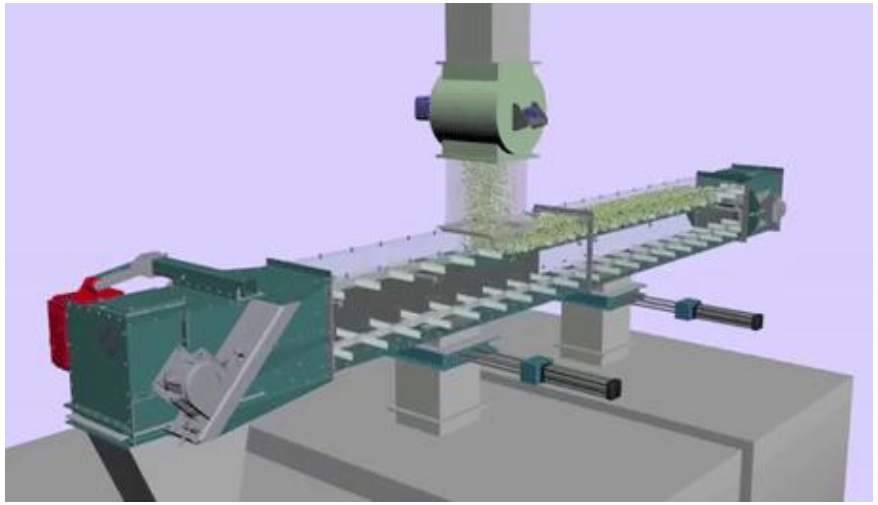




Outline of Lecture 1

- Concept of Knowledge Space
- Representation of 2D Geometry
- Computation of 2D Geometry
- Representation of 3D Geometry
- Computation of 3D Geometry

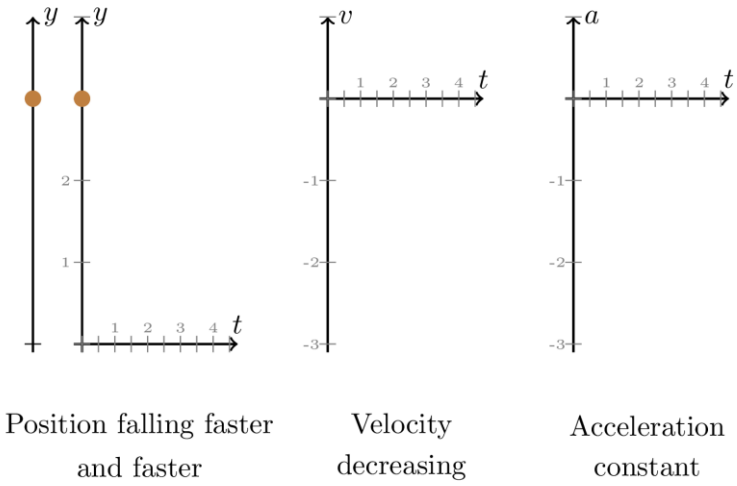




What is knowledge?

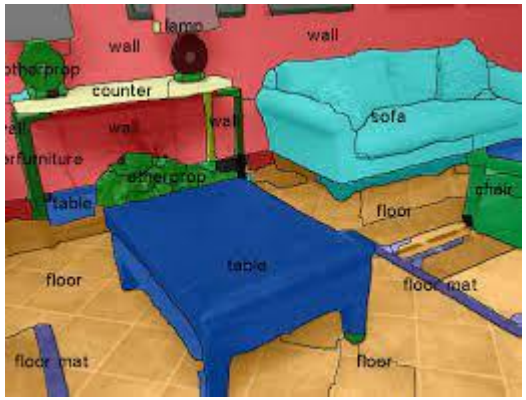
The knowledge about an entity consists of:

- Properties
- Constraints
- Behaviours



Position falling faster and faster | Velocity decreasing | Acceleration constant

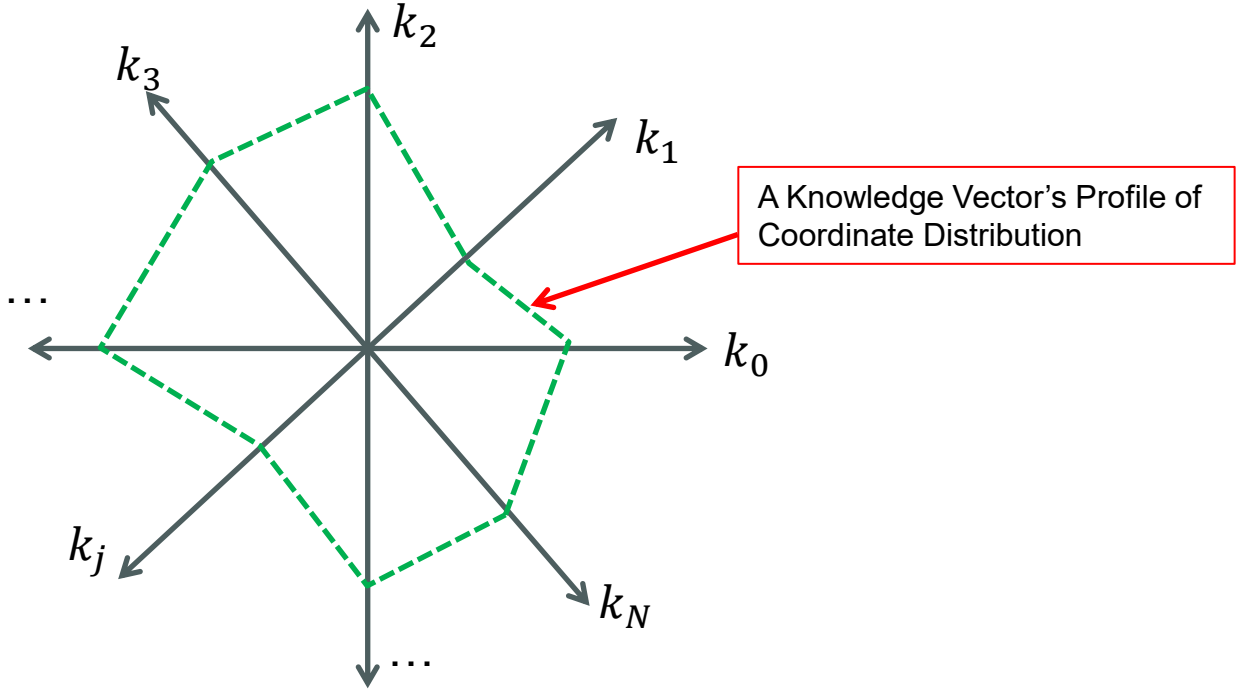
- The knowledge from images includes:
  - Photometry
  - Geometry



What is knowledge space?

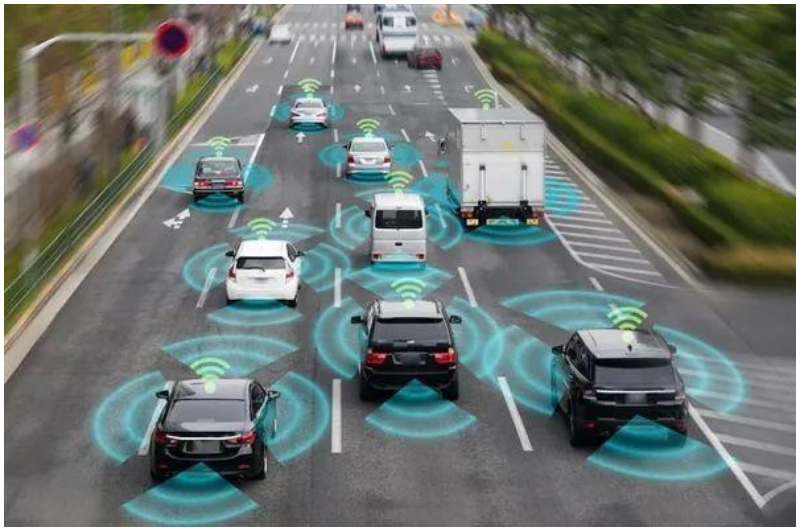
- Knowledge space refers a space of super-high dimension in which each location represents a knowledge vector such as:

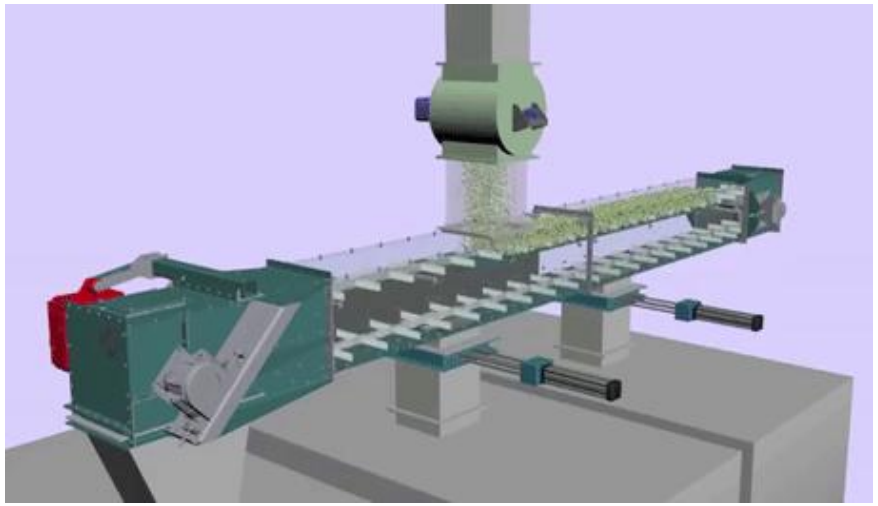
$$K_i = \{k_j, j = 0,1,2,3, \dots, N\}, i = 1,2,3, \dots, M$$



Outline of Lecture 1

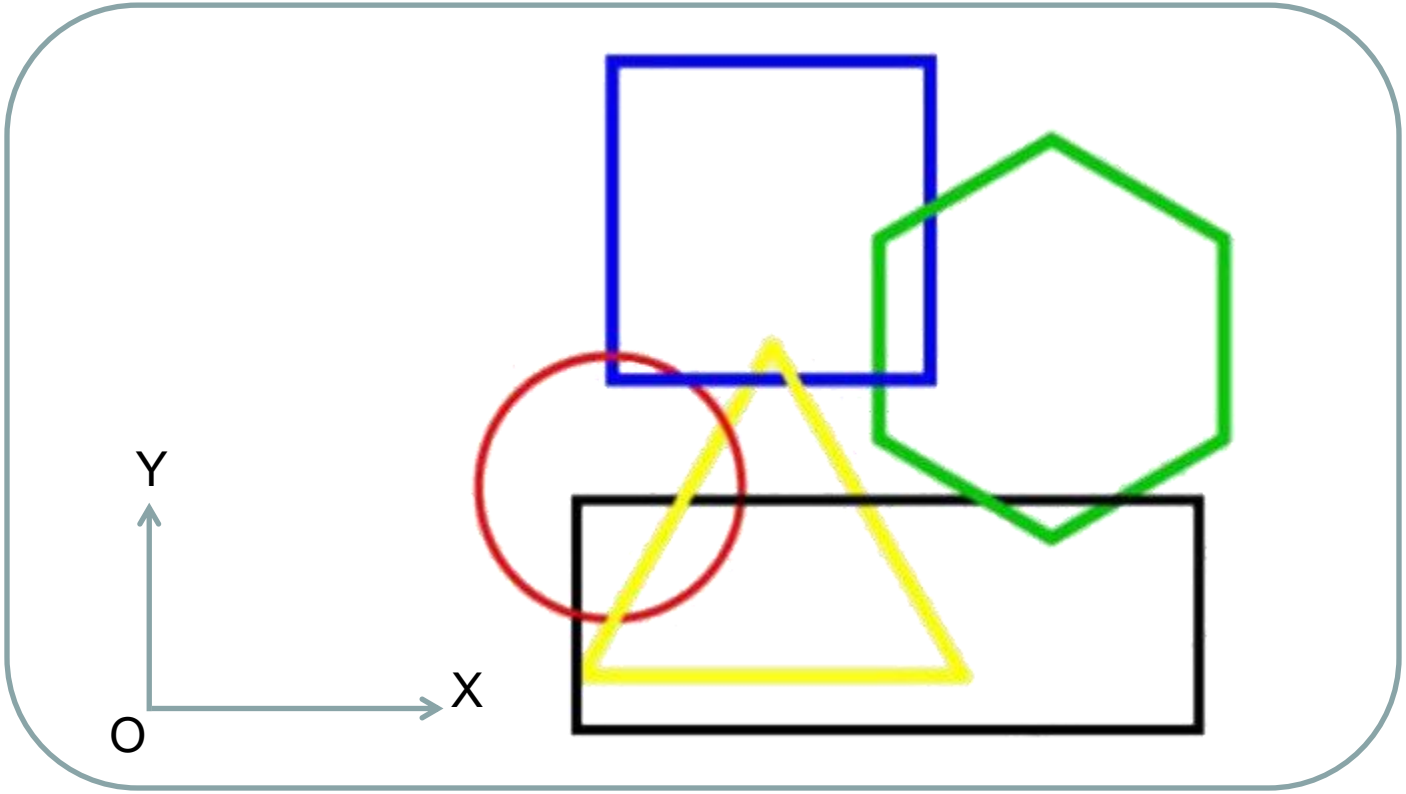
- Concept of Knowledge Space
- Representation of 2D Geometry
- Computation of 2D Geometry
- Representation of 3D Geometry
- Computation of 3D Geometry





Understanding 2D Geometry (1)

- 2D geometry refers to the appearance of physical entities in a two-dimensional space, or 2D space in short.
- 2D space consists of a set of positions which are fully determined with two coordinates.

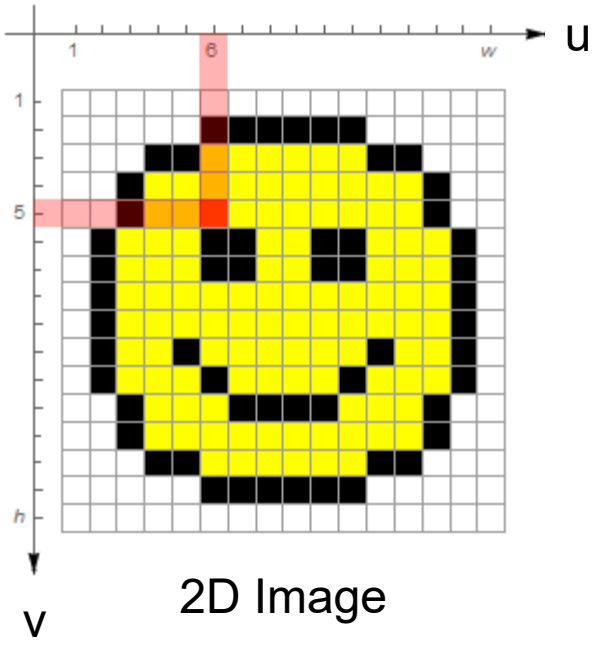


Example of Geometry in 2D Scene

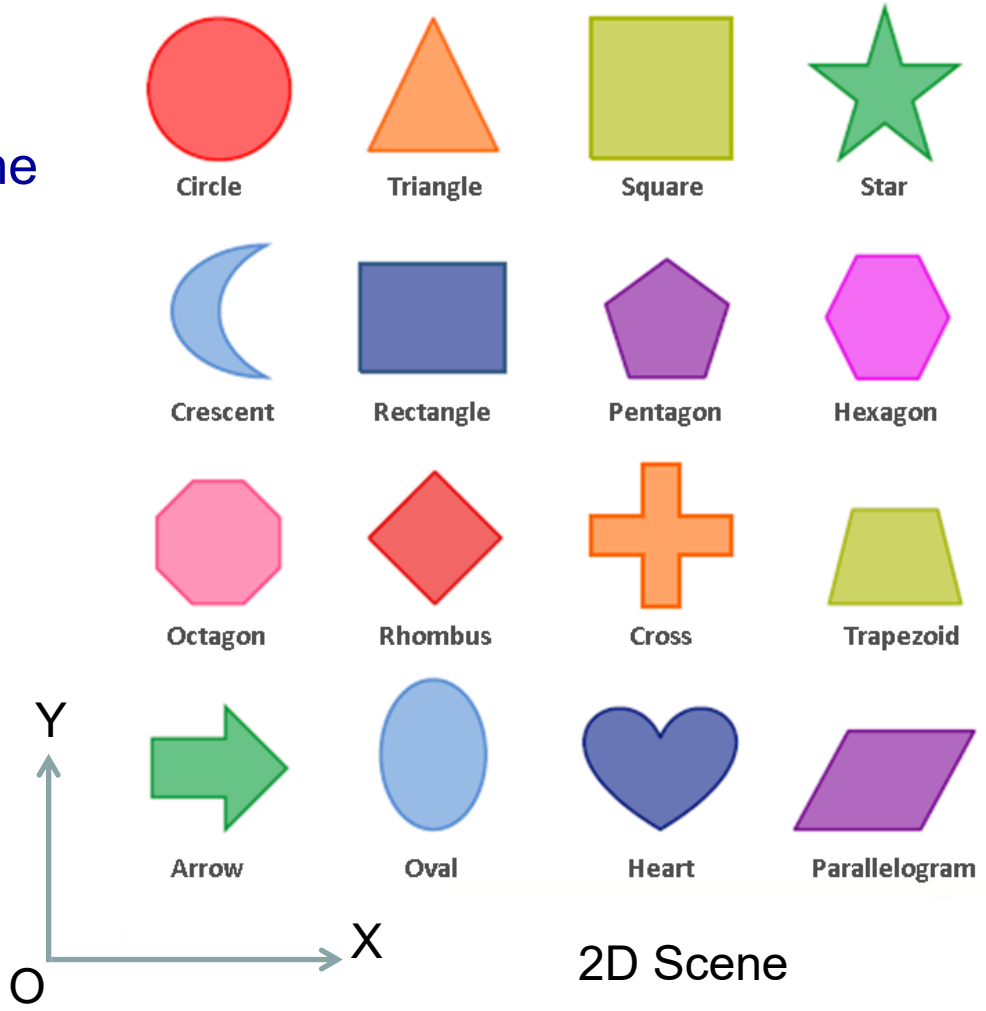
Example of Geometry in 2D Image

Understanding 2D Geometry (2)

- The appearance of physical entities in a 2D space is manifested in the form of shapes.



2D Image



2D Scene

Understanding 2D Geometry (3)

(Why to use the terminology of generative AI? What is not generative?)

- Complex shapes in a 2D space are the results of compositional rules such as:

Intersection at Points

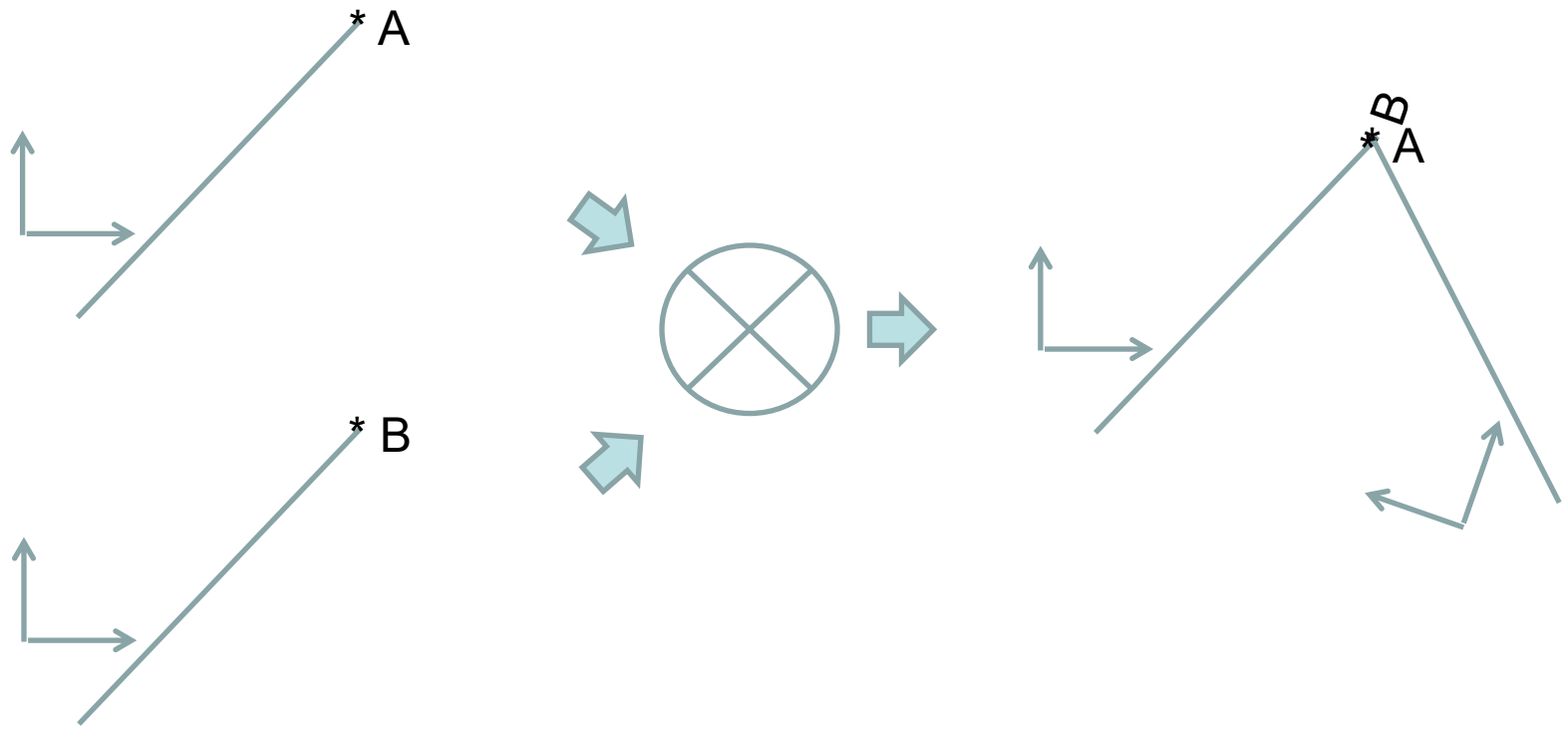
  - Connect Between Points(point of shape 1, point of shape 2) at Angle(angle between shape 1 and shape 2):
    - ConnectPointsAtAngle(point, point, angle)

Intersection at Curves

  - Connect Between Curves(curve of shape 1, curve of shape 2) with Offset(distance between the endpoints of two curves):
    - ConnectCurvesAtOffset(curve, curve, offset)

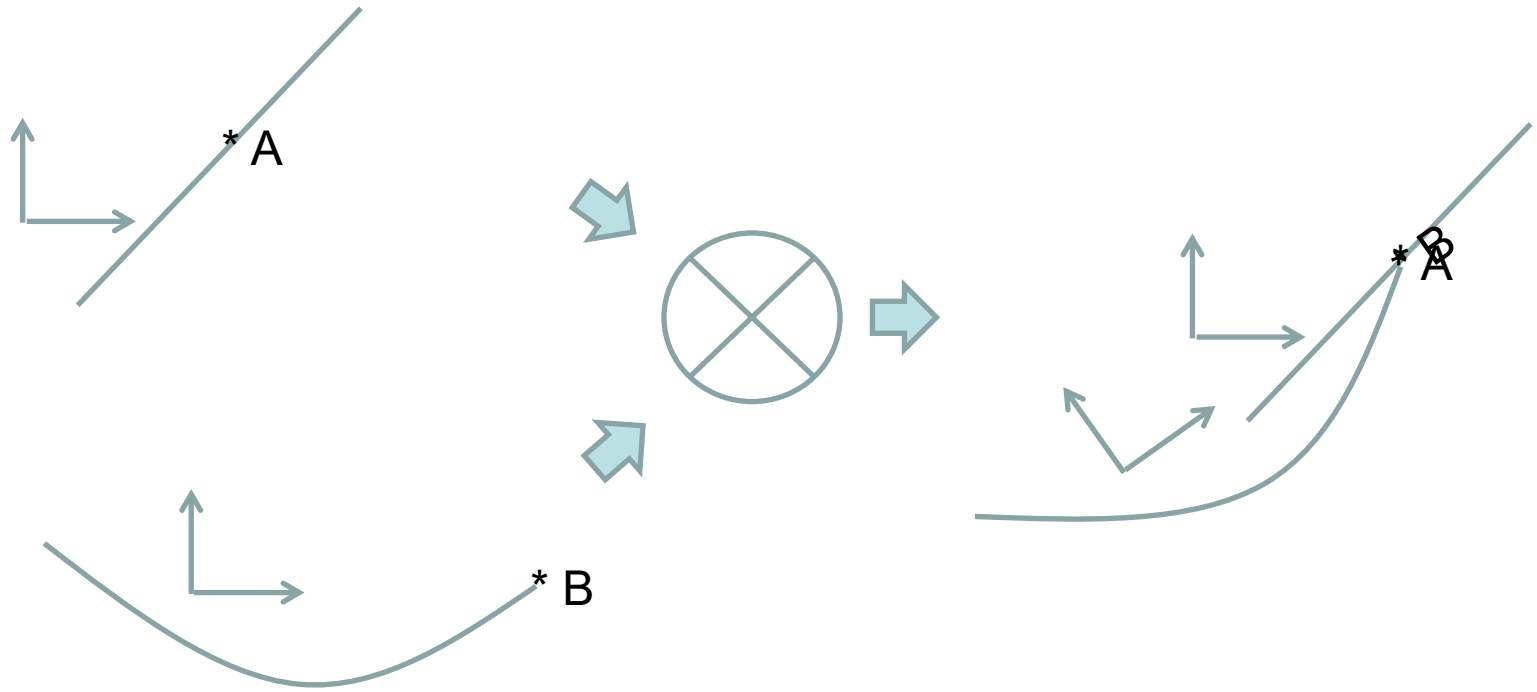
Example

- Connect Between Points(A, B) with Angle(60º)



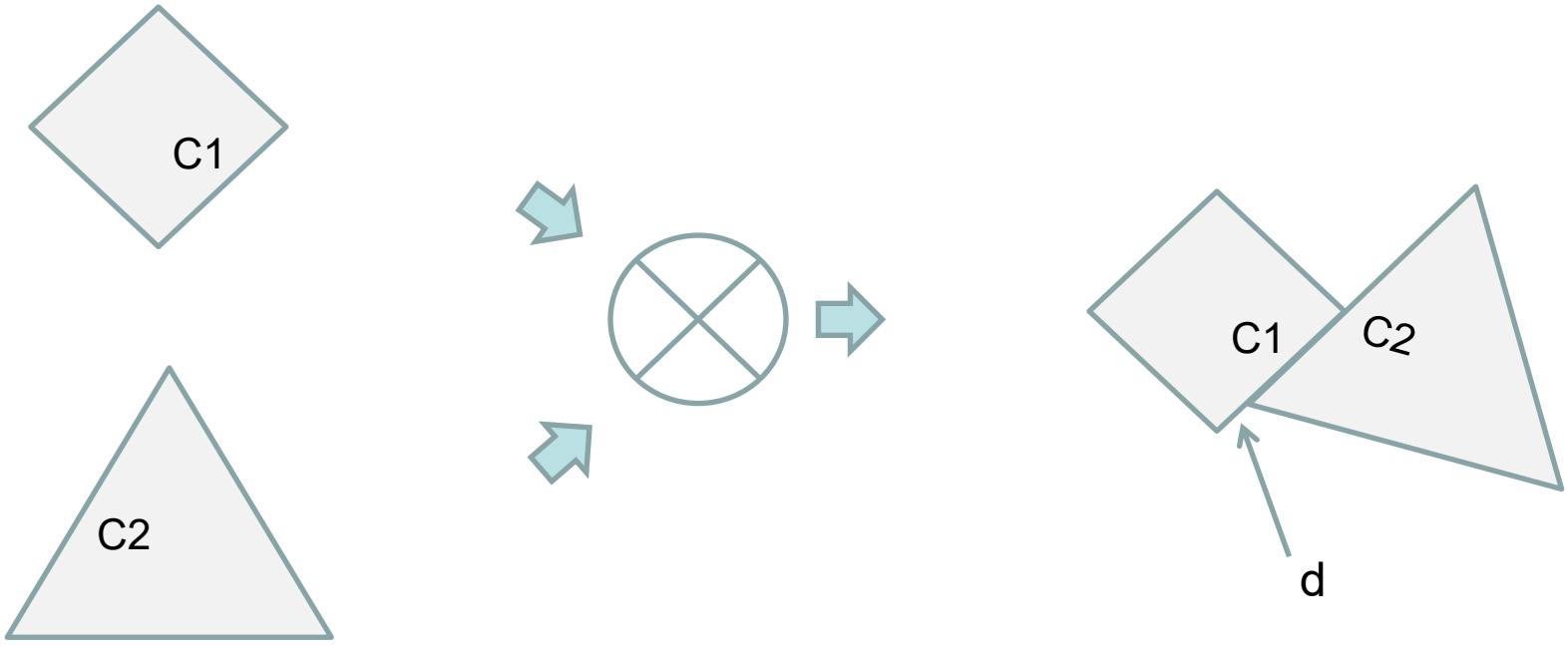
Example

- Connect Between Points(A, B) with Angle(45º)



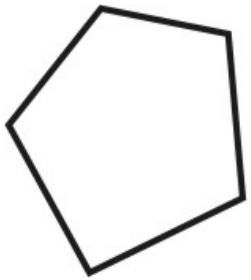
Example

- Connect Between Lines(C1, C2) with Offset(d)

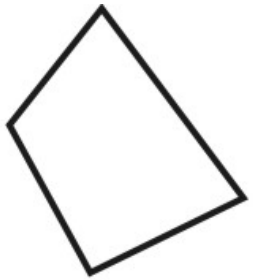


Understanding 2D Geometry (4)

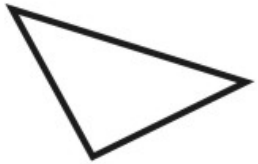
- The sum of inner angles of a 2D polygon is equal to (N-2) x 180 degrees.



N = 5
G = 540°



N = 4
G = 360°

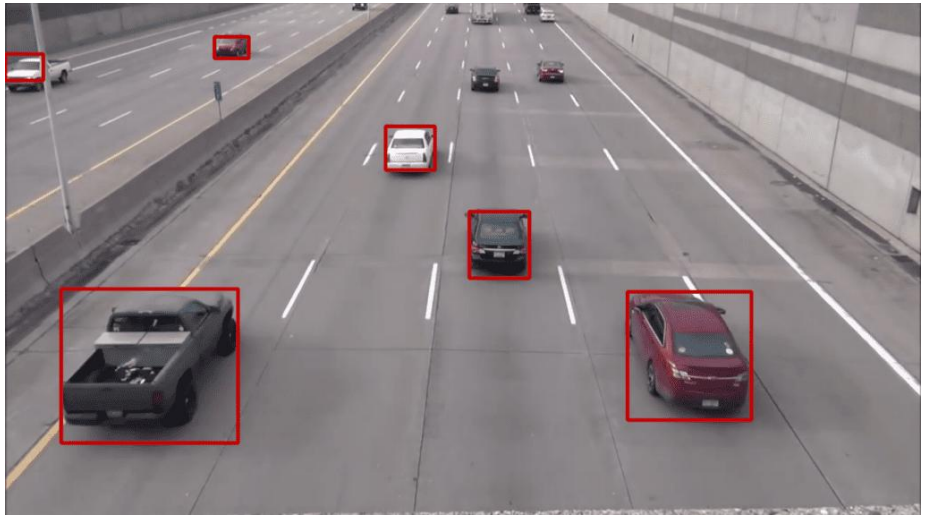


N = 3
G = 180°

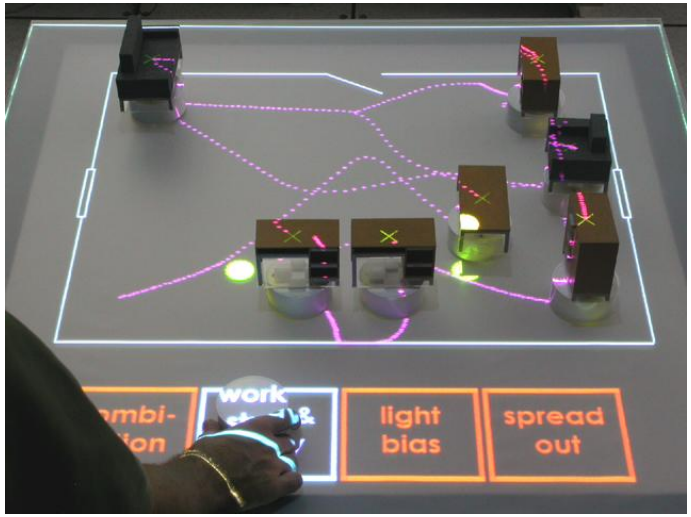
N = 2
G = 0°

Understanding 2D Geometry (5)

- The geometry of physical entities in a 2D space also includes their positions, orientations and travelled locations (i.e. motions).



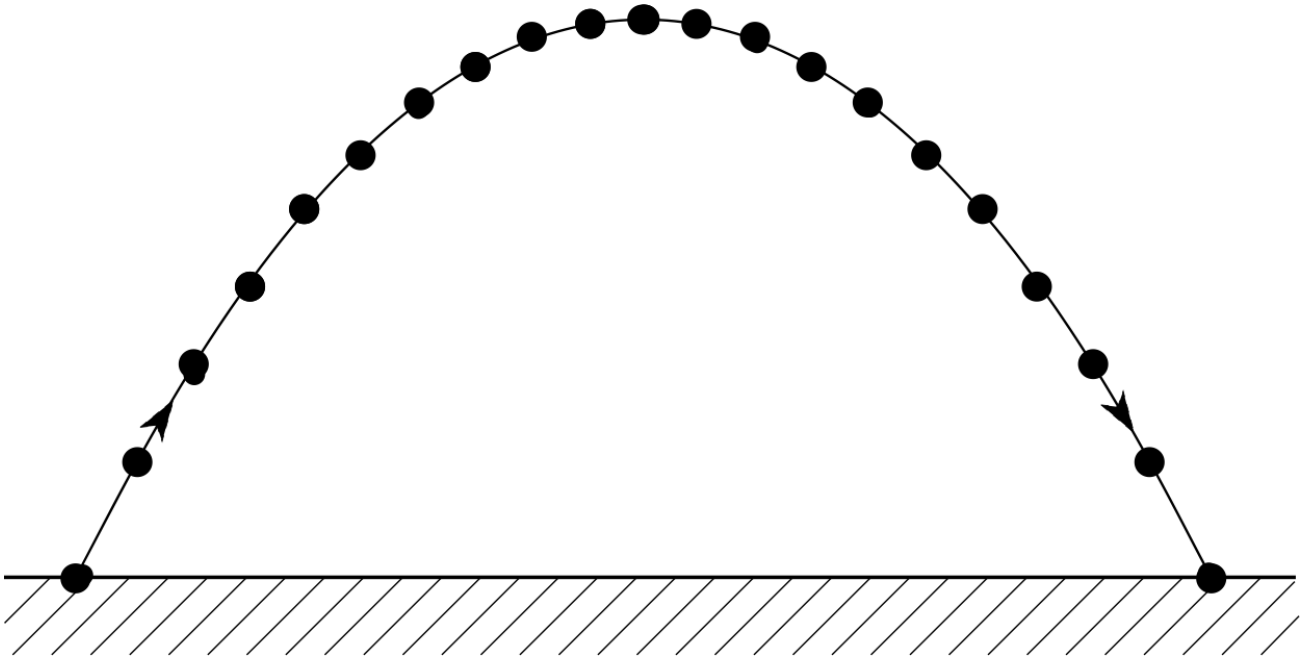
2D Image



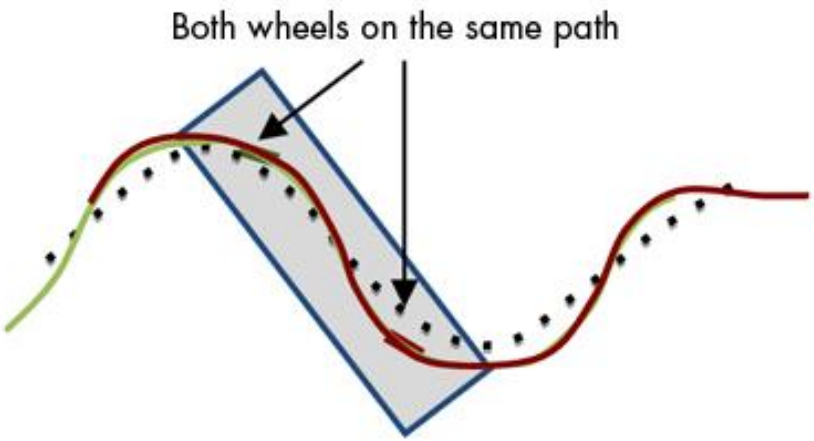
2D Scene

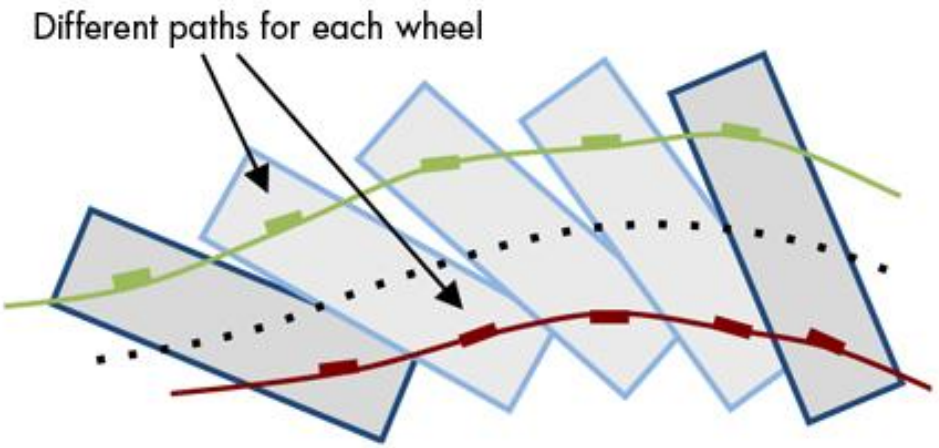
Understanding 2D Geometry (6)

- The spatial locations travelled or to be travelled are called paths.



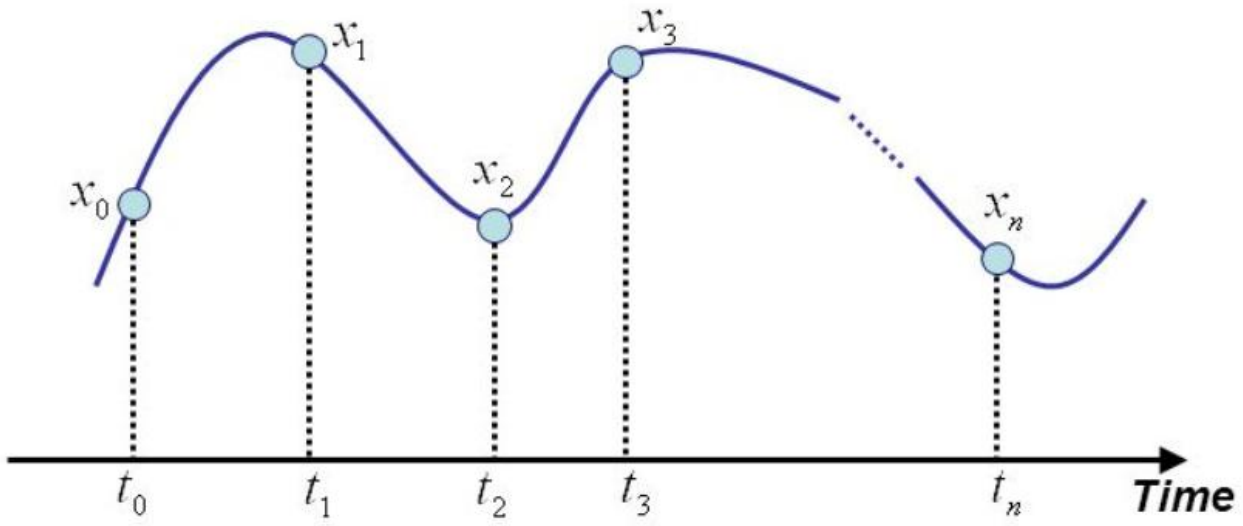
Example of Paths of a Mobile Base's Two Wheels …



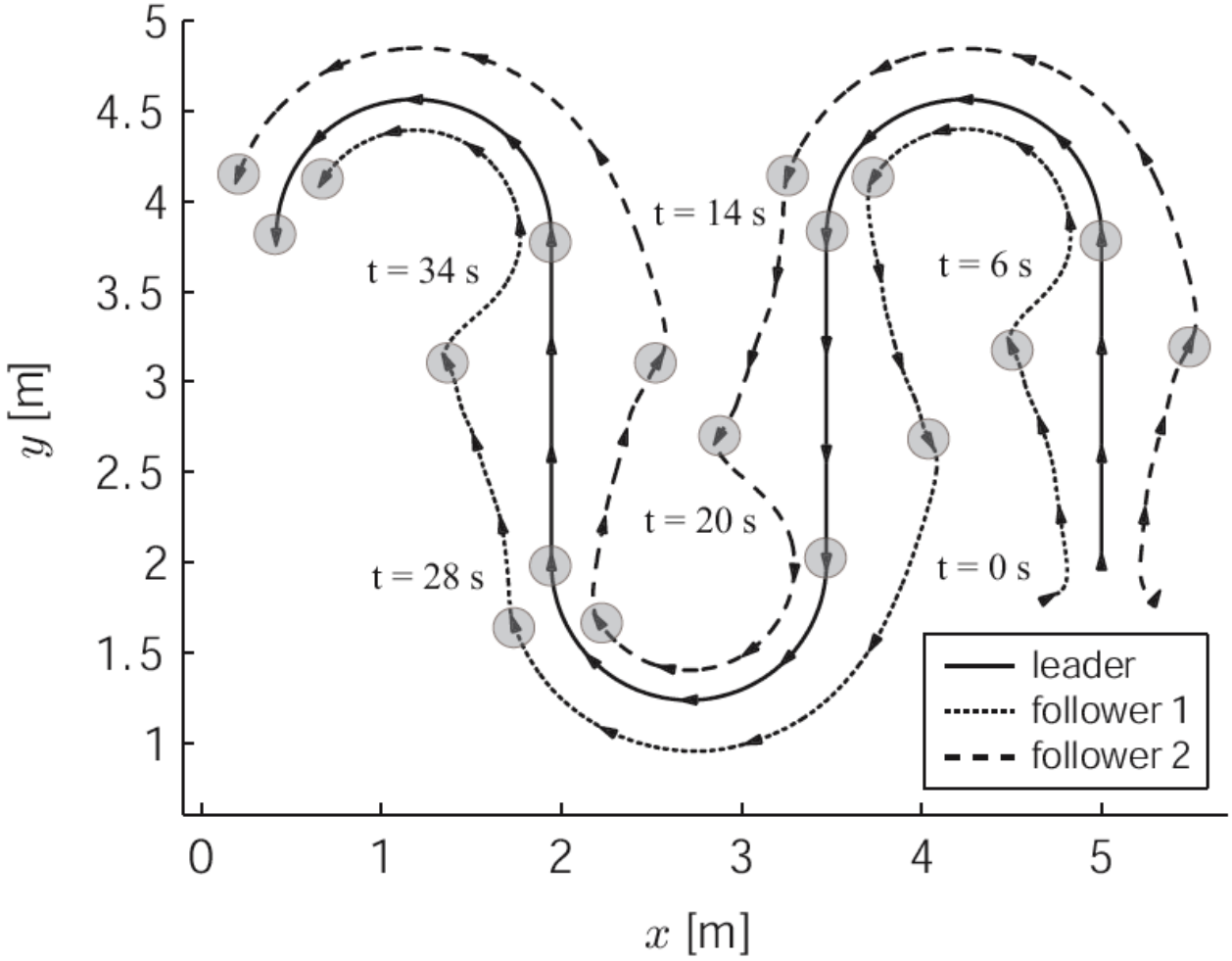


Understanding 2D Geometry (7)

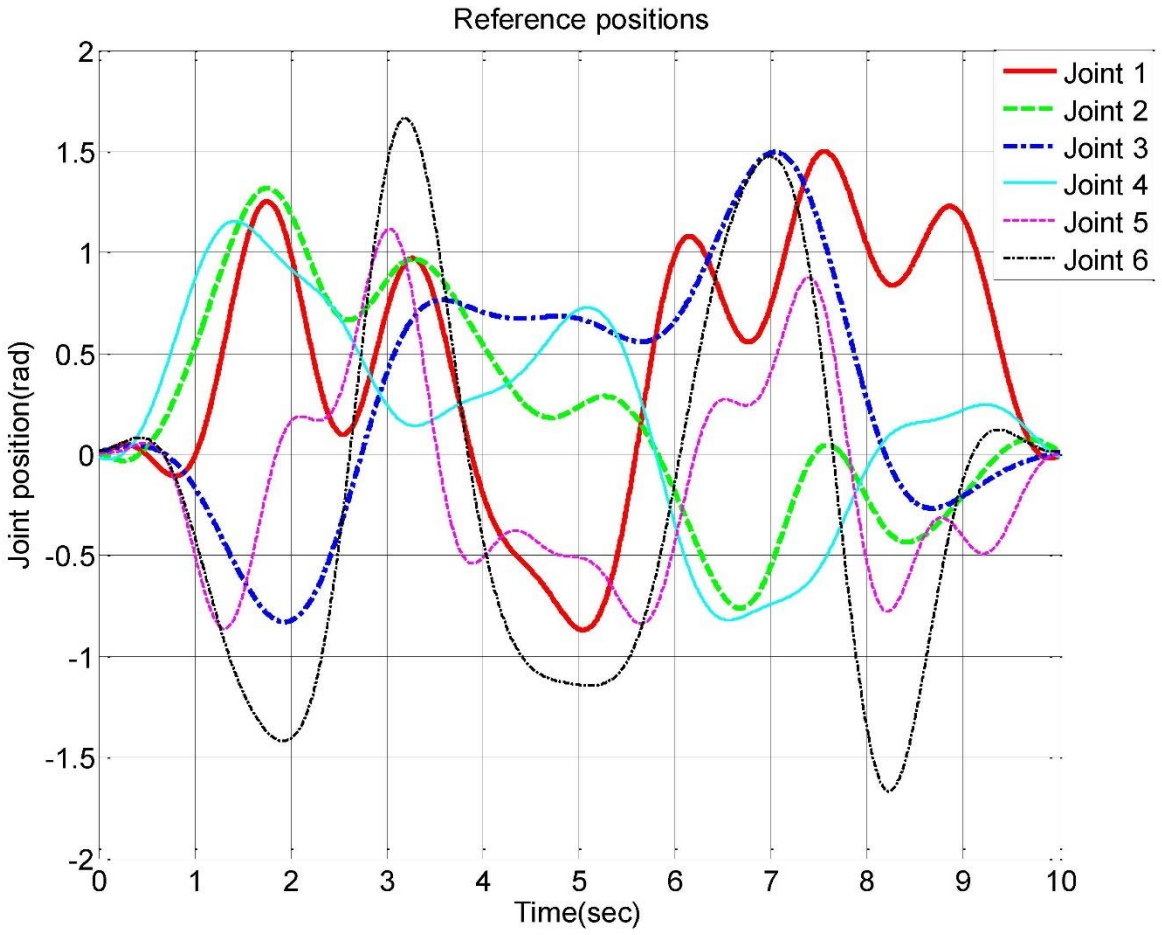
- The spatial locations with time constraint are called trajectories (i.e. equations of motions).



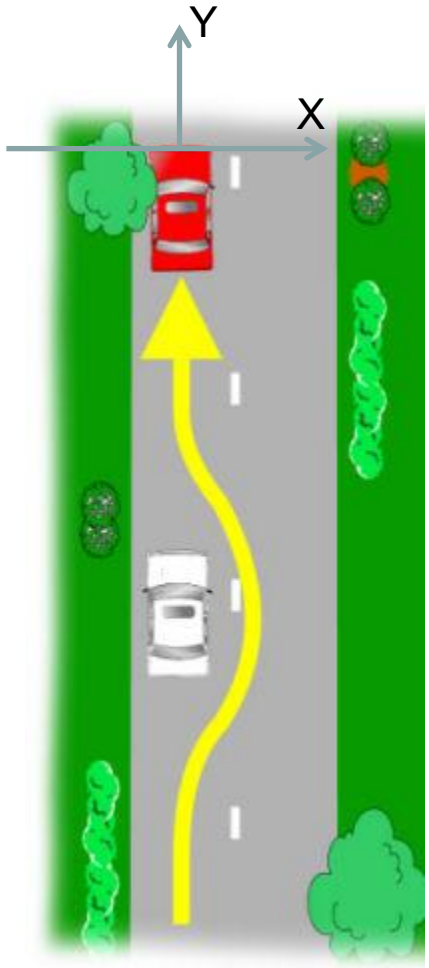
Example of Trajectories of Three Moving Platforms …

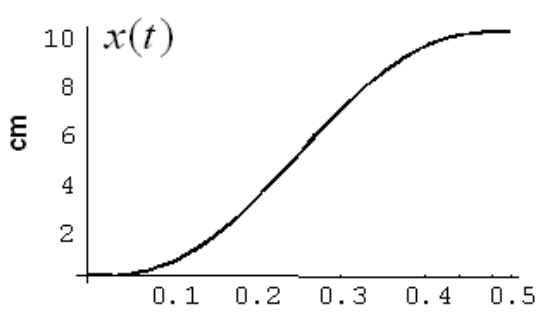


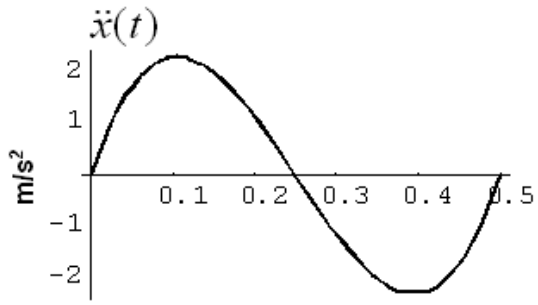
Example of Trajectories of a Humanoid Robot's Arm …

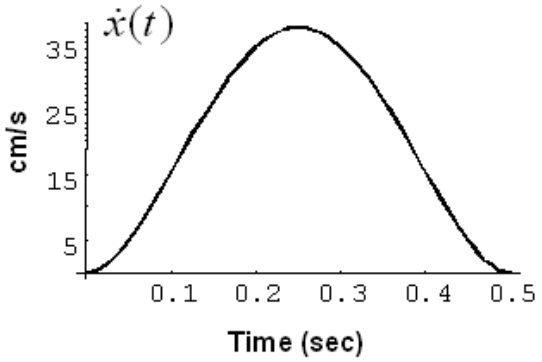


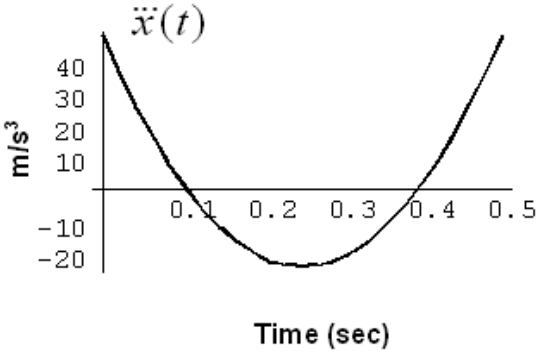
Example of Trajectories of a Vehicle's Body …



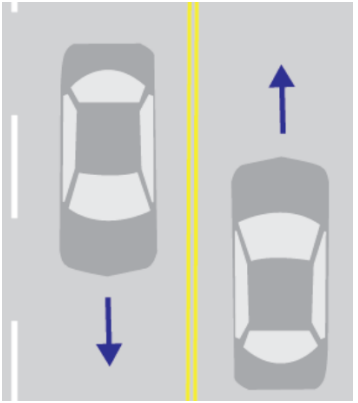




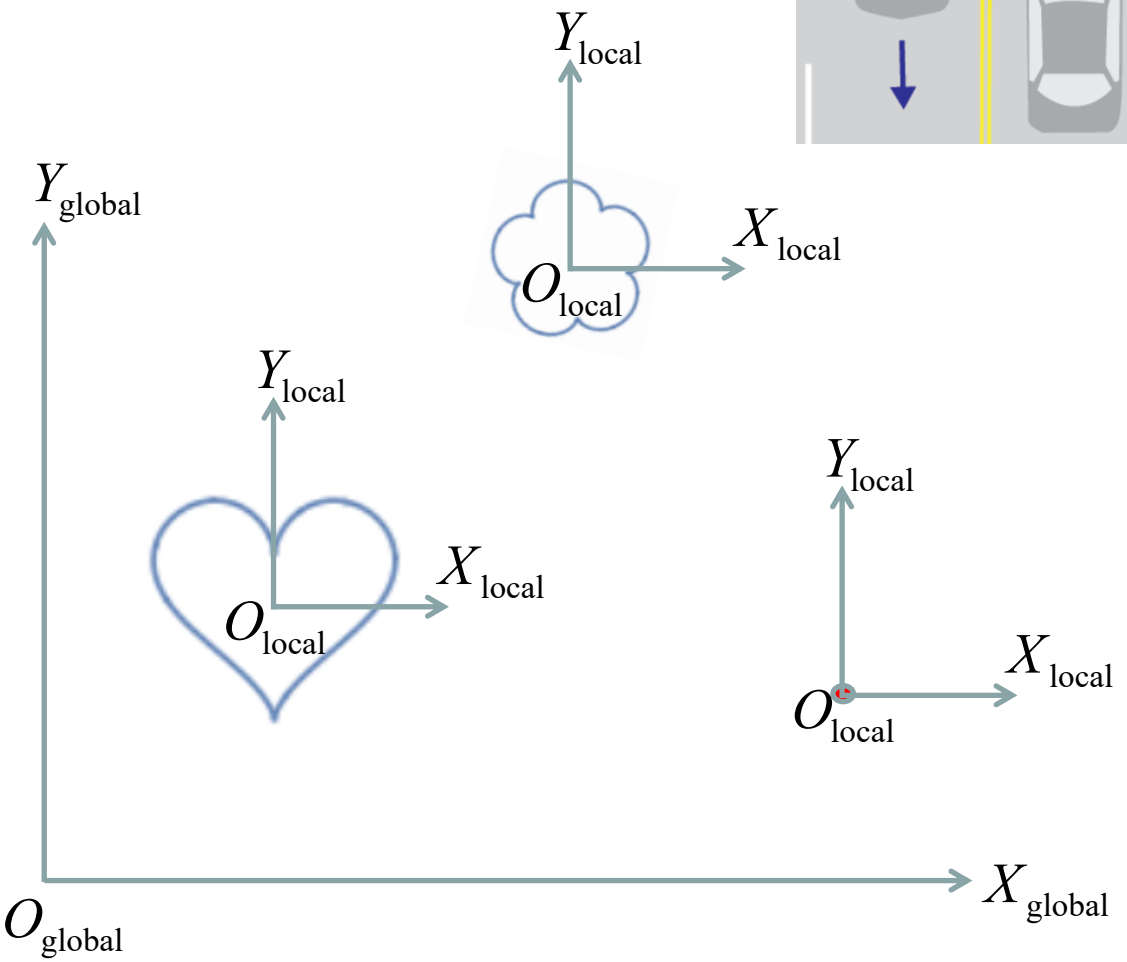




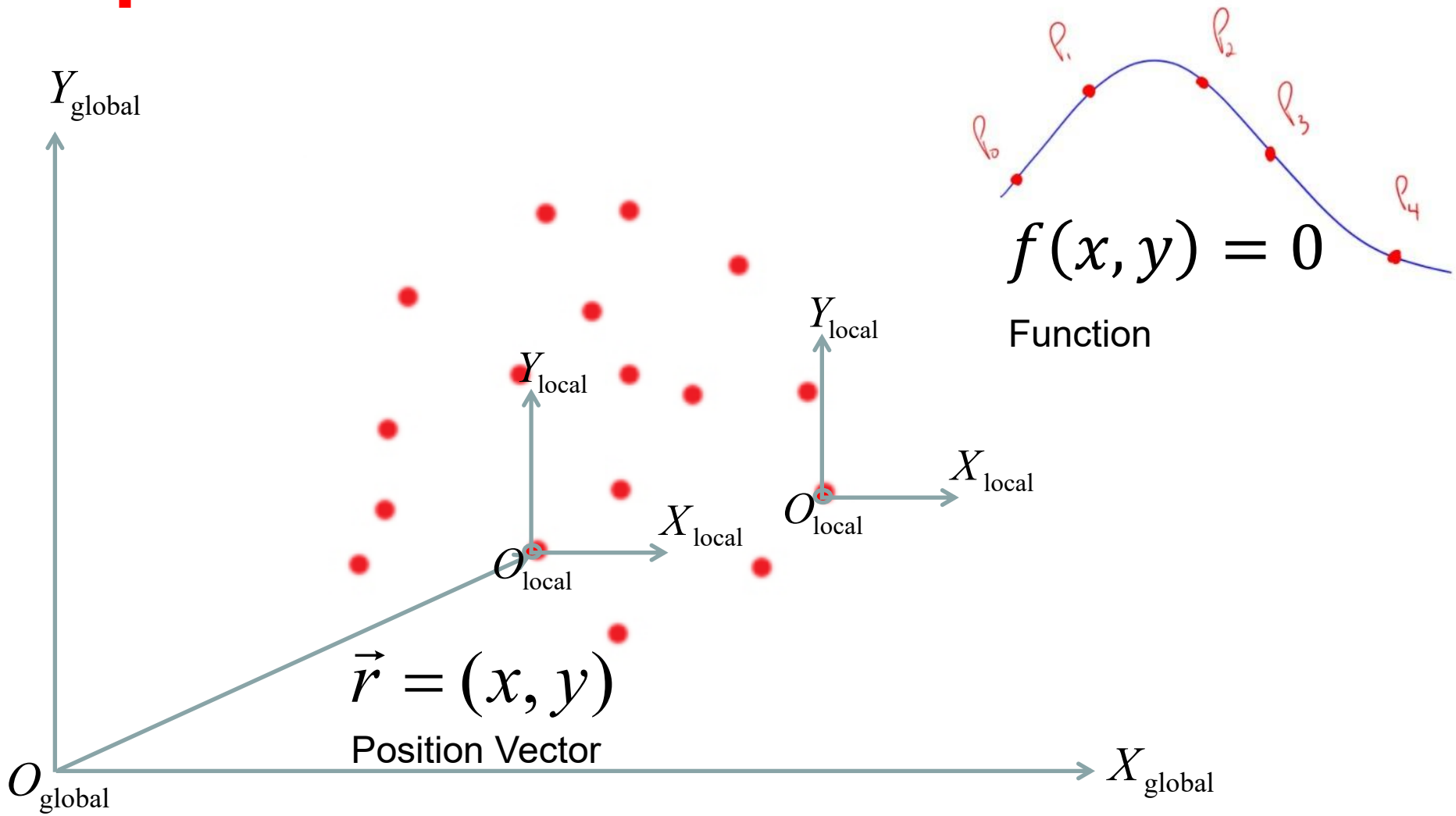
Two Types of Parameters in Geometry



- Pose:
  - It refers to positions and orientations of entities with respect to a Global Coordinate System.
- Shape:
  - It refers to outlines of entities with respect to Local Coordinate Systems assigned to these shapes, respectively.



Representation of Positions in 2D Space without Time Constraints

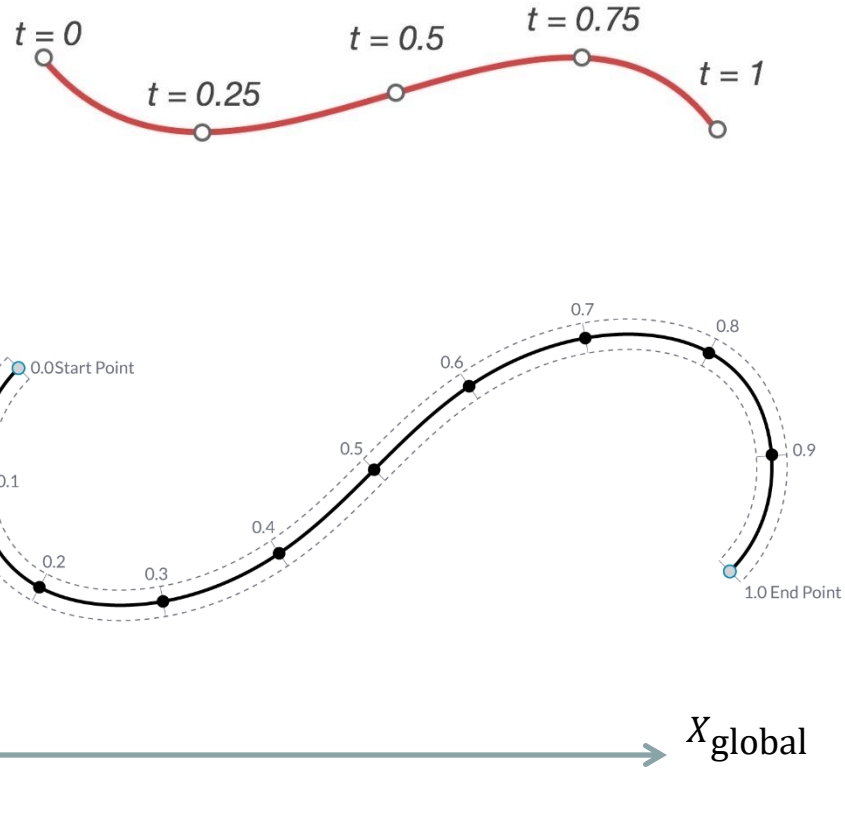


Representation of Positions in 2D Space with Time Constraints

$$\vec{r}(t) = \begin{pmatrix} x(t) \\ y(t) \end{pmatrix} = \begin{pmatrix} a_x t^3 + b_x t^2 + c_x t + d_x \\ a_y t^3 + b_y t^2 + c_y t + d_y \end{pmatrix}$$

$$\vec{r}(t) = \begin{pmatrix} x(t) \\ y(t) \end{pmatrix} = \begin{pmatrix} a_x t^2 + b_x t + c_x \\ a_y t^2 + b_y t + c_y \end{pmatrix}$$

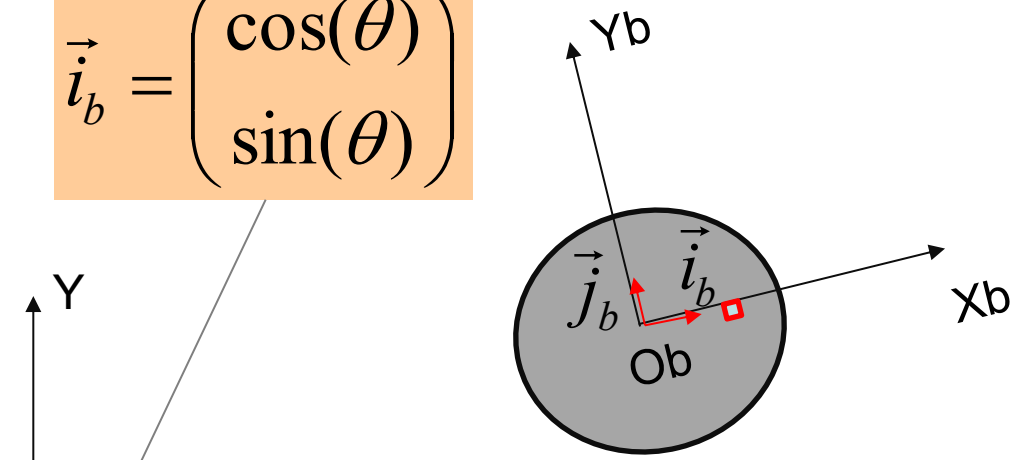
$$\vec{r}(t) = \begin{pmatrix} x(t) \\ y(t) \end{pmatrix} = \begin{pmatrix} a_x t + b_x \\ a_y t + b_y \end{pmatrix}$$

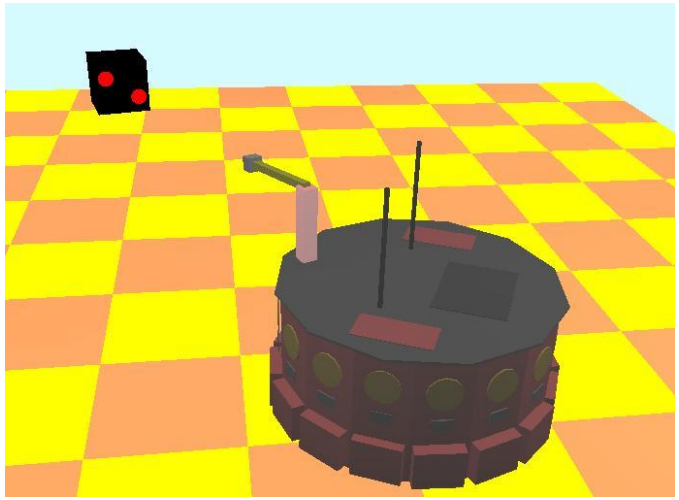


Representation of Orientations in 2D Space

$$\vec{j}_b = \begin{pmatrix} -\sin(\theta) \\ \cos(\theta) \end{pmatrix}$$

$$\vec{i}_b = \begin{pmatrix} \cos(\theta) \\ \sin(\theta) \end{pmatrix}$$





$$R_{t_0,t} = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix}$$

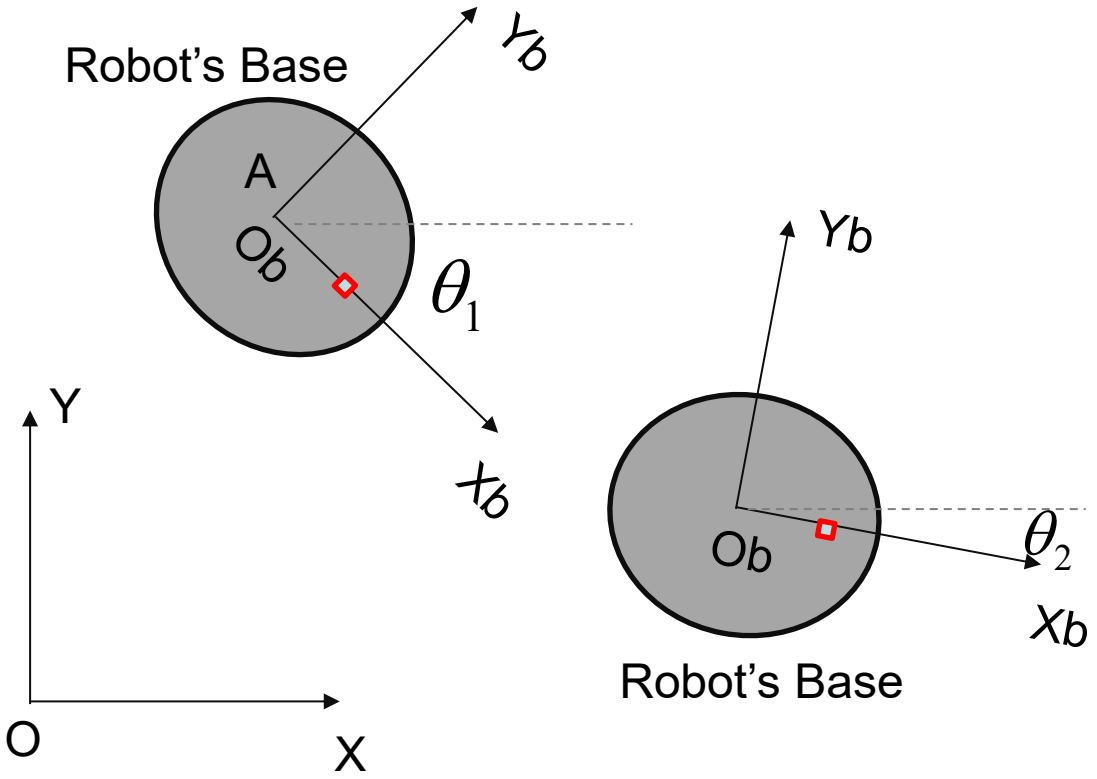
This is the orientation at time t, as the result of the change of orientation from time t0 to time t.

Example

- What are the orientations of the mobile base at times t1 and t2?
- Answer:

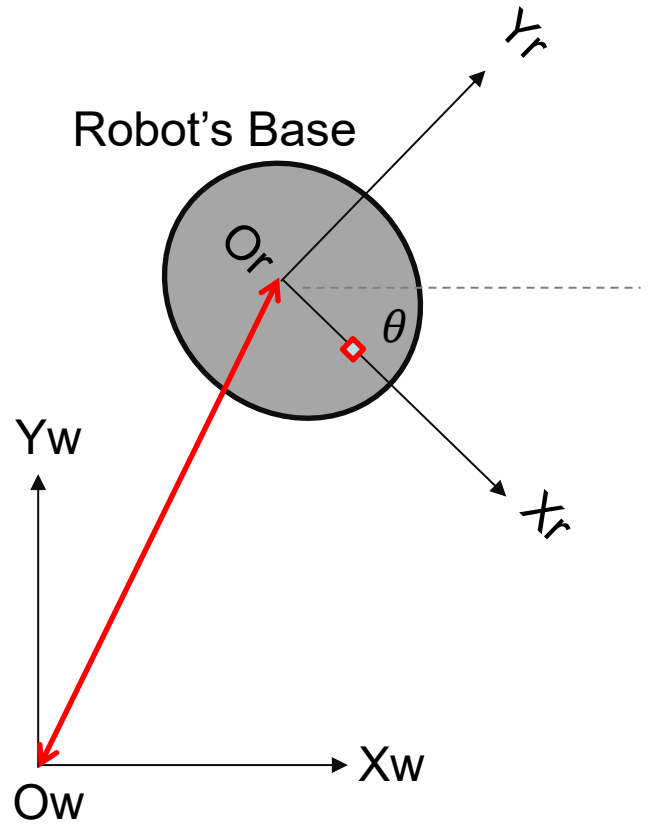
$$\theta(t_1) = \theta_1$$

$$\theta(t_2) = \theta_2$$



Example of Homogeneous Transformation Between Robot Frame and World Frame

This is the pose of the robot frame with respect to the world frame



$$H_{robot} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & x_r \\ \sin(\theta) & \cos(\theta) & y_r \\ 0 & 0 & 1 \end{bmatrix}$$

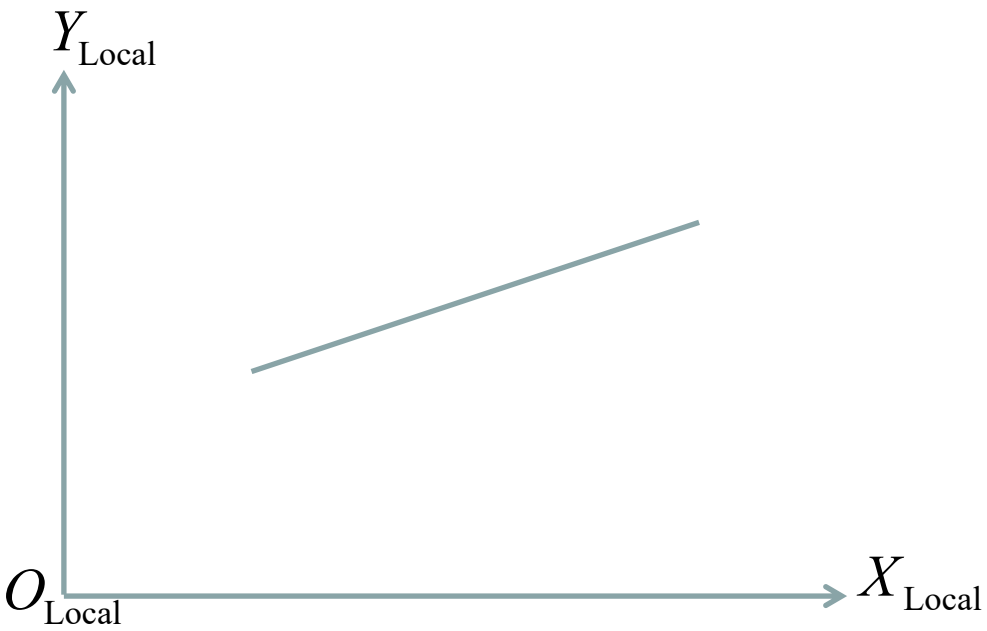
$$H_{world} = \begin{bmatrix} \cos(\theta) & \sin(\theta) & x_w \\ -\sin(\theta) & \cos(\theta) & y_w \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} x_w \\ y_w \end{bmatrix} = - \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} \times \begin{bmatrix} x_r \\ y_r \end{bmatrix}$$

$$H_{robot} \times H_{world} = I_{3\times3}$$

Representation of Line-Type Shape in 2D Space

$$y = ax + b$$

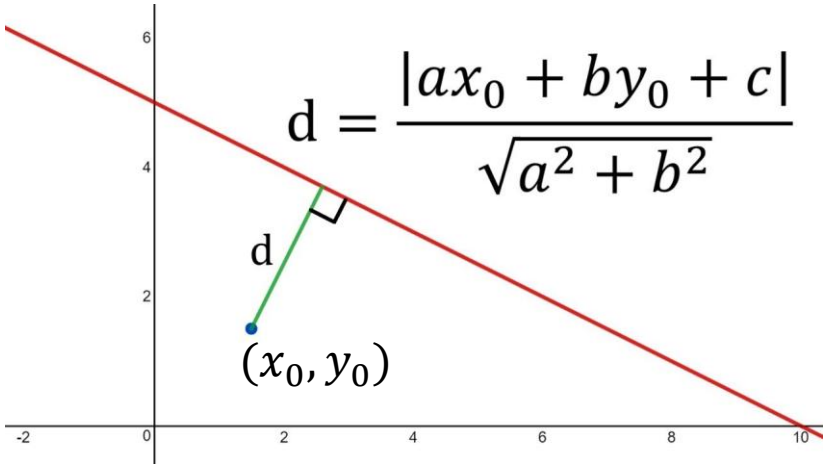


Or:

$$ax + by + c = 0$$

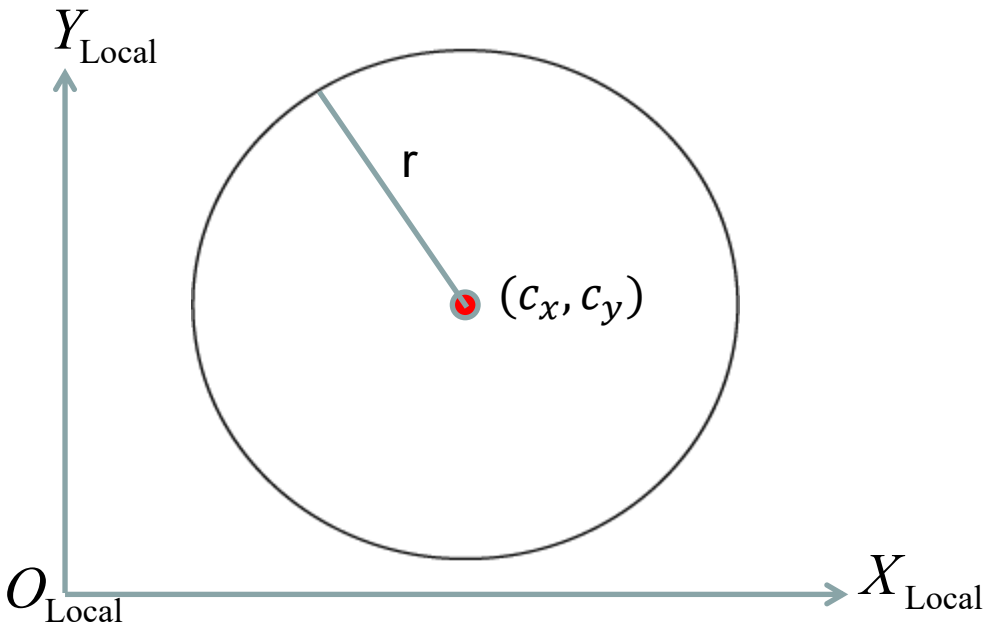
Or:

$$ax + by + c = 1$$



Representation of Circle-Type Shape in 2D Space

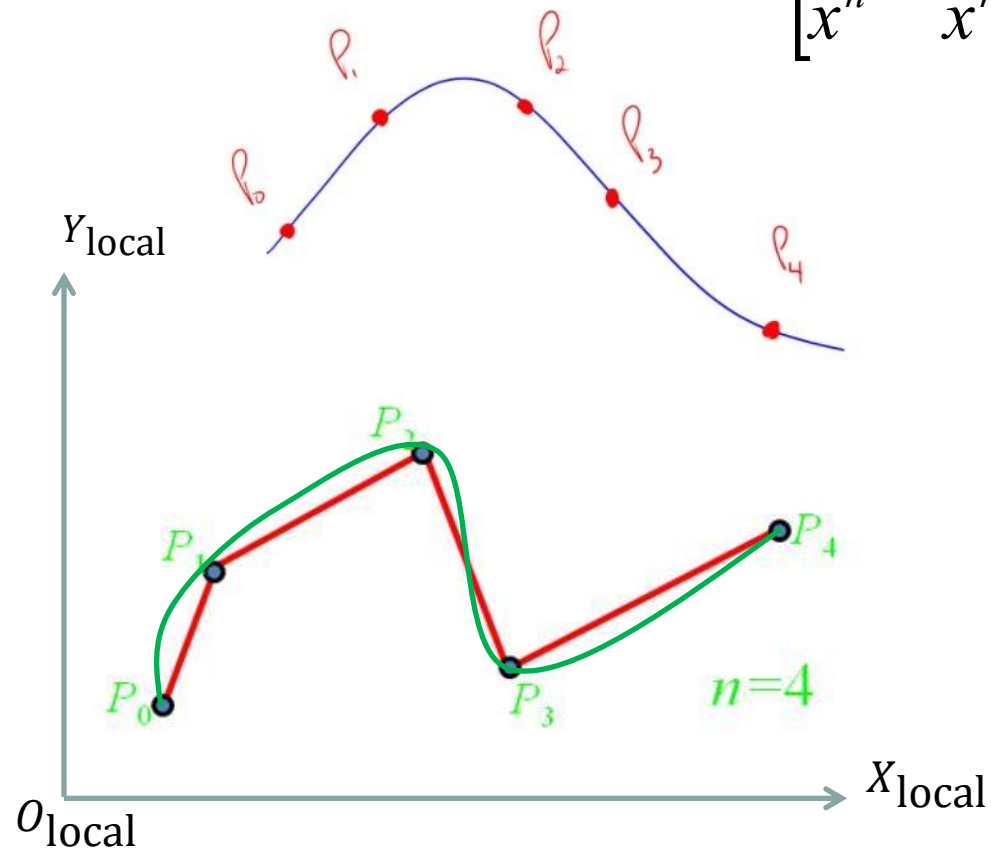
$$(x - c_x)^2 + (y - c_y)^2 = r^2$$



Representation of Curve-Type Shape in 2D Space

$$\begin{bmatrix} x^n & x^{n-1} & \dots & 1 \end{bmatrix} P_{n\times n} \begin{bmatrix} y^n \\ y^{n-1} \\ \dots \\ 1 \end{bmatrix} = 0$$

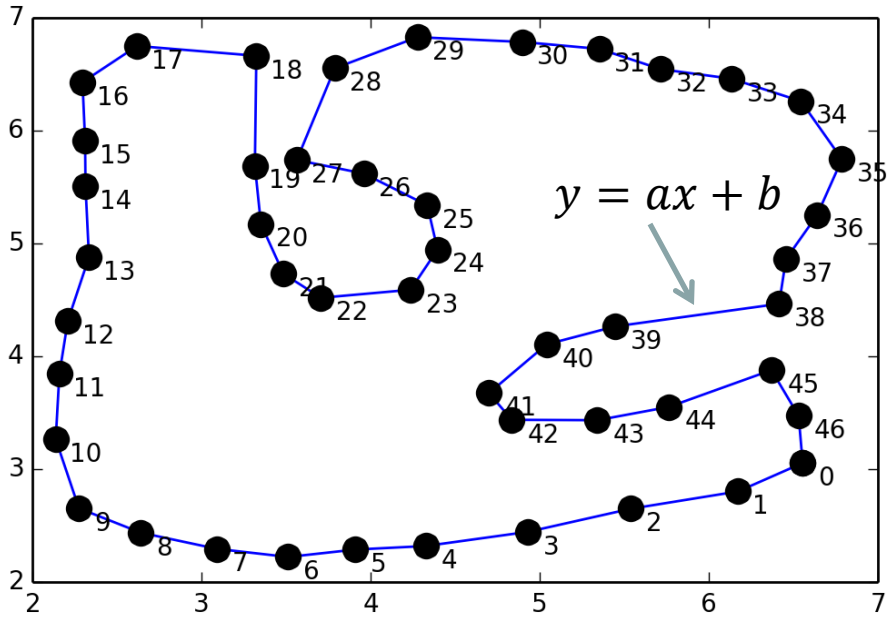
n=1,2,…

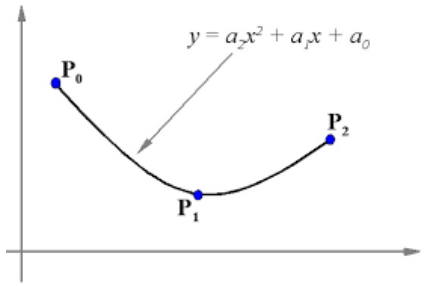


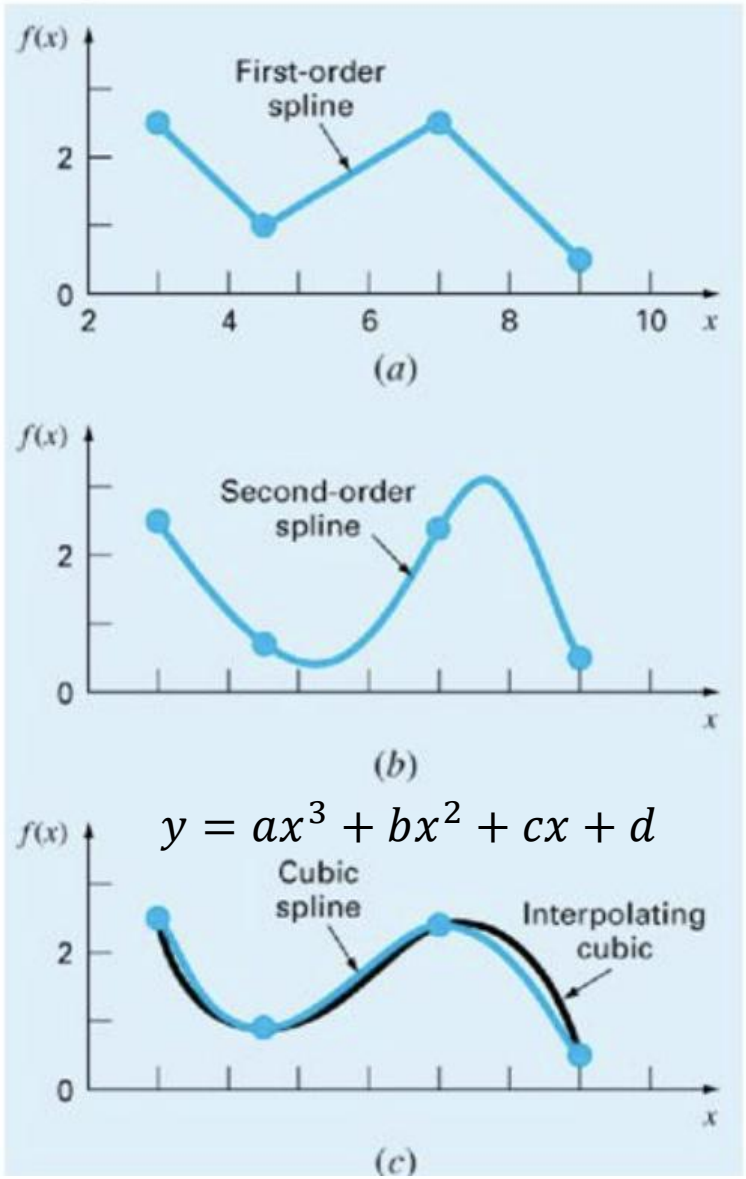
Any curve could be approximated by a series of interconnected line segments or curve segments.

Next Slide

Example of Using Splines …







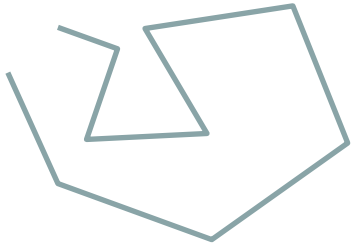
Representation of Curve-Type Shape

Data Structure of 2D Point

```
typedef struct
{
        short     u, v ;
        double  x, y ;
}
2DPoint ;
```

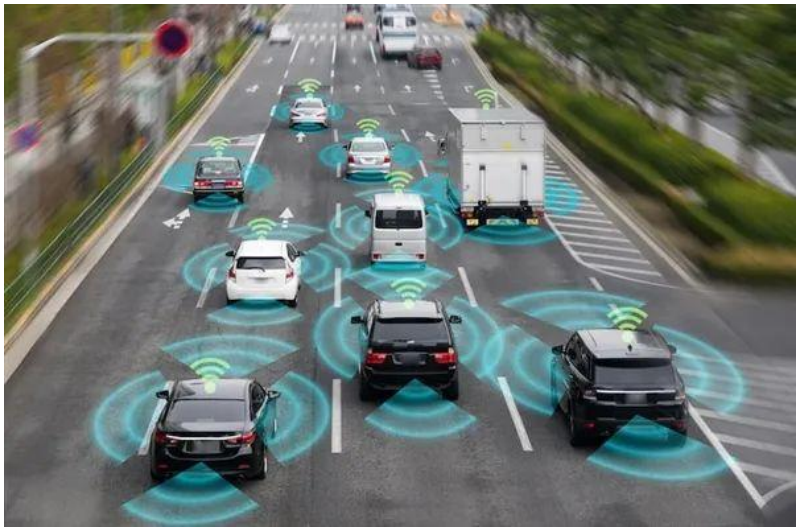
Data Structure of 2D Curve

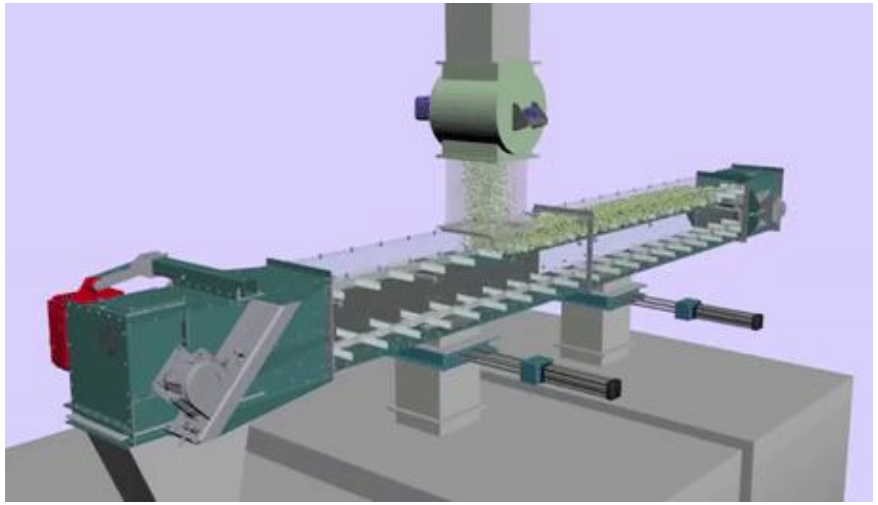
```
typedef struct
{
        2DPoint point_list[];
        short     number_point;
        short     curve_type;
}
2DCurve;
```



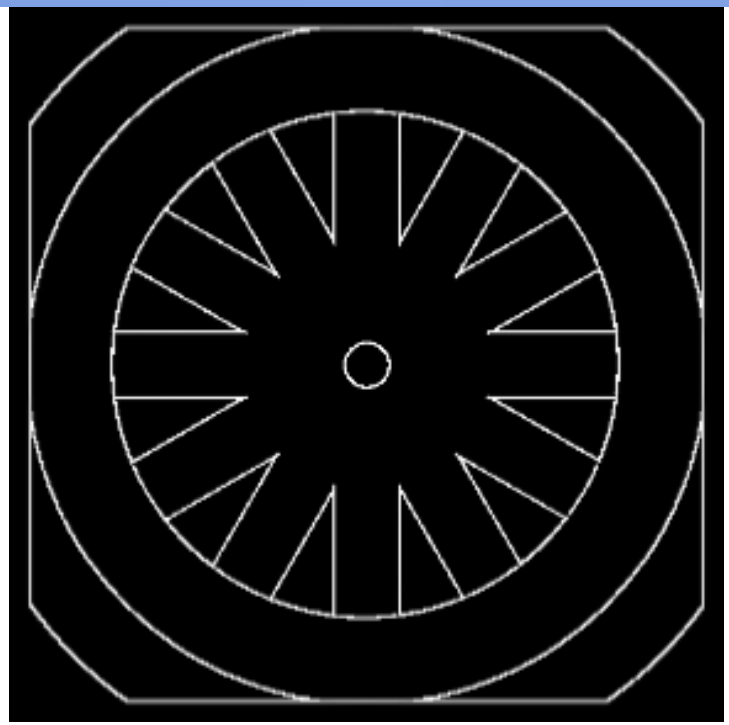
Outline of Lecture 1

- Concept of Knowledge Space
- Representation of 2D Geometry
- Computation of 2D Geometry
- Representation of 3D Geometry
- Computation of 3D Geometry

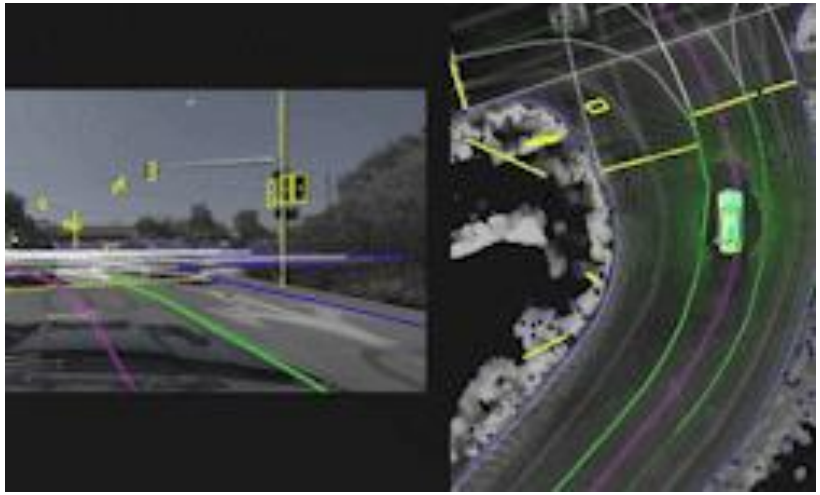




Three Questions …



- What are the available input?
- What are the expected outcomes?
- What are the solutions which will produce the expected outcomes from the available input?



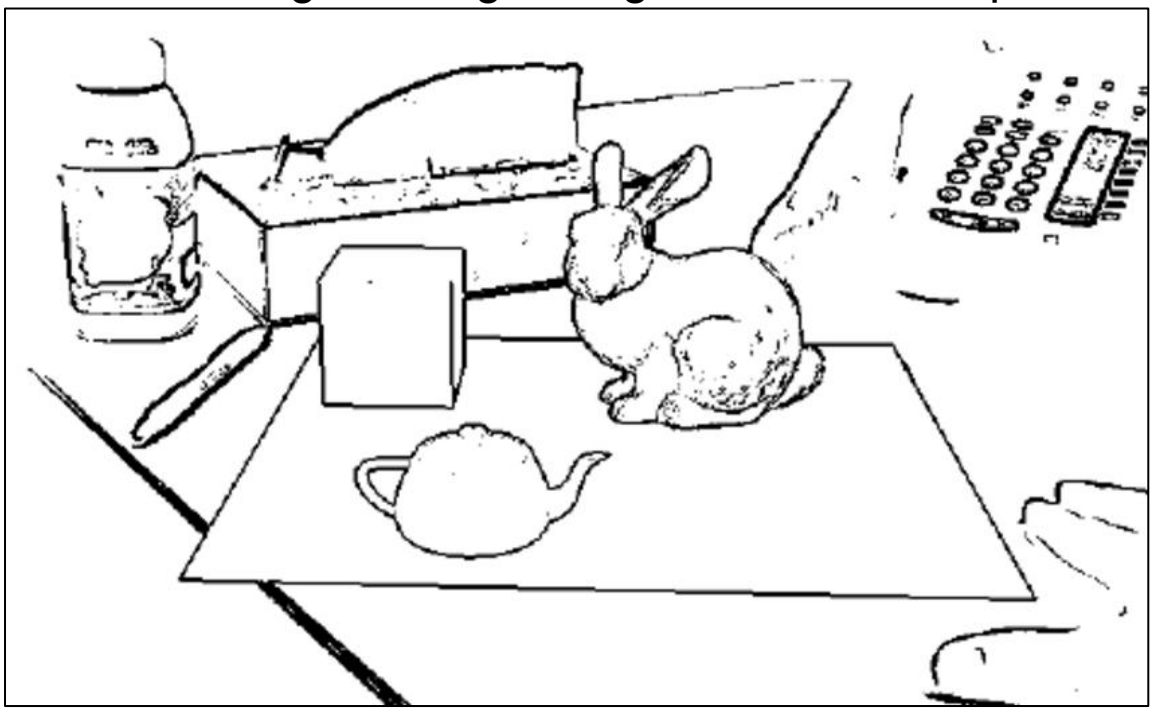
Challenge: How to obtain input?

The general approach consists of two steps:

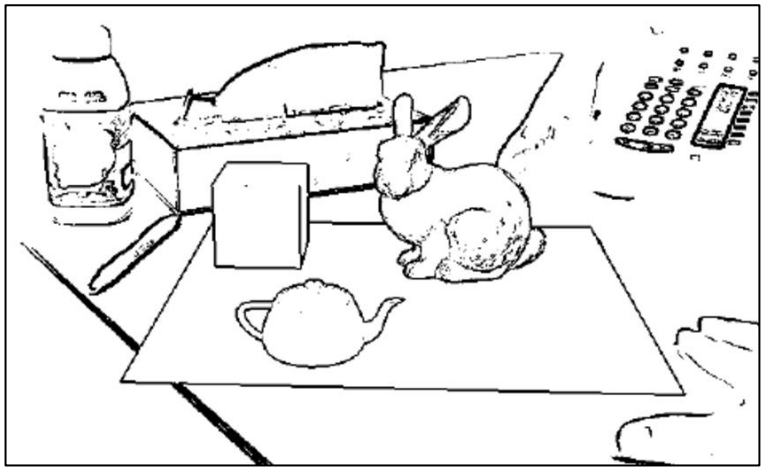
- Edge Detection
- Edge Linking

Challenge

Color Image → Edge Image → Curve Samples



How to transform color image into edge image?



- Edge Detection Algorithm:

Step 1: Compute Horizontal Edges.

Step 2: Select the local maxima along the vertical axis as the edge pixels.

Step 3: Compute Vertical Edges.

Step 4: Select the local maxima along the horizontal axis as the edge pixels.

Step 5: Combine all the edge pixels to form the output.

Sample Program of Doing Edge Detection …

```
void EdgeDetection(void)
{
        int  i, j, v ;

        // Compute Horizontal Edges
        memset(gpBuffer1, 255, 512*512) ;
        for (i = 1 ; i < 511 ; i++)
        {
            for (j = 1 ; j < 511 ; j++)
            {
                v = abs(gpBuffer0[(i-1)*512+j-1] + pBuffer0[(i-1)*512+j] +  gpBuffer0[(i-1)*512+j+1] -
                    gpBuffer0[(i+1)*512+j-1] - gpBuffer0[(i+1)*512+j] - gBuffer0[(i+1)*512+j+1]) ;
                if (v >= 255)   gpBuffer1[i*512+j] = 0;
                if (v > 35 && v < 255) gpBuffer1[i*512+j] = (unsigned char) (255 - v) ; // change it to negative edge map
            }
        }
        SelectMaximaAlongColumn() ; // select local minima negative edge-map

        // Compute Vertical Edges
        memset(gpBuffer2, 255, 512*512) ;
        for (i = 1 ; i < 511 ; i++)
        {
            for (j = 1 ; j < 511 ; j++)
            {
                v = abs(gpBuffer0[(i-1)*512+j-1] + gpBuffer0[(i)*512+j-1] + gpBuffer0[(i+1)*512+j-1] -
                    gpBuffer0[(i-1)*512+j+1] - gpBuffer0[(i)*512+j+1] - gpBuffer0[(i+1)*512+j+1]) ;
                if (v >= 255)   gpBuffer2[i*512+j] = 0 ;
                if (v > 35 && v < 255) gpBuffer2[i*512+j] = (unsigned char) (255 - v) ; // change it to negative edge map
            }
        }
        SelectMaximaAlongRow() ; // select local minima in negative edge-map

        // Display Horizontal Edges

        // Display Vertical Edges

        // Combined Edgemap
        memset(gpBuffer0, 255, 512*512) ;
        for (i = 0; i < 512*512 ; i++)
            if (gpBuffer1[i]  == 0 || gpBuffer2[i] == 0) gpBuffer0[i] = 0 ; // combine local minima in negative edge-map

}
```

Set output edge-map's background to be white

*

+1	+1	+1
0	0	0
-1	-1	-1

Ideal horizontal edge

*

+1	0	-1
+1	0	-1
+1	0	-1

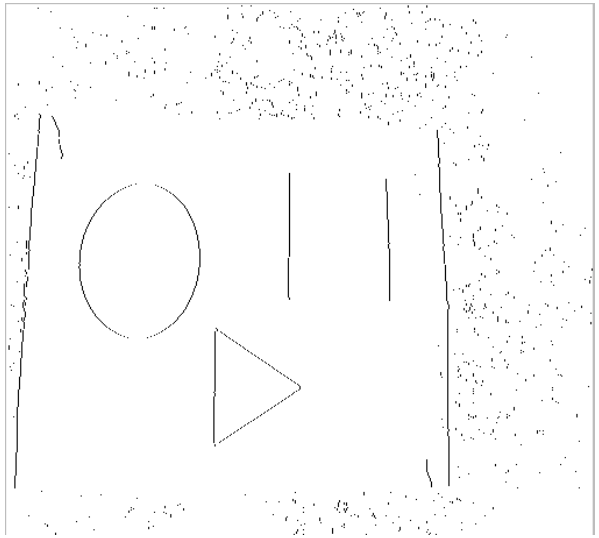
Ideal vertical edge

```
static SelectMaximaAlongColumn() // select local minima in negative edge-map
{
        int         i, j, v ;
        unsigned short temporaryBuffer[512*512] ;

        // step 1: smoothing in vertical direction
        memset(temporaryBuffer, 255*5, 512*512) ;
        for (i = 2 ; i < 512 - 2; i++)
        {
            for (j = 0 ; j < 512; j++)   // compute weighted sum
            {
                    temporaryBuffer[i*512+j] = 1*gpBuffer1[(i-2)*512+j] +
                                               2*gpBuffer1[(i-1)*512+j] +
                                               3*gpBuffer1[i*512+j] +
                                               2*gpBuffer1[(i+1)*512+j] +
                                               1*gpBuffer1[(i+2)*512+j] ;
            }
        }

        // step 2: select local minima in vertical direction
        memset(gpBuffer1, 255, 512*512) ;
        for (i = 2 ; i < 512 - 2; i++)
        {
            for (j = 0 ; j < 512; j++)
            {
                    v = temporaryBuffer[i*512+j] ;
                    if ( v < temporaryBuffer[(i-2)*512+j] &&
                         v < temporaryBuffer[(i-1)*512+j] &&
                         v < temporaryBuffer[(i+1)*512+j] &&
                         v < temporaryBuffer[(i+2)*512+j])
                    {
                            gpBuffer1[i*512+j] = (unsigned char) 0 ;  //use 0 as minimum value
                    }
            }
        }
}
```

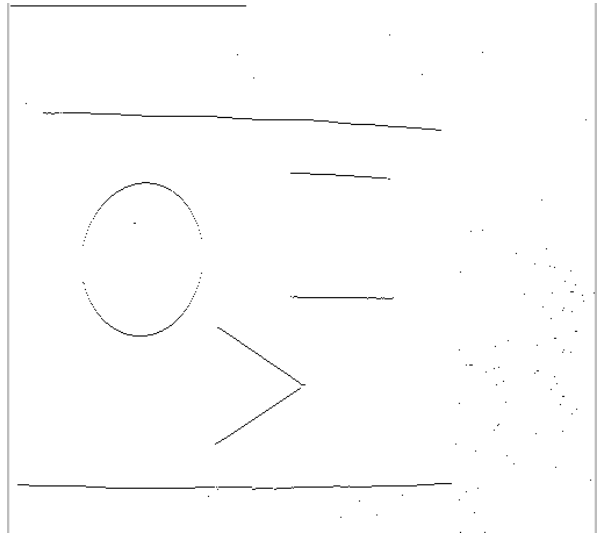
```
static SelectMaximaAlongRow() // select local minima in negative edge-map
{
        int         i, j, v ;
        unsigned short temporaryBuffer[512*512] ;

        // step 1: smoothing in horizontal direction
        memset(temporaryBuffer, 255*5, 512*512) ;
        for (i = 0 ; i < 512; i++)
        {
            for (j = 2 ; j < 512 - 2 ; j++)  // compute weighted sum
            {
                    temporaryBuffer[i*512+j] = 1*gpBuffer2[i*512+j-2] +
                                               2*gpBuffer2[i*512+j-1] +
                                               3*gpBuffer2[i*512+j] +
                                               2*gpBuffer2[i*512+j+1] +
                                               1*gpBuffer2[i*512+j+2] ;
            }
        }

        // step 2: select local minima in horizontal direction
        memset(gpBuffer2, 255, 512*512) ;
        for (i = 0 ; i < 512; i++)
        {
            for (j = 2 ; j < 512 - 2 ; j++)
            {
                    v = temporaryBuffer[i*512+j] ;
                    if ( v < temporaryBuffer[i*512+j-2] &&
                         v < temporaryBuffer[i*512+j-1] &&
                         v < temporaryBuffer[i*512+j+1] &&
                         v < temporaryBuffer[i*512+j+2])
                    {
                            gpBuffer2[i*512+j] = (unsigned char) 0 ; //use 0 as minimum value
                    }
            }
        }
}
```

Example of Results

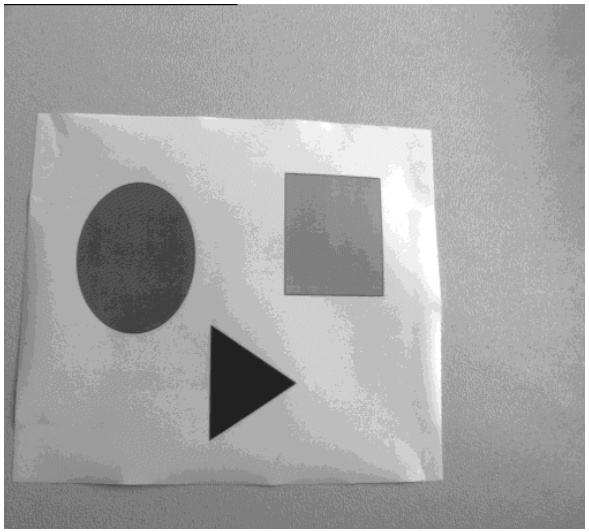
Vertical Edges



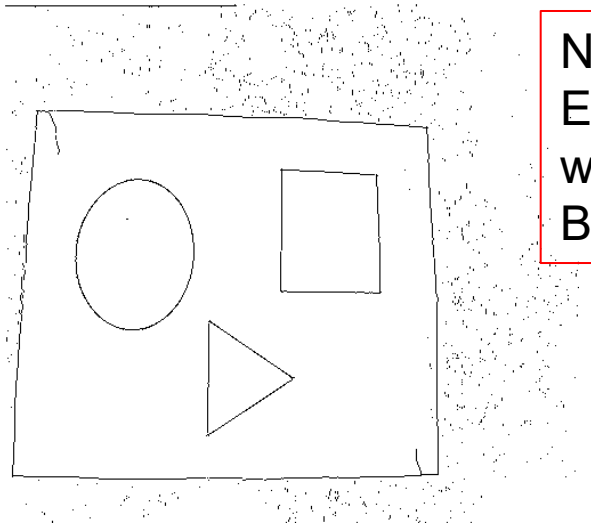
Horizontal Edges



Input Image

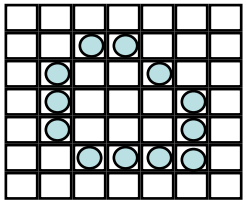


Combined Edges



Note:
Edge-Maps with White Background

How to transform edge image into curve samples?

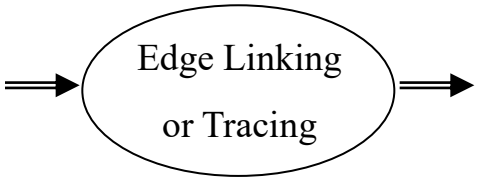


Main idea of doing edge linking:

- Scan input image of edges.
- If edge pixel at (i,j) is unlabeled, label it and define it as the currently labeled pixel.
- From the eight neighboring locations of (i,j), choose an edge pixel according to a user-defined condition (e.g. nearest to the center of unlabeled edges within a window).
  - If such a pixel exists, label it and define it as the currently-labeled edge pixel, and repeat the procedure of edge linking;
  - Otherwise, a curve sample has been traced (i.e. found) and labeled. Repeat the procedure of scanning the input image of edges.

Illustration of Linked Edges (i.e. Curve Samples)

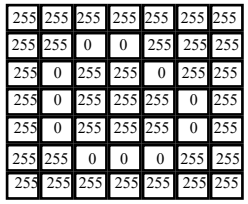
	0	1	2	3	4	5	6	7	8	9	10	11
0	255	255	255	255	255	255	255	255	255	255	255	255
1	255	0	0	0	0	255	255	255	0	0	255	255
2	255	0	255	255	0	255	255	0	255	255	0	255
3	255	0	255	255	0	255	255	0	255	255	0	255
4	255	0	255	255	0	255	255	0	255	255	0	255
5	255	0	255	255	0	255	255	255	0	0	255	255
6	255	0	255	255	0	255	255	255	255	255	255	255
7	255	0	0	0	0	255	255	255	255	255	255	255
8	255	255	255	255	255	255	255	255	255	255	255	255
9	255	255	255	255	255	255	255	255	255	255	255	255



	0	1	2	3	4	5	6	7	8	9	10	11
0	255	255	255	255	255	255	255	255	255	255	255	255
1	255	1	1	1	255	255	255	255	2	2	255	255
2	255	1	255	255	1	255	255	2	255	255	2	255
3	255	1	255	255	1	255	255	2	255	255	2	255
4	255	1	255	255	1	255	255	2	255	255	2	255
5	255	1	255	255	1	255	255	255	2	2	255	255
6	255	1	255	255	1	255	255	255	255	255	255	255
7	255	255	1	1	255	255	255	255	255	255	255	255
8	255	255	255	255	255	255	255	255	255	255	255	255
9	255	255	255	255	255	255	255	255	255	255	255	255

Note:
Edge-Maps with White Background

An Edge Linking Algorithm …



255	255	255	255	255	255	255
255	255	0	0	255	255	255
255	0	255	255	0	255	255
255	0	255	255	255	0	255
255	0	255	255	255	0	255
255	255	0	0	0	255	255
255	255	255	255	255	255	255

Step 1: Initialize the label-map and set the label variable to 1 (n = 1).

Step 2: Scan the edge-map row by row and column by column.

Step 3: If encounter an unlabeled edge-pixel, define it as the current edge-pixel.
Compute the estimated center of unlabeled edges within a window which is centered at the current edge-pixel.

Step 4: Assign the label "n" to the current edge-pixel. Now, this edge is labeled.

Step 5: Find all the unlabeled neighbors around the current edge-pixel.
If there is no unlabeled neighbor, a curve sample has been found and go to Step 7.

Step 6: Among the unlabeled neighbors, choose the one which is the nearest edge to the estimated center. Define it as the new current edge-pixel.
Repeat Step 4, Step 5 and Step 6.

Step 7: Increment the label (n = n + 1) and continue from Step 2.

Xie M. and Thonnat M. 1992, An Algorithm for Finding Closed Curves, Pattern Recognition Letters, Vol. 13, No. 1, pp. 73-81.

Program of Doing Edge Linking …

Edge-Map

255	255	255	255	255	255	255
255	255	0	0	255	255	255
255	0	255	255	0	255	255
255	0	255	255	255	0	255
255	0	255	255	255	0	255
255	255	0	0	0	255	255
255	255	255	255	255	255	255

Label-Map

255	255	255	255	255	255	255
255	255	255	255	255	255	255
255	255	255	255	255	255	255
255	255	255	255	255	255	255
255	255	255	255	255	255	255
255	255	255	255	255	255	255
255	255	255	255	255	255	255

```
/* assuming that the pixel values of all the edge pixels are "0" */

unsigned char edgemap[512*512] ;
unsigned char labelmap[512*512] ;

static  int       cx, cy ;        /* the estimated center of unlabeled edges      */
static  int       newx, newy ;  /* the location of next "current edge-pixel"    */
static  double  min_distance ;
static  int       sx, sy ;

static  void  ComputeCenterWithinWindowAt(int x0, int y0)
{
        /* update the center's coordinates (pcx, pcy) */
}

static  void  VerifyNextUnlabeledEdge(int x, int y)
{
       /* verify the neighbor whether it is the nearest one to (pcx, pcy) */
}

static void  LinkEdge(int x, int y, int n)
{
      /* recursively link the edge-pixels */
}

main(int argc, char **argv)
{
      int    x,  y, n ;

      n = 1 ;   sx = 10 ;      sy = 10 ;          // set the size of window to be 10x10
      memset(labelmap, 255, 512*512) ;   // clear the content of labelmap

      for (y = 1 ; y < 511 ; y++)
            for (x = 1 ; x < 511 ; x++)
                  if (edgemap[y*512+x] == 0 && labelmap[y*512+x] == 255)
                  {
                        ComputeCenterWithinWindowAt(x, y);
                        LinkEdgeAtCurrentLocation(x, y, n) ; // do edge linking
                        n++ ;
                  }
}
```

Encounter an Unlabeled Edge

```
static  void   VerifyNextUnlabeledEdge(int x, int y)
{
        double  distance ;
        // the candidate must be unlabeled edge
        if (edgemap[y*512+x] != 0 || labelmap[y*512+x] != 255)
              return ;

        distance = sqrt((cx - x)*(cx - x) + (cy - y)*(cy - y)) ;
        if (distance < min_distance)
        {
                    min_distance = distance;
                    newx = x ;
                    newy = y ;
        }
}
```

Edgemap

255	255	255	255	255	255	255
255	255	0	0	255	255	255
255	0	255	255	0	255	255
255	0	255	255	255	0	255
255	0	255	255	255	0	255
255	255	0	0	0	255	255
255	255	255	255	255	255	255

Labelmap

255	255	255	255	255	255	255
255	255	255	255	255	255	255
255	255	255	255	255	255	255
255	255	255	255	255	255	255
255	255	255	255	255	255	255
255	255	255	255	255	255	255
255	255	255	255	255	255	255

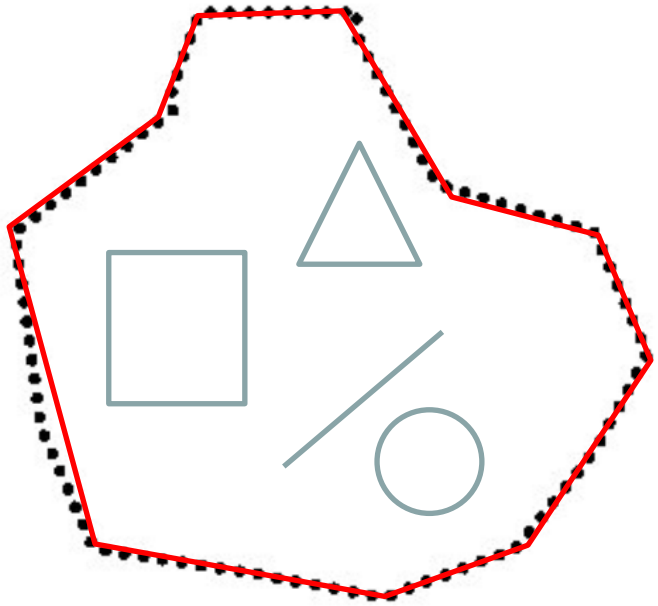
```
static  void  ComputeCenterWithinWindowAtCentre(int x0, int y0)
{
        int  r, c, n ;
        // use unlabeled edges to compute the center inside window
        cx = 0 ; cy = 0 ; n = 0 ;
        for (r = -sy/2 ; r < sy/2 ; r++)  // r is the increment of row index
              // c is the increment of column index
              for (c = -sx/2; c < sx/2 ; c++)
                    if (edgemap[(y0+r)*512+x0+c] == 0 &&
                        labelmap[(y0+r)*512+x0+c] == 255)
                    {
                        cx = cx + x0 + c ;
                        cy = cy + y0 + r ;
                        n = n + 1 ;
                     }
        if (n != 0)
        {
             cx = cx/n ;
             cy = cy/n ;
         }
}
```

```
static void  LinkEdgeAtCurrentLocation(int x, int y, int n)
{
        labelmap[y*512+x] = n ;    /*  assign label to the edge  */

        min_distance = 1000000.0 ;

        // Check what the next edge is to be linked
        VerifyNextUnlabeledEdge(x-1, y-1) ;   // neighbor 1
        VerifyNextUnlabeledEdge(x, y-1) ;     // neighbor 2
        VerifyNextUnlabeledEdge(x+1,y-1);     // neighbor 3
        VerifyNextUnlabeledEdge(x-1,y) ;      // neighbor 4
        VerifyNextUnlabeledEdge(x+1,y) ;      // neighbor 5
        VerifyNextUnlabeledEdge(x-1, y+1) ;   // neighbor 6
        VerifyNextUnlabeledEdge(x, y+1) ;     // neighbor 7
        VerifyNextUnlabeledEdge(x+1, y+1) ; // neighbor 8

         if (min_distance != 1000000.0)
               LinkEdgeAtCurrentLocation(newx, newy, n) ;
}
```

(NOTE: This is called Dynamic Programming)

What are the possible outputs from the process of edge linking?

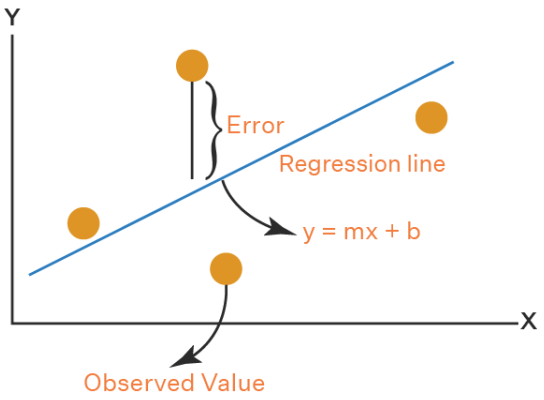
- Linked Edges from Line Segments
- Linked Edges from Curve Segments
- Linked Edges from Circles
- Linked Edges from Ellipses



- Linked Edges from Closed Loops (e.g. triangles, rectangles, polygons, etc)

Estimation of Equation of Line

- Input:
  - A list of points as training set, which belong to a line.
    $List = \{(x_i, y_i), i = 0,1,2, \dots, n\}$
- Output:
  - The coefficients of the equation of line
    $ax + by + c = 1$
- Solution: Least-Squares Method



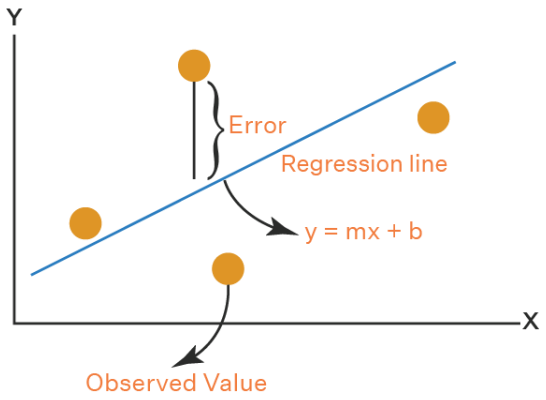
Solution

$$ax_i + by_i + c = 1, i = 0,1,2,3, \dots n$$

$$A = \begin{bmatrix} x_0 & y_0 & 1 \\ x_1 & y_1 & 1 \\ \dots & \dots & \dots \\ x_n & y_n & 1 \end{bmatrix} \qquad X = \begin{bmatrix} a \\ b \\ c \end{bmatrix} \qquad B = \begin{bmatrix} 1 \\ 1 \\ \dots \\ 1 \end{bmatrix}$$

$$AX = B$$

$$X = (A^t A)^{-1}(A^t B)$$



Example

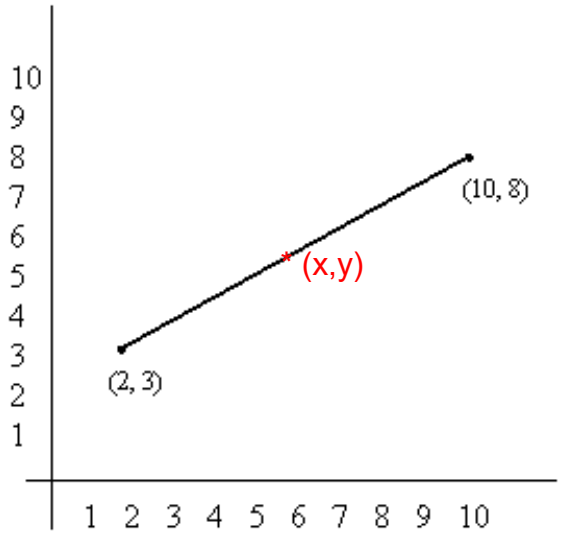
- A line in a 2D space passes through two points: (2,3) and (10,8). What are the parameters of the line?

- Answer:

$$\frac{y-3}{x-2}=\frac{8-3}{10-2}$$

$$y=\frac{5}{8}(x-2)+3$$

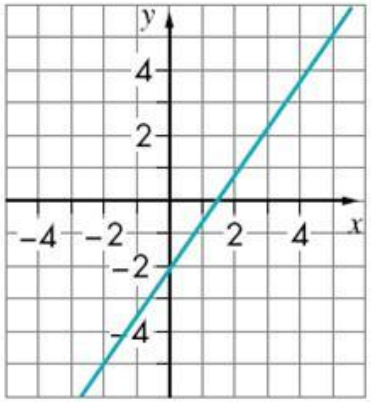
$$y=\frac{5}{8}x+\frac{7}{4} \quad \Rightarrow \quad (a,b)=(\frac{5}{8},\frac{7}{4})$$



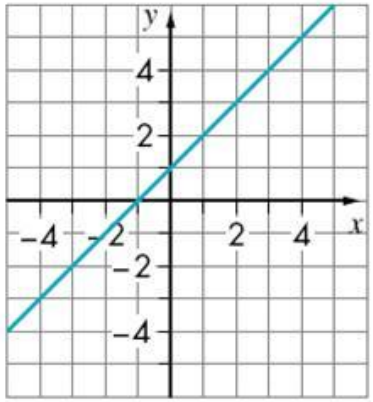
Exercises

Find the equations of these straight lines.

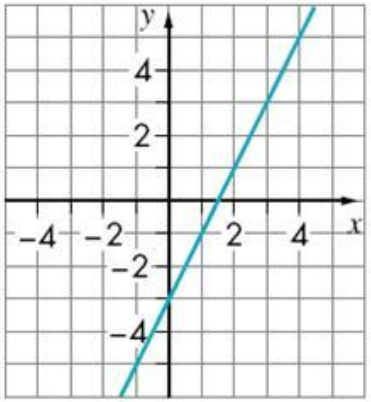
a



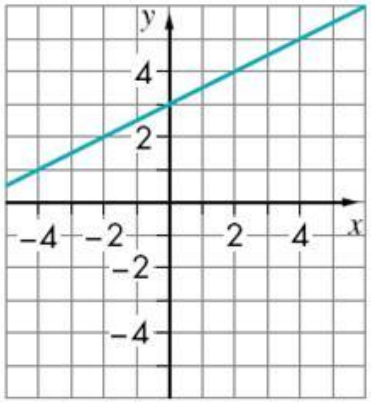
b



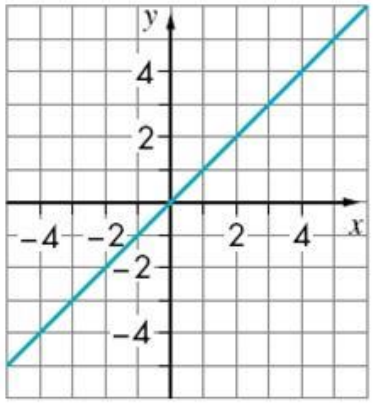
c



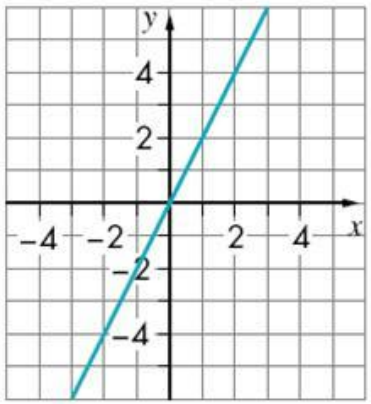
d



e



f



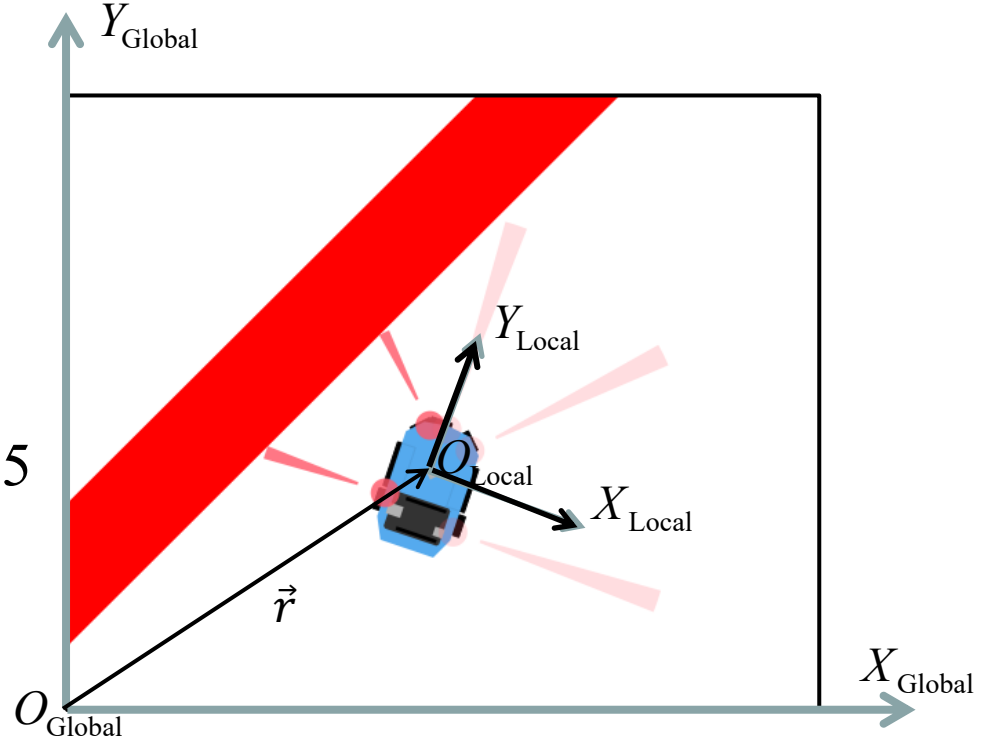
Example

- A mobile robot moves on a floor by following a straight line. It passes point A(2.0, 2.0) (cm) at time t1 = 1.0 s, and point B(4.0, 7.0) (cm) at t2 = 3.0 s. What are the time functions representing the coordinates of the origin of the local coordinate system assigned to the mobile robot?
- Answer:

$$a_x \times 1.0 + b_x = 2.0 \qquad a_x \times 3.0 + b_x = 4.0 \Rightarrow a_x = 1.0,\ b_x = 1.0$$

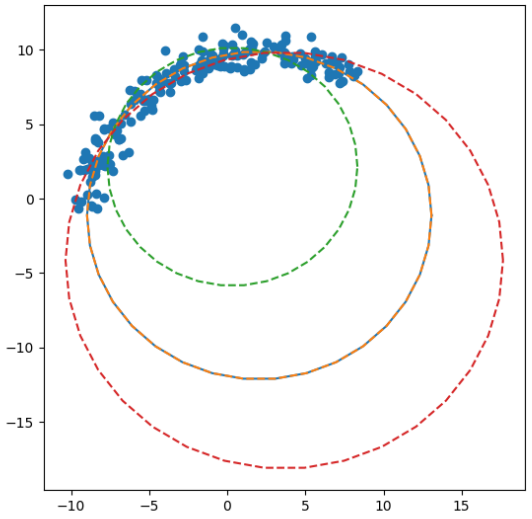
$$a_y \times 1.0 + b_y = 2.0 \qquad a_y \times 3.0 + b_y = 7.0 \Rightarrow a_y = 2.5,\ b_y = -0.5$$

$$\vec{r}(t) = \begin{pmatrix} x(t) \\ y(t) \end{pmatrix} = \begin{pmatrix} t + 1.0 \\ 2.5t - 0.5 \end{pmatrix}$$



Estimation of Equation of Circle

- Input:
  - A list of points as training set, which belong to a circle.
    $List = \{(x_i, y_i), i = 0,1,2, \dots, n\}$
- Output:
  - The coefficients of the equation of circle
    $(x - c_x)^2 + (y - c_y)^2 = r^2$
- Solution: Least-Squares Method



Solution

$$(x_i - c_x)^2 + (y_i - c_y)^2 = r_i^2, i = 0,1,2,3, \dots n$$

$$x_i^2 - 2x_i c_x + c_x^2 + y_i^2 - 2y_i c_y + c_y^2 = r_i^2, i = 0,1,2,3, \dots n$$

$$x_0^2 - 2x_0 c_x + c_x^2 + y_0^2 - 2y_0 c_y + c_y^2 = r_0^2$$

$$x_1^2 - 2x_1 c_x + c_x^2 + y_1^2 - 2y_1 c_y + c_y^2 = r_1^2$$

……

$$x_i^2 - 2x_i c_x + c_x^2 + y_i^2 - 2y_i c_y + c_y^2 = r_i^2$$

……

$$x_n^2 - 2x_n c_x + c_x^2 + y_n^2 - 2y_n c_y + c_y^2 = r_n^2$$

Solution (continued)

$$2(x_0 - x_1)c_x + 2(y_0 - y_1)c_y = x_0^2 - x_1^2 + y_0^2 - y_1^2$$

……

$$2(x_0 - x_i)c_x + 2(y_0 - y_i)c_y = x_0^2 - x_i^2 + y_0^2 - y_i^2$$

……

$$2(x_0 - x_n)c_x + 2(y_0 - y_n)c_y = x_0^2 - x_n^2 + y_0^2 - y_n^2$$

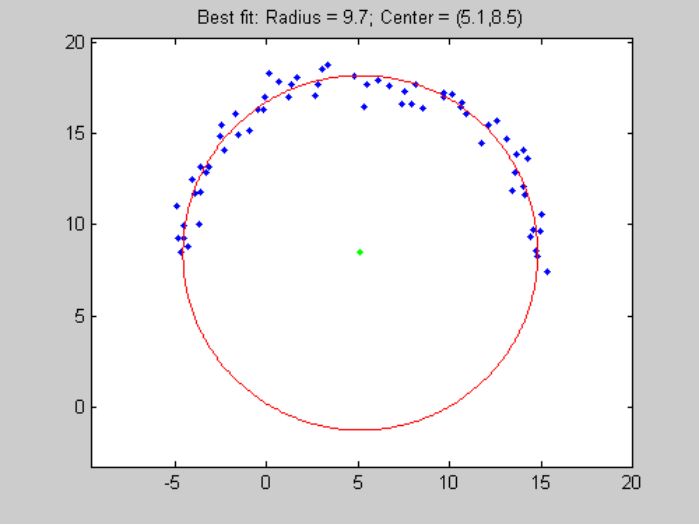
$$A = \begin{bmatrix} 2(x_0 - x_1) & 2(y_0 - y_1) \\ \dots & \dots \\ 2(x_0 - x_i) & 2(y_0 - y_i) \\ \dots & \dots \\ 2(x_0 - n) & 2(y_0 - y_n) \end{bmatrix} \quad X = \begin{bmatrix} c_x \\ c_y \end{bmatrix} \quad B = \begin{bmatrix} x_0^2 - x_1^2 + y_0^2 - y_1^2 \\ \dots \\ x_0^2 - x_i^2 + y_0^2 - y_i^2 \\ \dots \\ x_0^2 - x_n^2 + y_0^2 - y_n^2 \end{bmatrix}$$

Solution (continued)

$$A = \begin{bmatrix} 2(x_0 - x_1) & 2(y_0 - y_1) \\ \dots & \dots \\ 2(x_0 - x_i) & 2(y_0 - y_i) \\ \dots & \dots \\ 2(x_0 - n) & 2(y_0 - y_n) \end{bmatrix} \qquad X = \begin{bmatrix} c_x \\ c_y \end{bmatrix} \qquad B = \begin{bmatrix} x_0^2 - x_1^2 + y_0^2 - y_1^2 \\ \dots \\ x_0^2 - x_i^2 + y_0^2 - y_i^2 \\ \dots \\ x_0^2 - x_n^2 + y_0^2 - y_n^2 \end{bmatrix}$$

$$AX = B$$

$$X = (A^t A)^{-1}(A^t B)$$

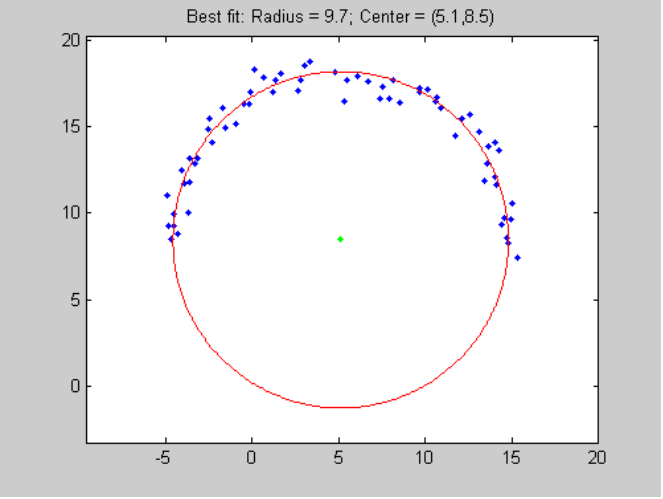


Solution (continued)

$$(x_i - c_x)^2 + (y_i - c_y)^2 = r_i^2, i = 0,1,2,3, \ldots n$$

$$r_i = \sqrt{(x_i - c_x)^2 + (y_i - c_y)^2}, i = 0,1,2,3, \ldots n$$

$$r = \frac{\sum_{i=0}^{n} r_i}{n}$$

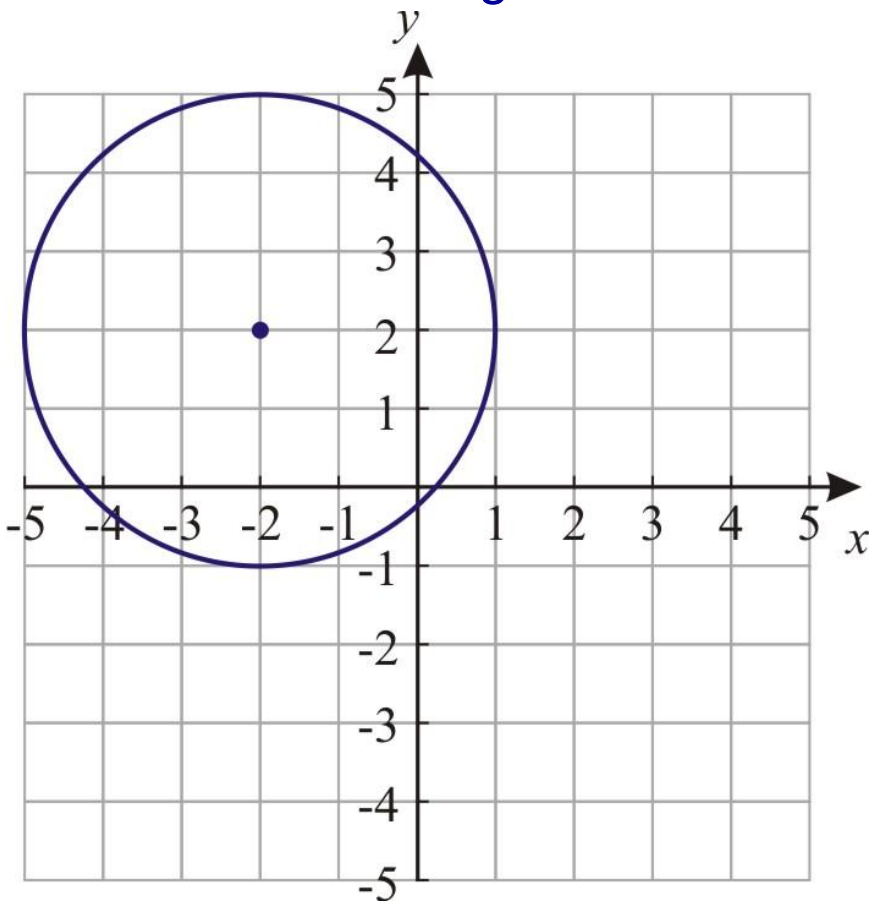


Example

- What are the parameters of the circle shown in the figure?

- Answer:

$$\begin{pmatrix} x_c \\ y_c \\ r \end{pmatrix} = \begin{pmatrix} -2 \\ 2 \\ 3 \end{pmatrix}$$



Example

- A mechanical gear is under visual inspection. If the tips of the gear's three teeth are at the positions (x1, y1), (x2, y2) and (x3, y3), what are the parameters of the circle which envelopes the tips of all the teeth of the gear?

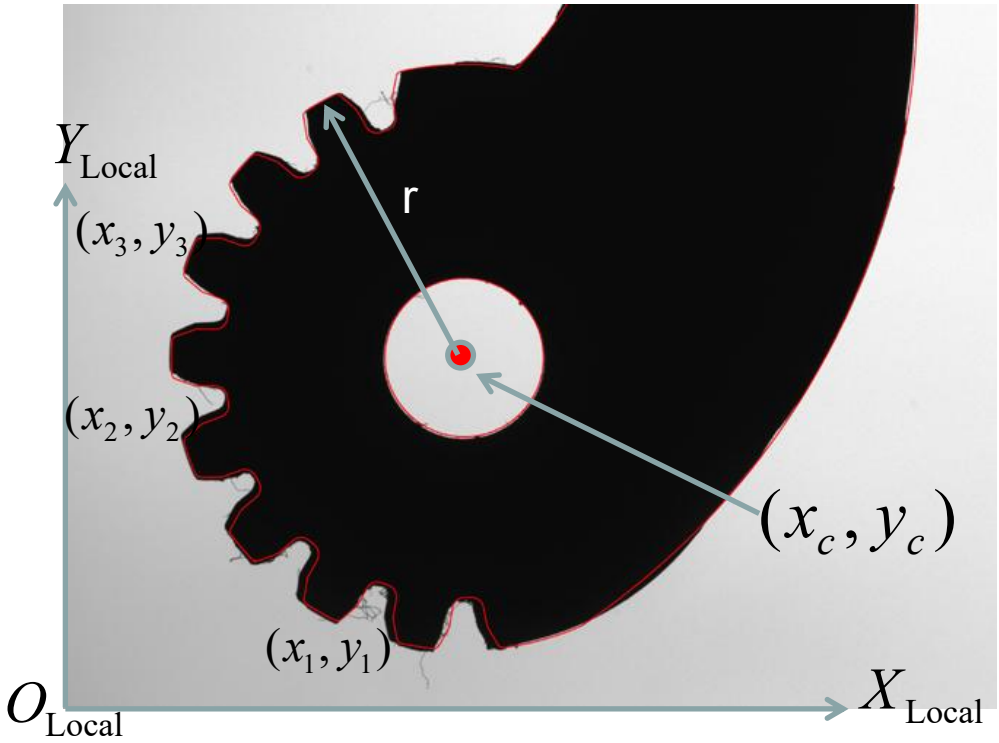
- Answer:

$$(x_1 - x_c)^2 + (y_1 - y_c)^2 = r^2$$

$$(x_2 - x_c)^2 + (y_2 - y_c)^2 = r^2$$

$$(x_3 - x_c)^2 + (y_3 - y_c)^2 = r^2$$

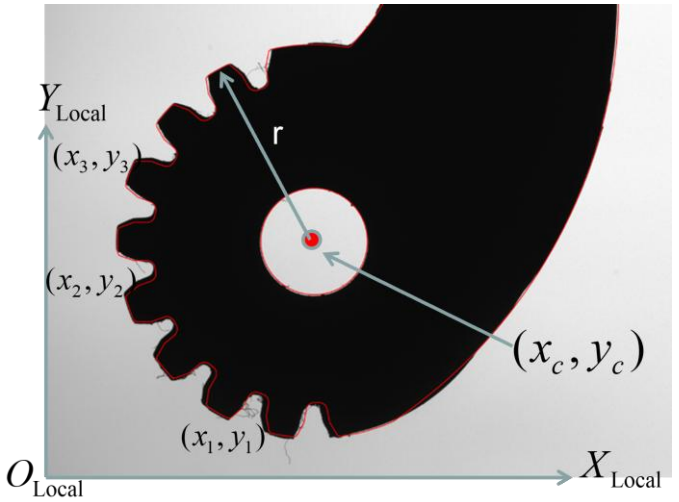
(to continue)



$$x_1^2 - 2x_1x_c + x_c^2 + y_1^2 - 2y_1y_c + y_c^2 = r^2$$

$$x_2^2 - 2x_2x_c + x_c^2 + y_2^2 - 2y_2y_c + y_c^2 = r^2$$

$$x_3^2 - 2x_3x_c + x_c^2 + y_3^2 - 2y_3y_c + y_c^2 = r^2$$



$$2(x_1 - x_2)x_c + 2(y_1 - y_2)y_c = x_1^2 - x_2^2 + y_1^2 - y_2^2$$

$$2(x_1 - x_3)x_c + 2(y_1 - y_3)y_c = x_1^2 - x_3^2 + y_1^2 - y_3^2$$

$$\begin{bmatrix} 2(x_1 - x_2) & 2(y_1 - y_2) \\ 2(x_1 - x_3) & 2(y_1 - y_3) \end{bmatrix} \bullet \begin{bmatrix} x_c \\ y_c \end{bmatrix} = \begin{bmatrix} x_1^2 - x_2^2 + y_1^2 - y_2^2 \\ x_1^2 - x_3^2 + y_1^2 - y_3^2 \end{bmatrix}$$

Circle's Center:

$$\begin{bmatrix} x_c \\ y_c \end{bmatrix} = \begin{bmatrix} 2(x_1 - x_2) & 2(y_1 - y_2) \\ 2(x_1 - x_3) & 2(y_1 - y_3) \end{bmatrix}^{-1} \bullet \begin{bmatrix} x_1^2 - x_2^2 + y_1^2 - y_2^2 \\ x_1^2 - x_3^2 + y_1^2 - y_3^2 \end{bmatrix}$$

$$\begin{bmatrix} x_c \\ y_c \end{bmatrix} = \begin{bmatrix} 2(x_1 - x_2) & 2(y_1 - y_2) \\ 2(x_1 - x_3) & 2(y_1 - y_3) \end{bmatrix}^{-1} \bullet \begin{bmatrix} x_1^2 - x_2^2 + y_1^2 - y_2^2 \\ x_1^2 - x_3^2 + y_1^2 - y_3^2 \end{bmatrix}$$

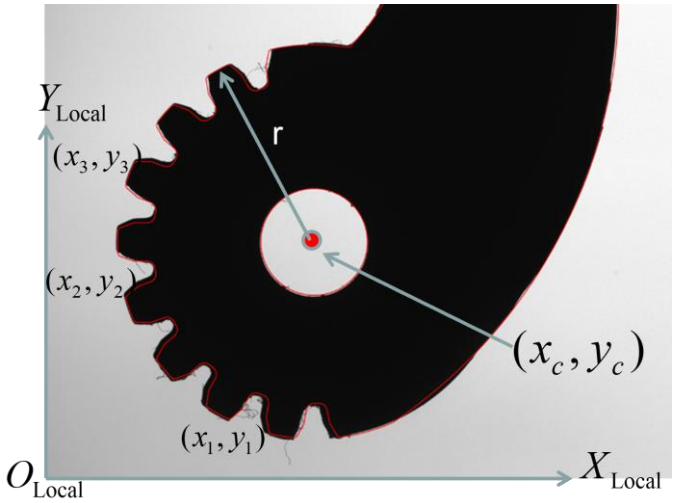
$$r_1 = \sqrt{(x_1 - x_c)^2 + (y_1 - y_c)^2}$$

$$r_2 = \sqrt{(x_2 - x_c)^2 + (y_2 - y_c)^2}$$

$$r_3 = \sqrt{(x_3 - x_c)^2 + (y_3 - y_c)^2}$$

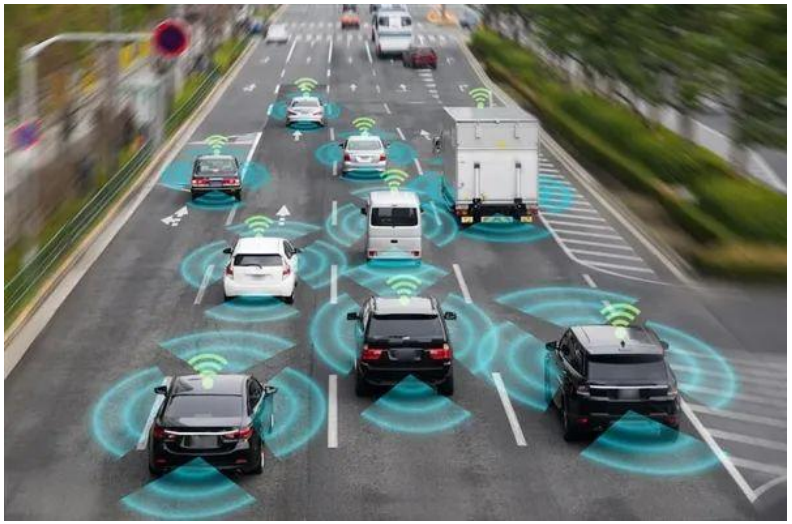
Circle's Radius:

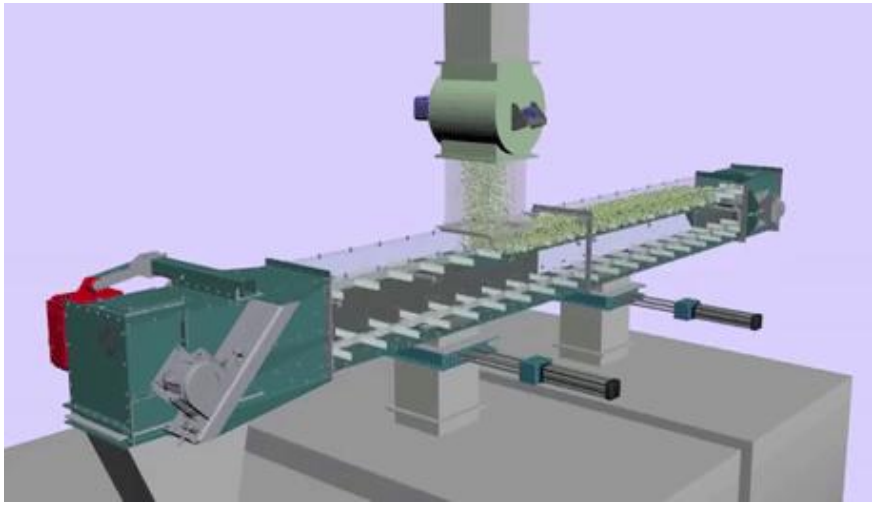
$$r = \frac{r_1 + r_2 + r_3}{3}$$



Outline of Lecture 1

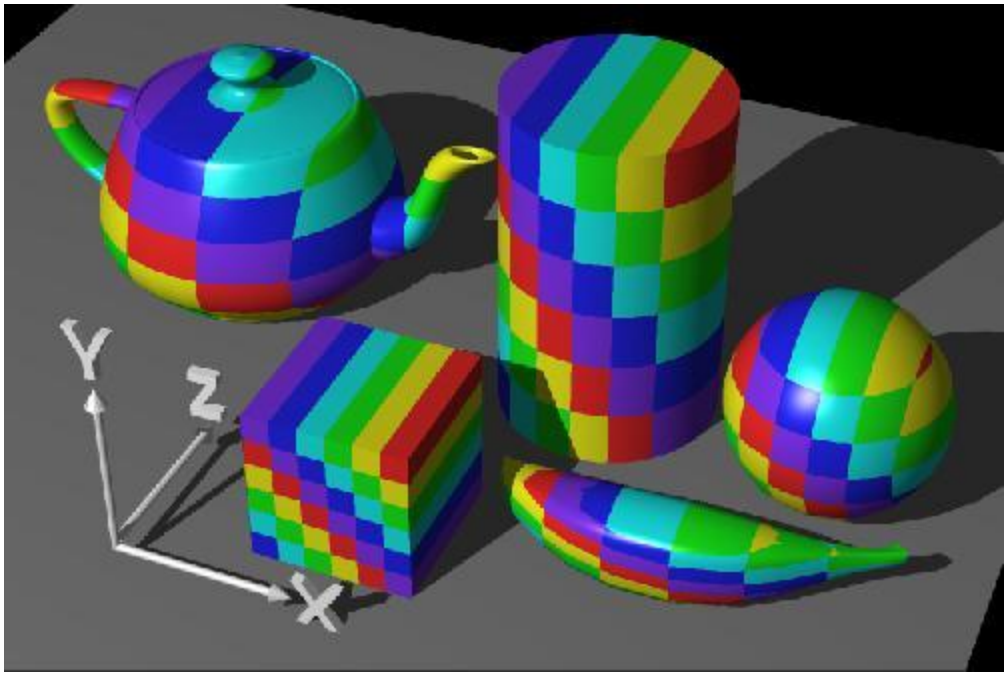
- Concept of Knowledge Space
- Representation of 2D Geometry
- Computation of 2D Geometry
- Representation of 3D Geometry
- Computation of 3D Geometry





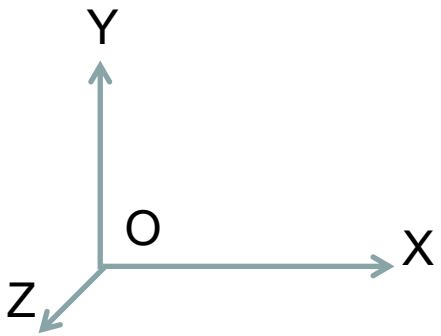
Understanding 3D Geometry (1)

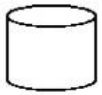
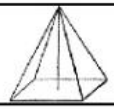
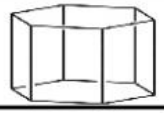
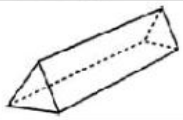
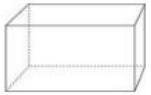
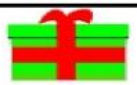
- 3D geometry refers to the appearance of physical entities in a three-dimensional space.
- 3D space consists of a set of positions which are fully determined with three coordinates.



Understanding 3D Geometry (2)

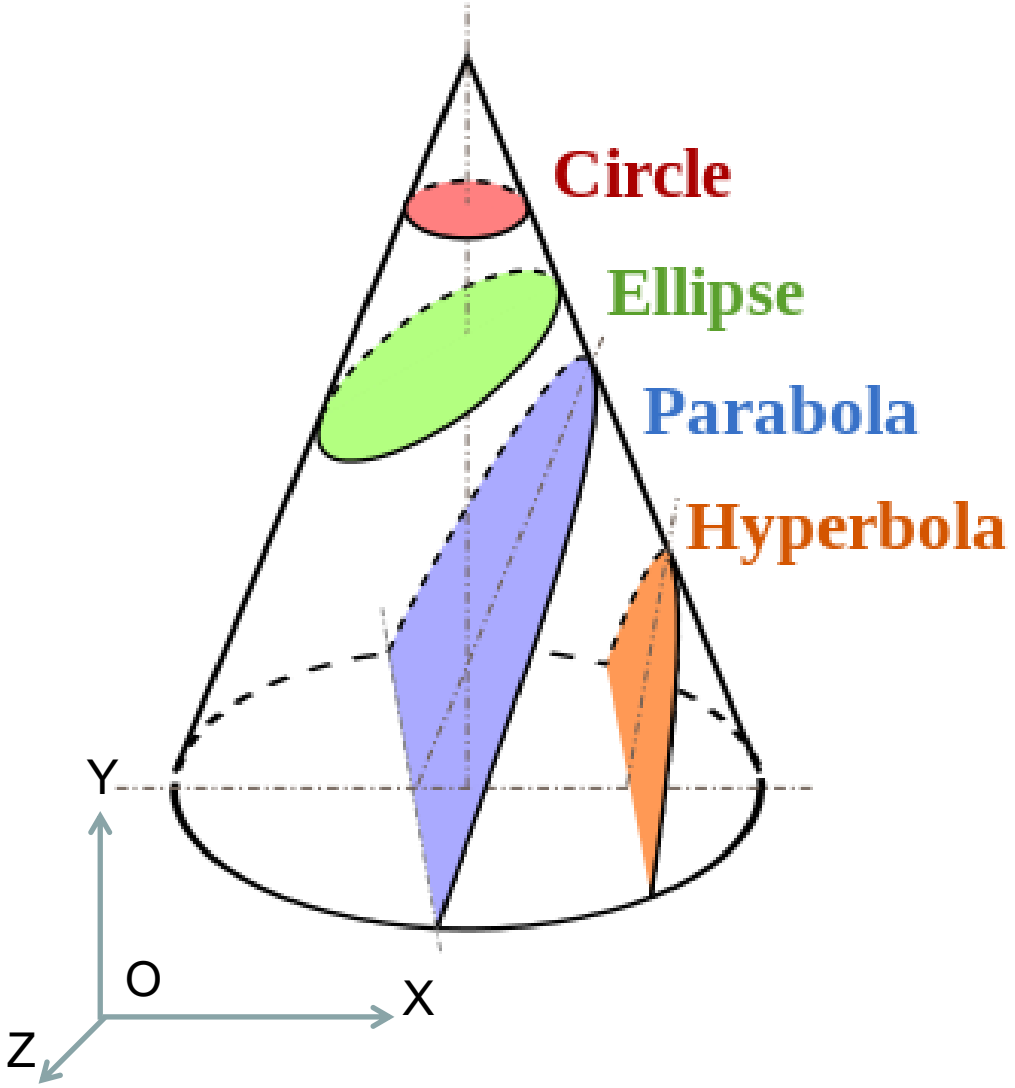
- The appearance of physical entities in a 3D space is manifested in the form of 3D shapes which are also called objects.



3-D Shape	Picture	Example
cube		
cylinder		
sphere		
pyramid		
hexagonal prism		
triangular prism		
cone		
rectangular prism		

Example of Shapes in 3D Space

- Intersections between 3D shape and planes will result in 2D shapes.
- 3D shape has outer surface, inner surface and body between them.
- 2D shape has outer edge, inner edge and body between them.



Understanding 3D Geometry (3)

(Why to use the terminology of generative AI? What is not generative?)

- Complex shapes in a 3D space are the results of compositional rules such as:

Intersection at Points

  - Connect Between Points(point of shape 1, point of shape 2) at Angle(angle between shape 1 and shape 2)

Intersection at Curves

  - Connect Between Curves(curve of shape 1, curve of shape 2) with Offset(distance between the endpoints of two curves)

Intersection at Surfaces

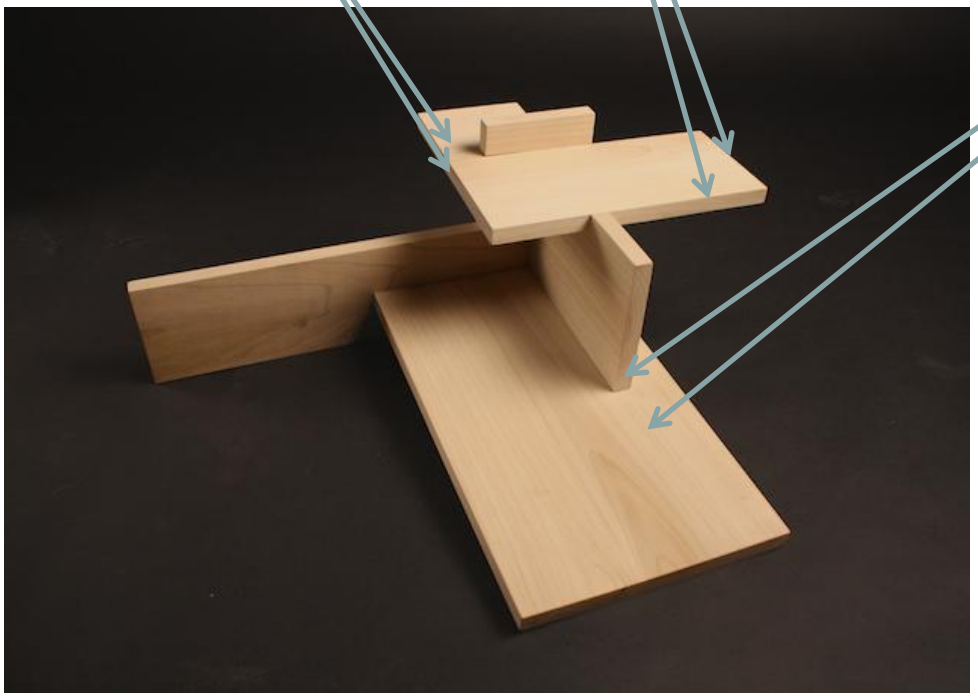
  - Connect Between Surfaces(surface of shape 1, surface of shape 2) with Orientation(angle about normal to surface) and Offset(distance between the origins of two surfaces)

Example

Edge Lines
Connected Between Points

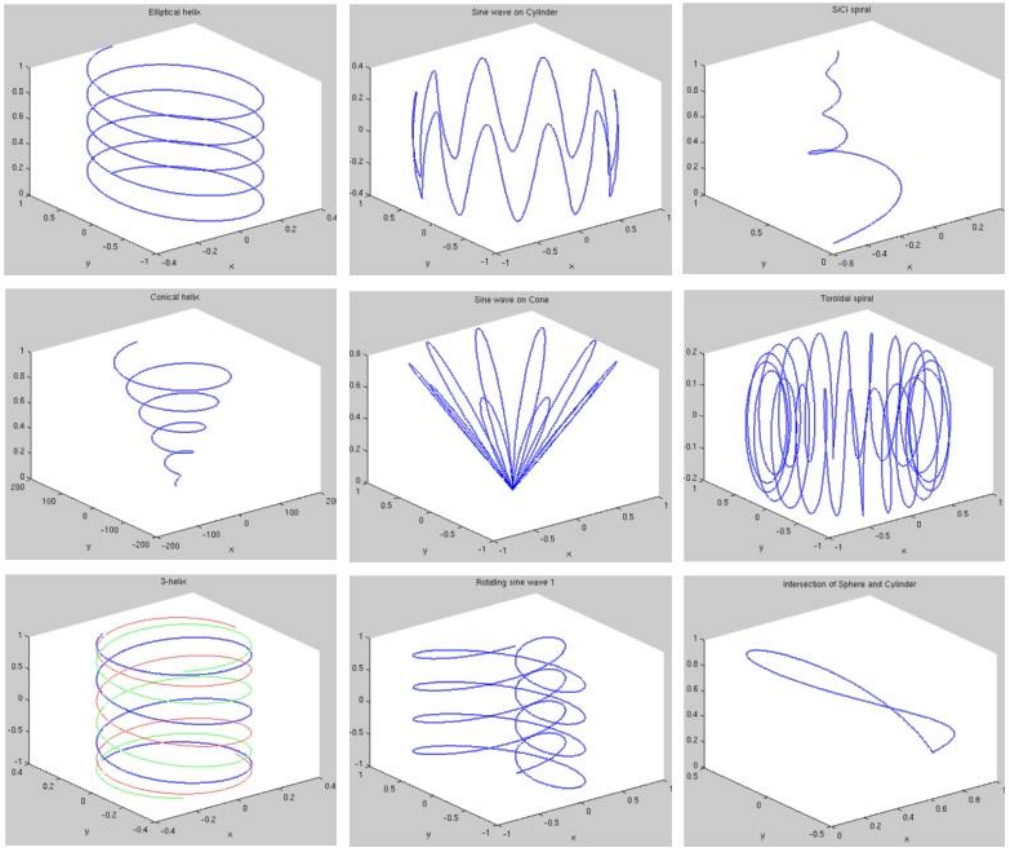
Surfaces
Connected Between Lines

Blocks
Connected Between Surfaces



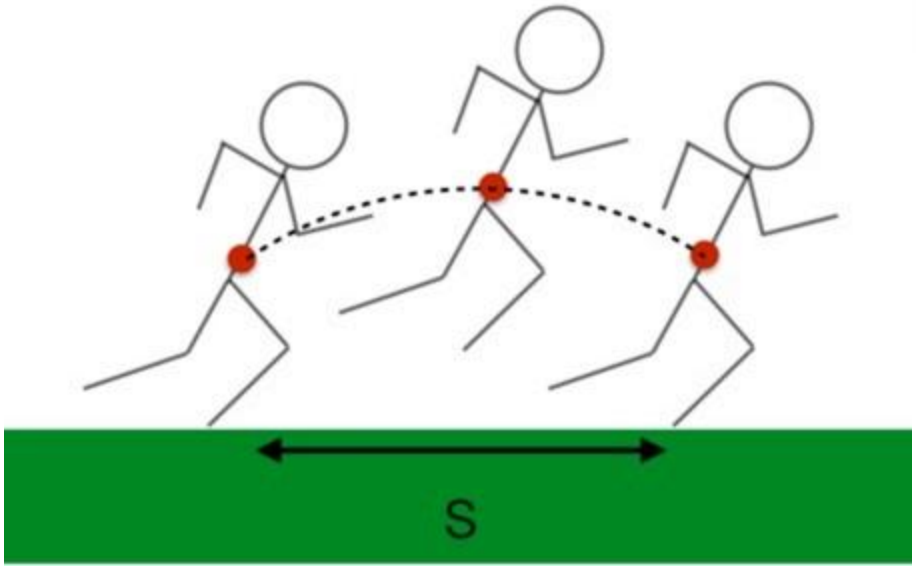
Understanding 3D Geometry (4)

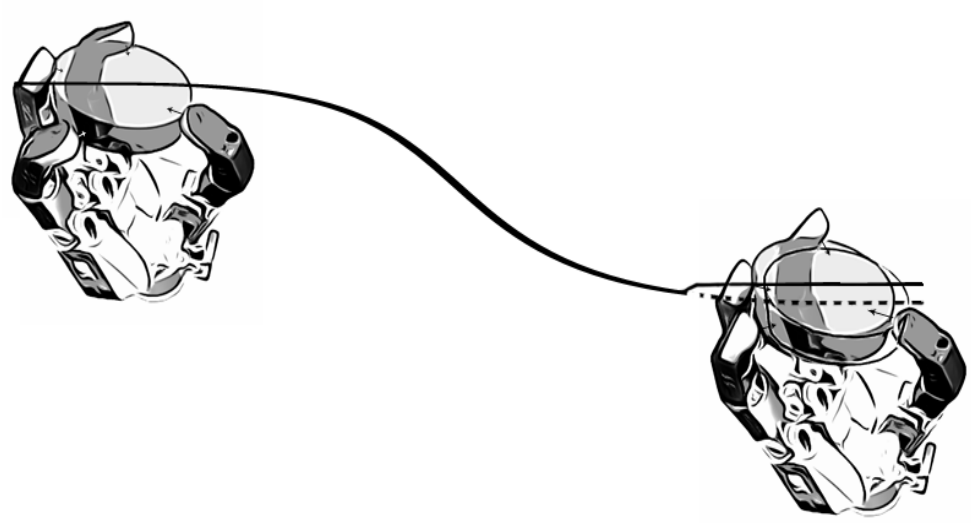
- The geometry of physical entities in a 3D space also includes their positions, orientations and travelled locations (i.e. motions).



Understanding 3D Geometry (5)

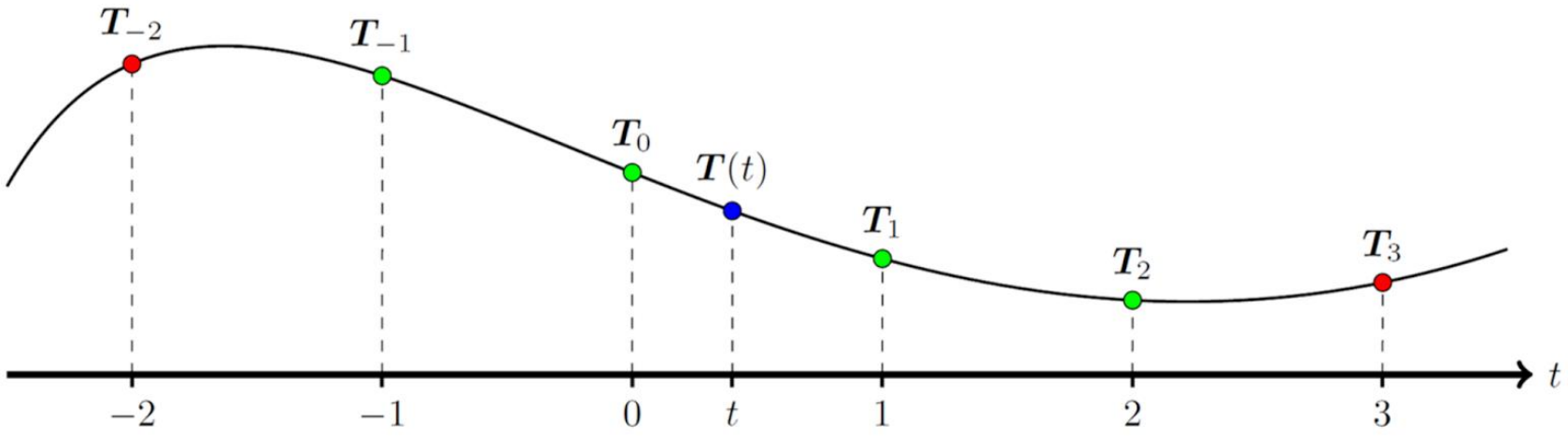
- The spatial locations travelled or to be travelled are called paths.



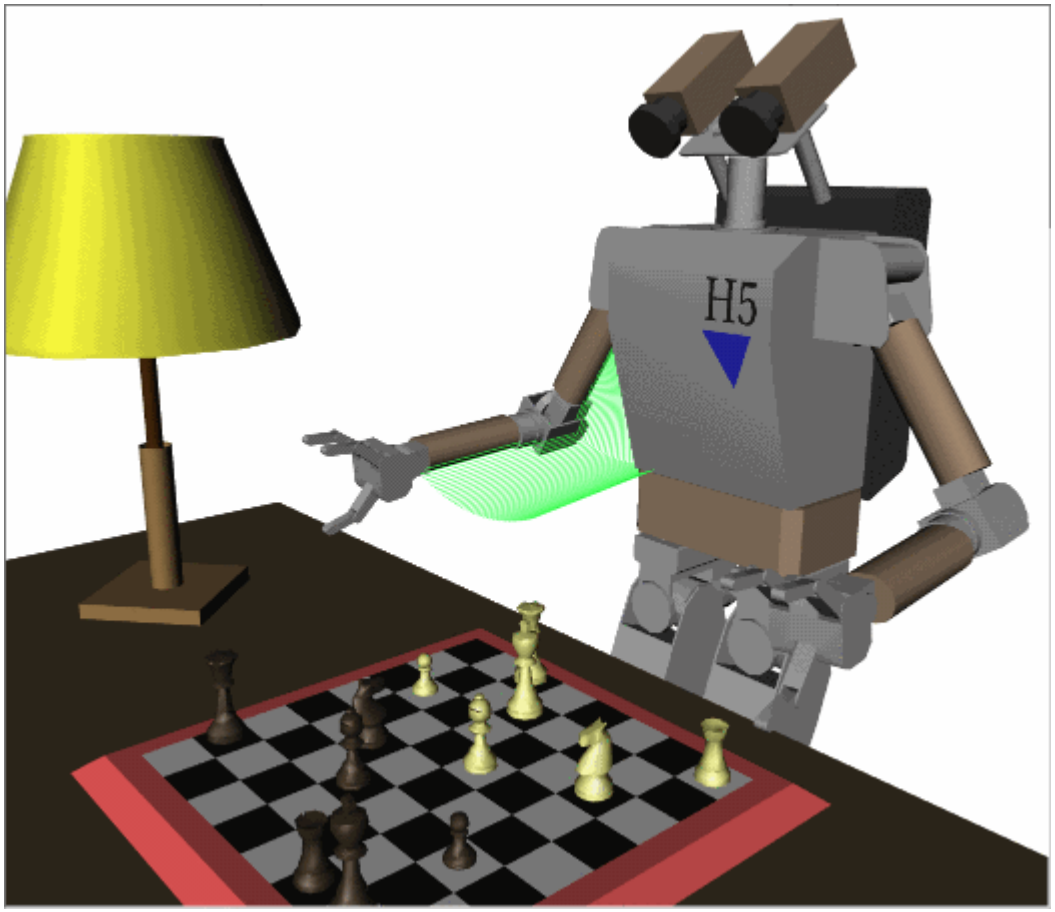


Understanding 3D Geometry (6)

- The spatial locations with time constraint are called trajectories.

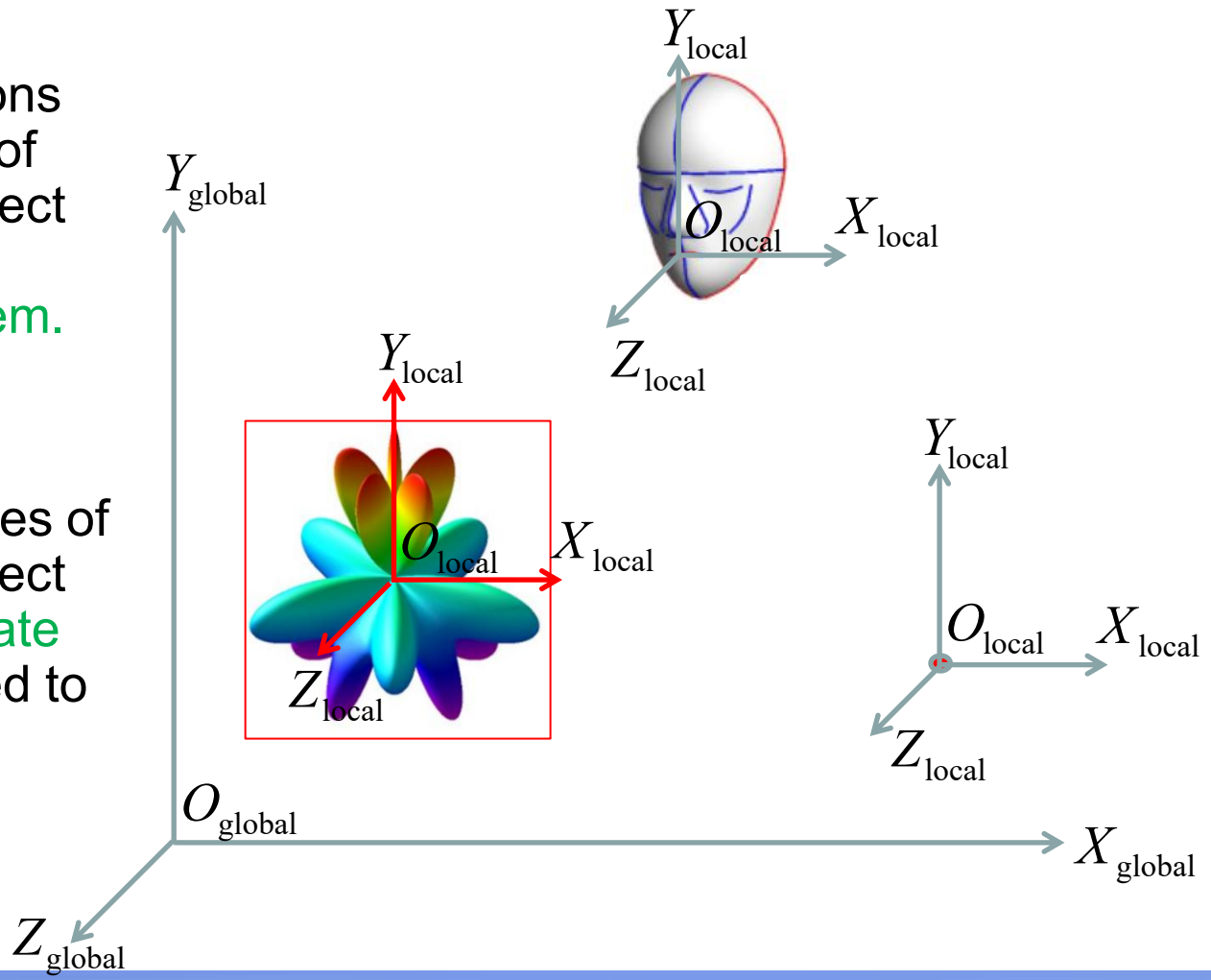


Example of Robot Performing Motion with Time Constraint



Two Types of Parameters in Geometry

- Pose:
  - It refers to positions and orientations of entities with respect to a Global Coordinate System.
- Shape:
  - It refers to surfaces of entities with respect to Local Coordinate Systems assigned to these shapes, respectively.

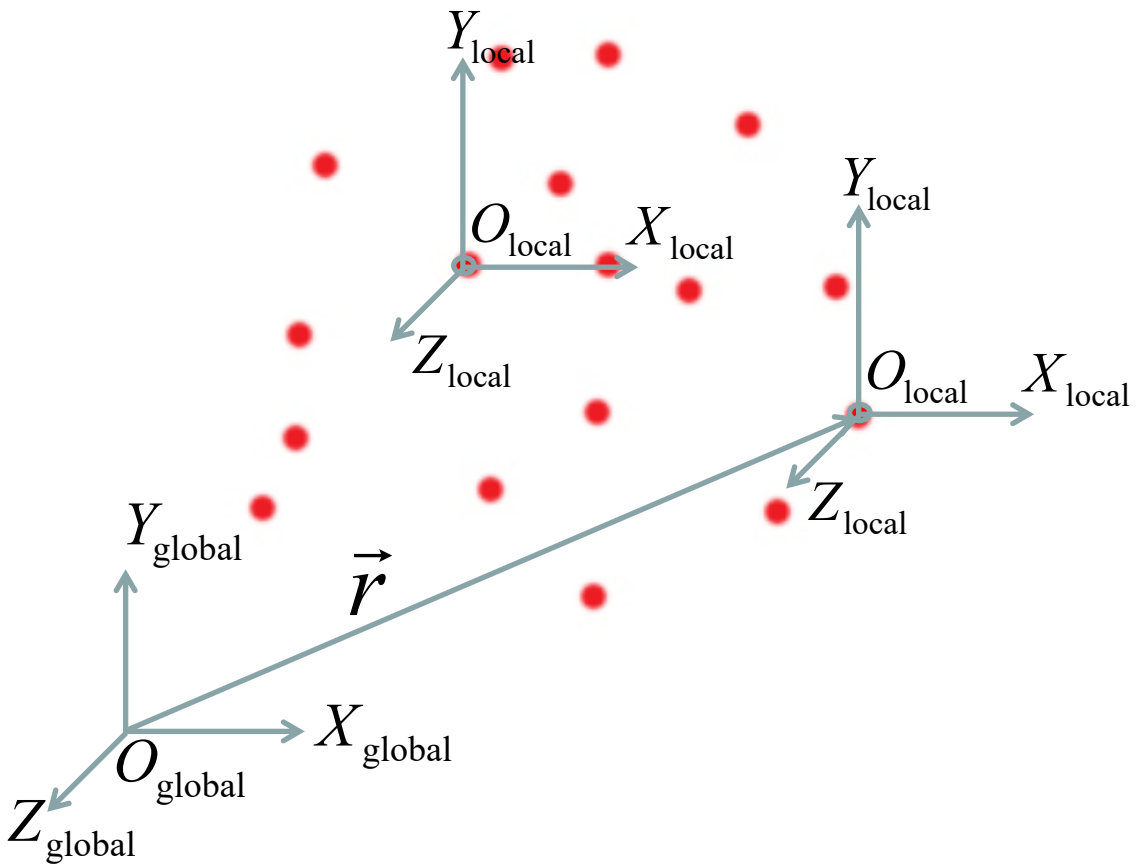


Representation of Positions in 3D Space without Time Constraints

$$\vec{r} = (x, y, z)$$

$$f(x, y) = 0$$

$$f(x, z) = 0$$

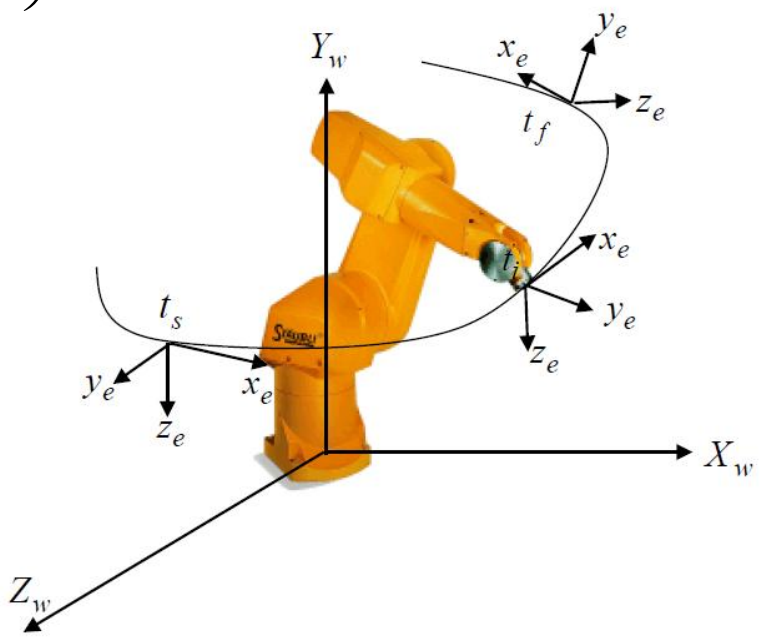


Representation of Positions in 3D Space with Time Constraints

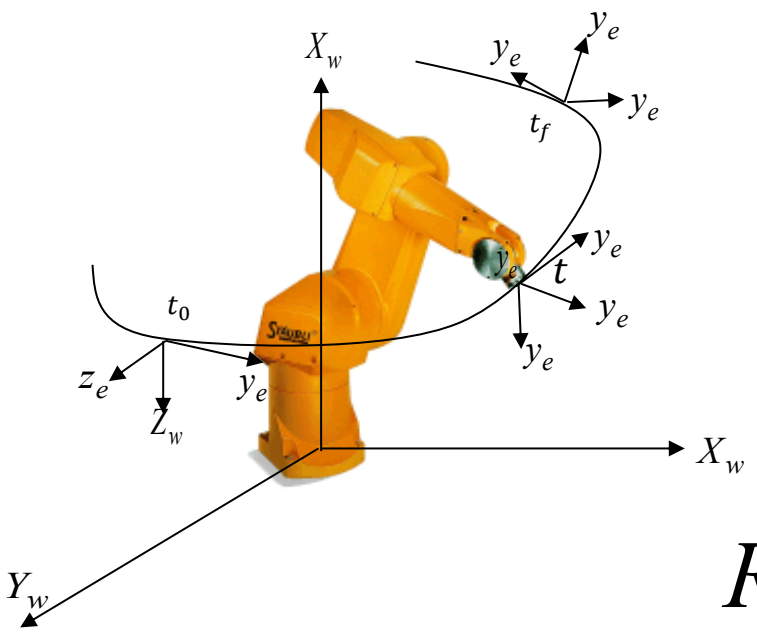
$$\vec{r}(t) = \begin{pmatrix} x(t) \\ y(t) \\ z(t) \end{pmatrix} = \begin{pmatrix} a_x t^3 + b_x t^2 + c_x t + d_x \\ a_y t^3 + b_y t^2 + c_y t + d_y \\ a_z t^3 + b_z t^2 + c_z t + d_z \end{pmatrix}$$

$$\vec{r}(t) = \begin{pmatrix} x(t) \\ y(t) \\ z(t) \end{pmatrix} = \begin{pmatrix} a_x t^2 + b_x t + c_x \\ a_y t^2 + b_y t + c_y \\ a_z t^2 + b_z t + c_z \end{pmatrix}$$

$$\vec{r}(t) = \begin{pmatrix} x(t) \\ y(t) \\ z(t) \end{pmatrix} = \begin{pmatrix} a_x t + b_x \\ a_y t + b_y \\ a_z t + b_z \end{pmatrix}$$



Representation of Orientations in 3D Space



X Axis's Unit Vector

Y Axis's Unit Vector

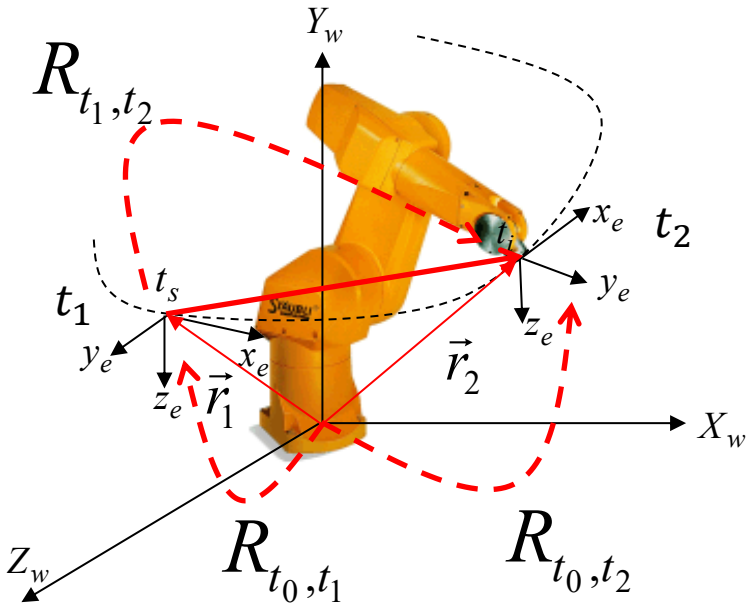
Z Axis's Unit Vector

$$R_{t_0,t} = \begin{bmatrix} x_i(t) & x_j(t) & x_k(t) \\ y_i(t) & y_j(t) & y_k(t) \\ z_i(t) & z_j(t) & z_k(t) \end{bmatrix}$$

This is the orientation at time t, as the result of the change of orientation from time t0 to time t.

Homogeneous Transformation

(Robot Arm)



$$H_{tooltip} = \begin{bmatrix} R_{tooltip} & T_{tooltip} \\ 0 \quad 0 \quad 0 & 1 \end{bmatrix}$$

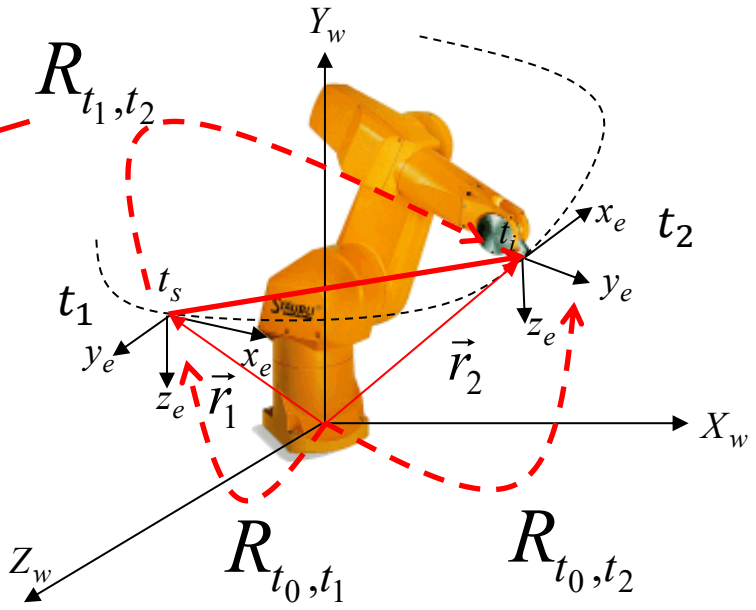
$$H_{world} = \begin{bmatrix} R_{tooltip}^{-1} & -R_{tooltip}^{-1} \times T_{tooltip} \\ 0 \quad 0 \quad 0 & 1 \end{bmatrix}$$

$$H_{tooltip} \times H_{world} = I_{4\times4}$$

$$H_{tooltip} = H_{world}^{-1}$$

Equivalent Axis and Angle of Rotation in 3D Space

$$R_{t_1,t_2} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$



This is the orientation at time t2, as the result of the change of orientation from time t1 to time t2.

Equivalent Axis of Rotation:

$$\vec{r}\Big|_{t_1,t_2} = \begin{bmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{bmatrix}$$

Equivalent Angle of Rotation:

$$\theta_{t_1,t_2} = \arccos\left(\frac{r_{11} + r_{22} + r_{33} - 1}{2}\right)$$

Simple and Useful for Adding Time Constraint

Applications with Equivalent Axis and Angle of Rotation: Animation / Planning

- Euler Equation:

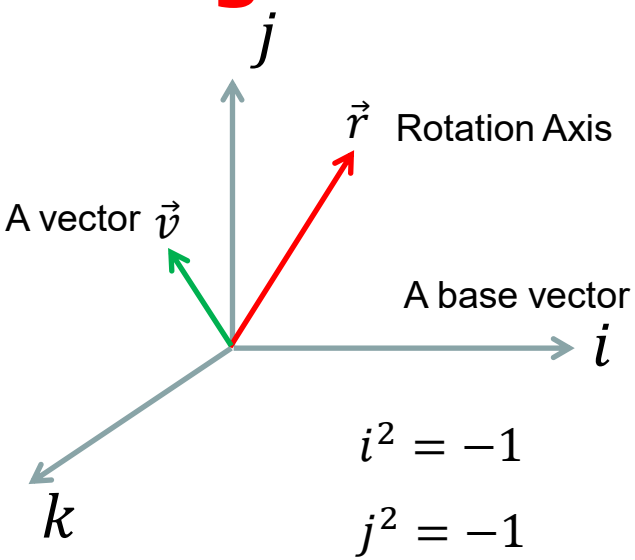
$$e^{j\omega} = \cos(\omega) + j\,sin(\omega)$$

- Unit Axis of Rotation:

$$\vec{r} = r_x i + r_y j + r_z k$$

- Angle of Rotation: θ

- Unit Quaternion: $q = e^{\theta\vec{r}} = \cos(\theta) + \sin(\theta)\vec{r}$



$i^2 = -1$

$j^2 = -1$

$k^2 = -1$

$ijk = -1$

- Rotation of Vector:

$$\vec{v}_{after} = q \cdot \vec{v}_{before} = (\cos(\theta) + \sin(\theta)\vec{r}) \cdot \vec{v}_{before}$$

More About Quaternion

- Non-Unit Quaternion: $q = (a, \vec{v}) = a + b\vec{i} + c\vec{j} + d\vec{k}$

- Unit Quaternion: $q = (\cos(\theta), \sin(\theta)\vec{r}) = \cos(\theta) + \sin(\theta)\vec{r}$

  Unit vector of axis of rotation

- Sum of Two Quaternions: $q_1 = (a_1, \vec{v}_1) \quad q_2 = (a_2, \vec{v}_2)$

  $$q_1 + q_2 = (a_1 + a_2, \vec{v}_1 + \vec{v}_2)$$

- Multiplication of Two Quaternions: $q_1 = (a_1, \vec{v}_1) \quad q_2 = (a_2, \vec{v}_2)$

  $$q_1 \cdot q_2 = (a_1 a_2 - \vec{v}_1 \cdot \vec{v}_2, \quad a_1\vec{v}_2 + a_2\vec{v}_1 + \vec{v}_1 \times \vec{v}_2)$$

- Equation of Rotating a Vector: $v = (0, \vec{v}) \quad q = (\cos(\theta), \sin(\theta)\vec{r})$

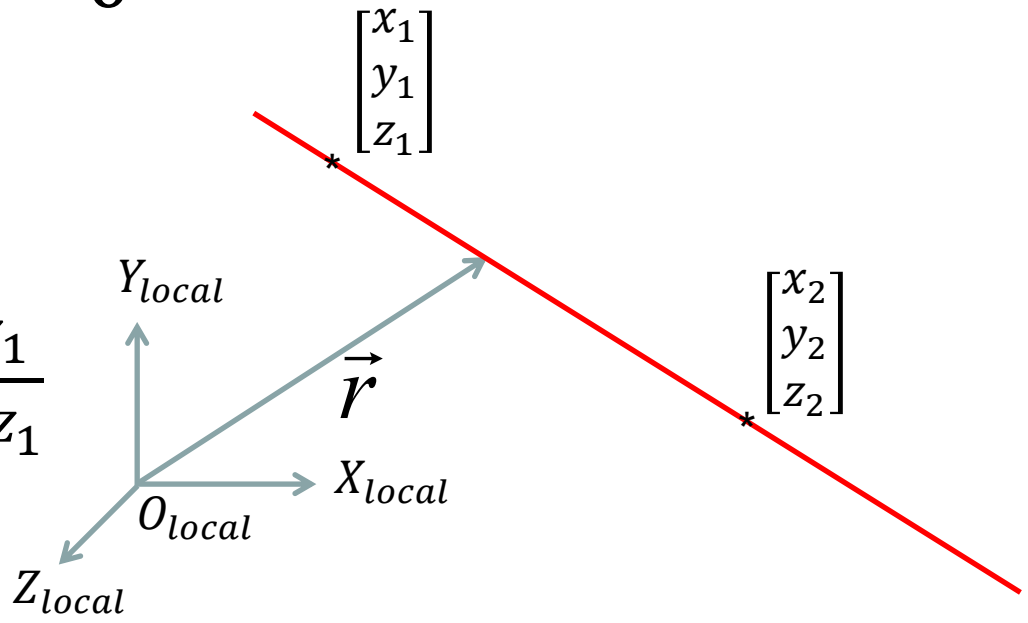
  $$v_{new} = q \cdot v$$

Representation of Line-Type Shape in 3D Space

$$\begin{cases} ax + by + cz + d = 0 \\ ex + fy + gz + h = 0 \end{cases}$$

Or:

$$\frac{x - x_1}{x_2 - x_1} = \frac{y - y_1}{y_2 - y_1} = \frac{z - z_1}{z_2 - z_1}$$



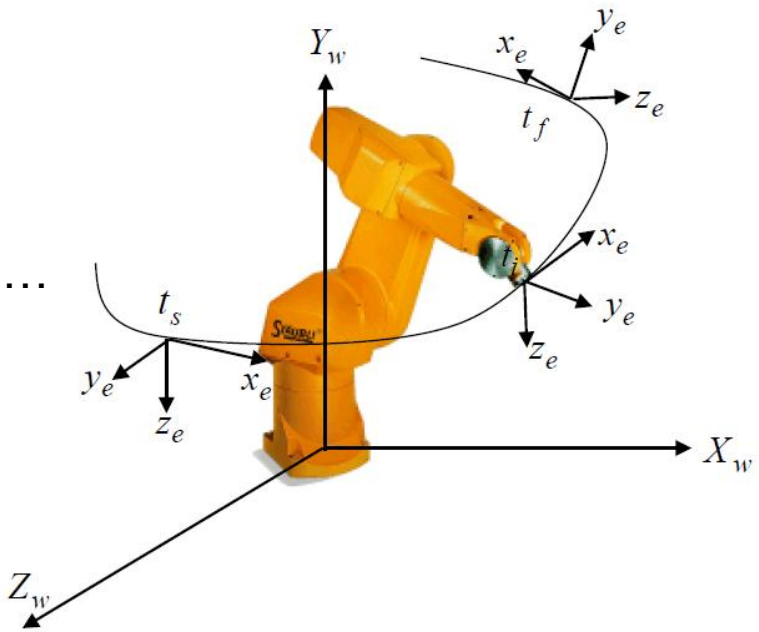
Representation of Curve-Type Shape in 3D Space

$$f(x,z)=0 \quad \Rightarrow \quad \begin{bmatrix} x^n & x^{n-1} & \dots & 1 \end{bmatrix} A_{n\times n} \begin{bmatrix} z^n \\ z^{n-1} \\ \dots \\ 1 \end{bmatrix} = 0$$

n=1,2,…

$$f(y,z)=0 \quad \Rightarrow \quad \begin{bmatrix} y^n & y^{n-1} & \dots & 1 \end{bmatrix} B_{n\times n} \begin{bmatrix} z^n \\ z^{n-1} \\ \dots \\ 1 \end{bmatrix} = 0$$

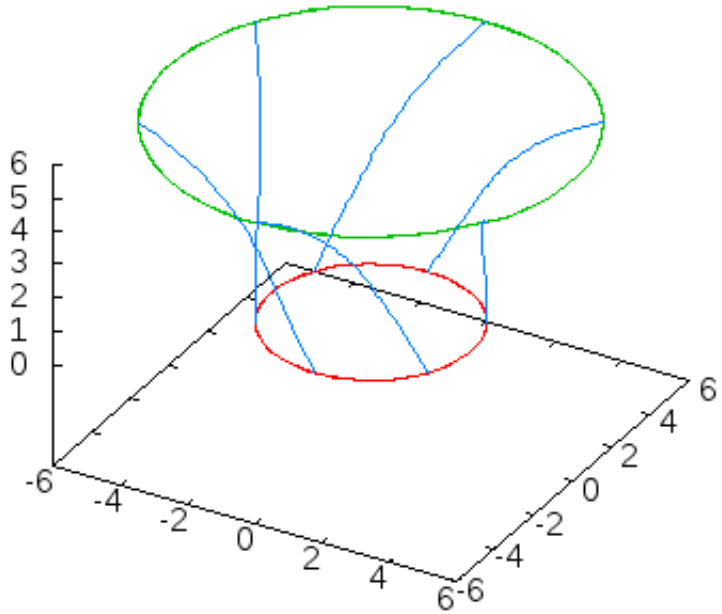
n=1,2,…



Representation of Spherical Shape in 3D Space

$$(x - c_x)^2 + (y - c_y)^2 + (z - c_z)^2 = r^2$$

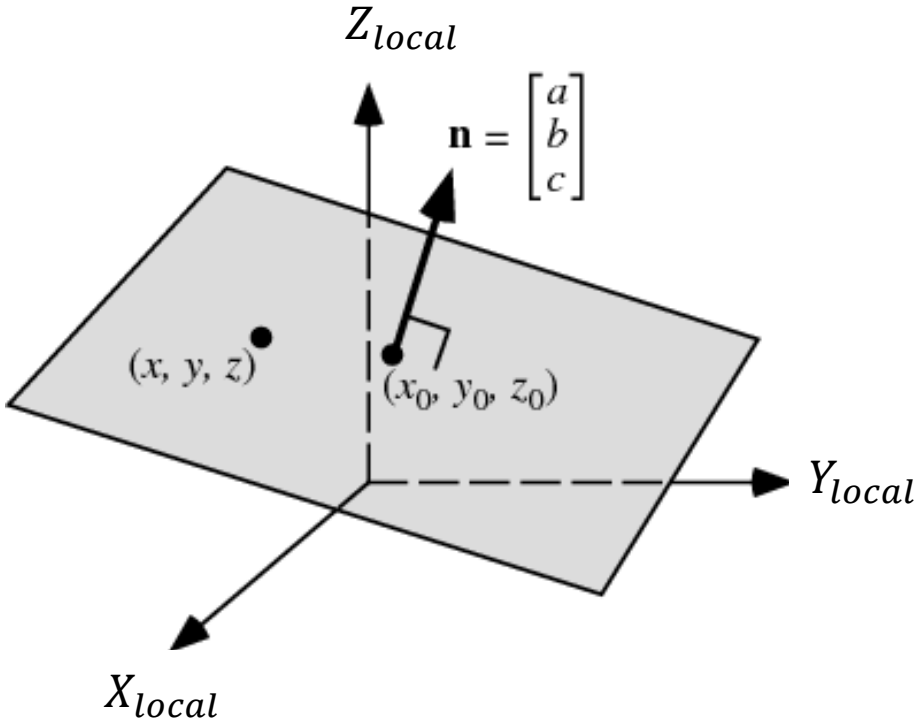
A 3D circle is the intersection between a sphere and a plane.



$$ax + by + cz + d = 0$$

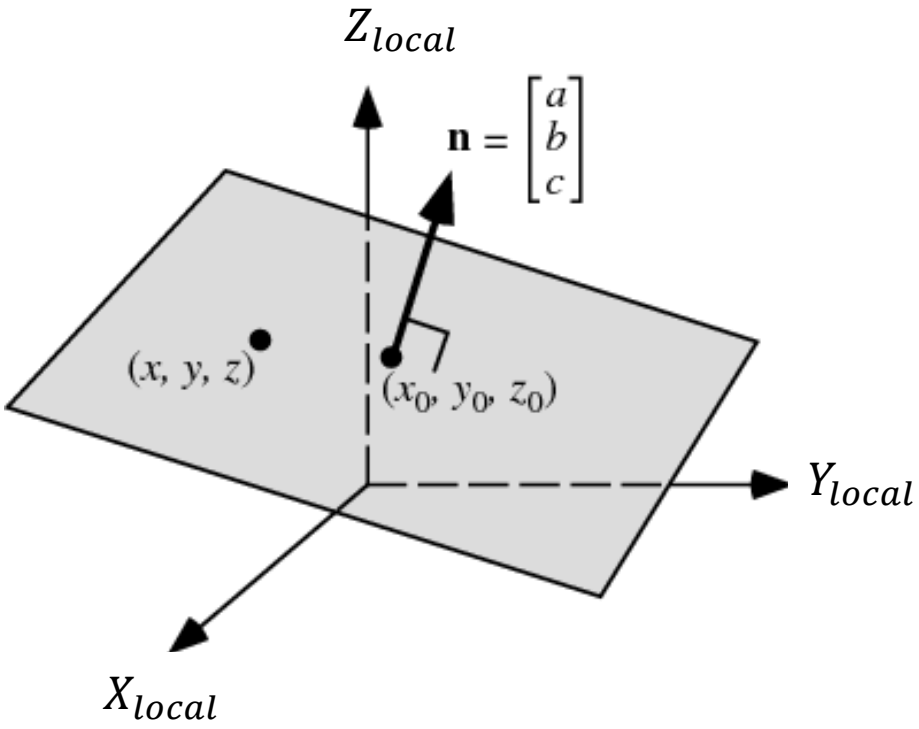
Representation of Plane-Type Shape (e.g. Rectangle or Triangle) in 3D Space

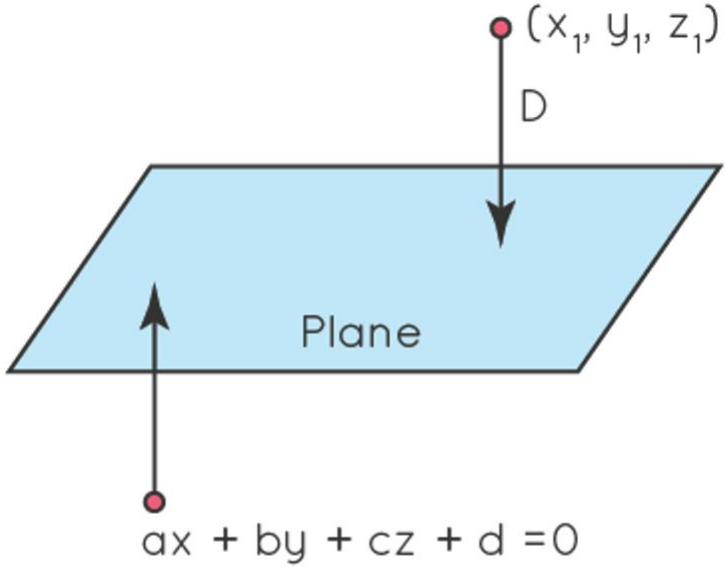
$$ax + by + cz + d = 0$$



How to compute the distance from a point to a plane?

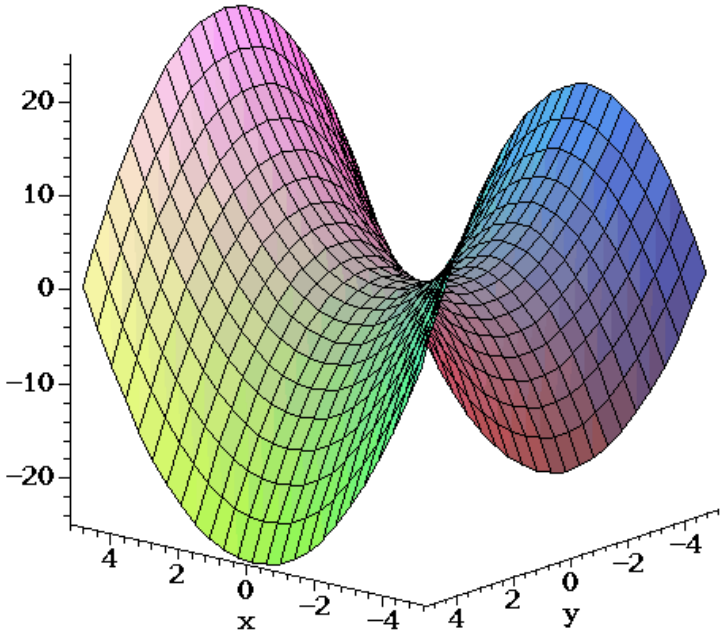
$$ax + by + cz + d = 0$$

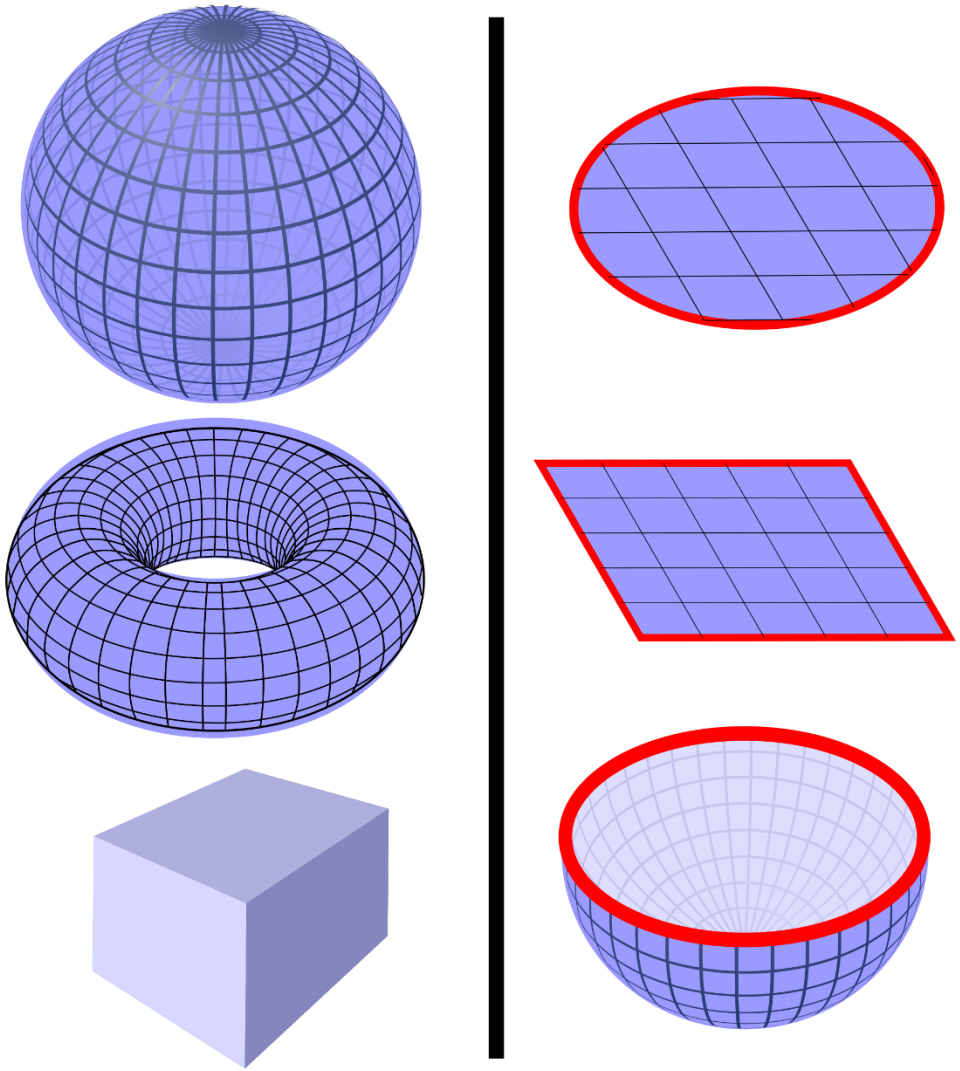




$$D = \frac{|ax_1 + by_1 + cz_1 + d|}{\sqrt{a^2 + b^2 + c^2}}$$

Example





Any surface could be approximated by a matrix of Triangles or Rectangles

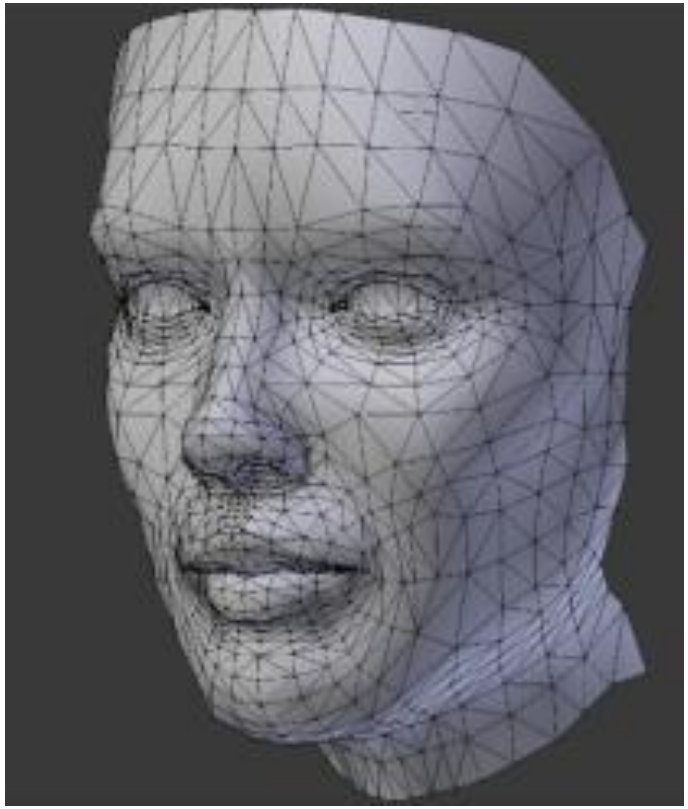
Next Slide

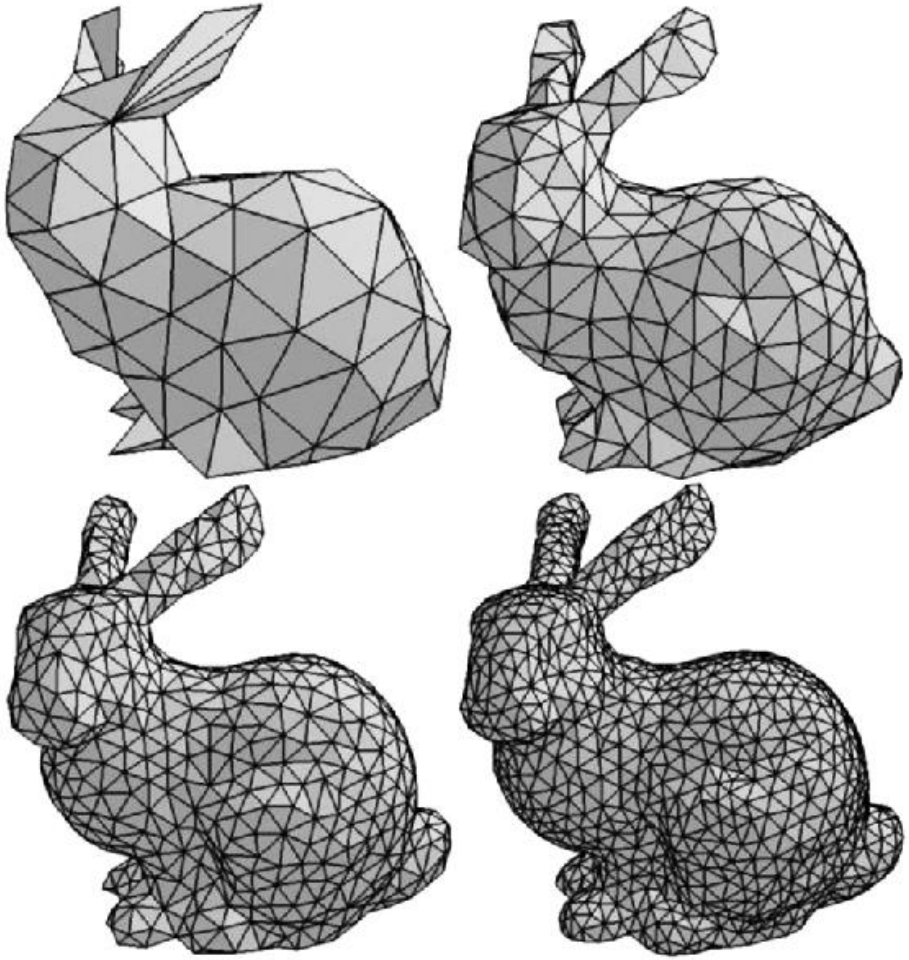
One example ...

Question: Given four points in 3D scene, could they be represented by a rectangle?

More examples …

Question: Given three points in 3D scene, could they be represented by a triangle?

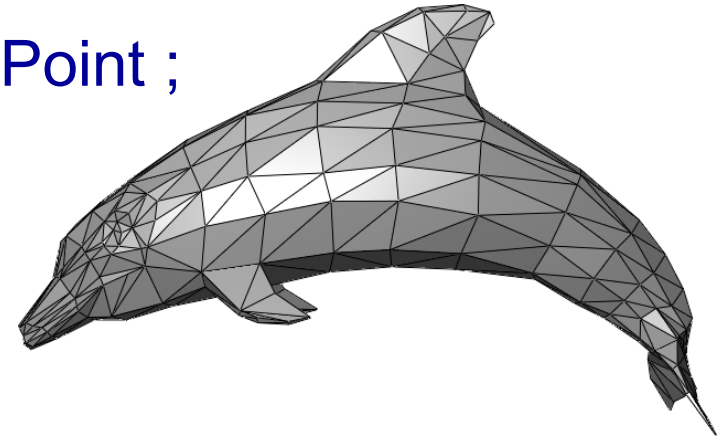




Representation of 3D Surface

Data Structure of 3D Point

```
typedef struct
{
        short    u, v ;
        double  x, y, z ;
}
3DPoint ;
```

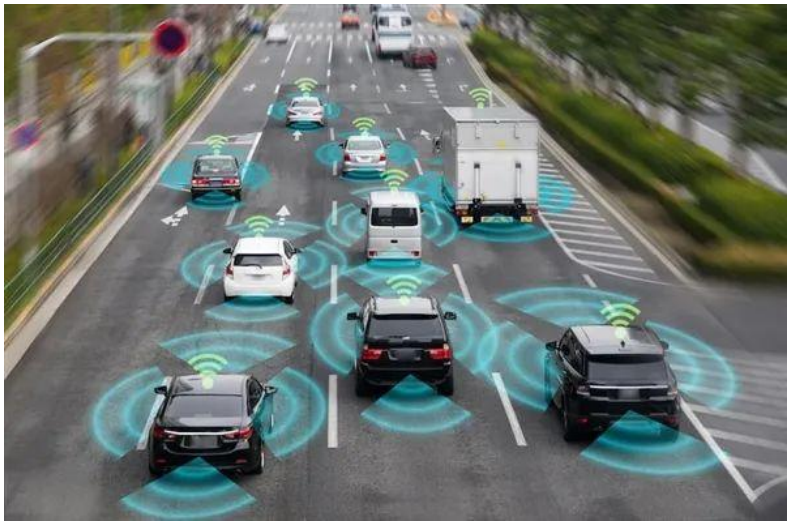


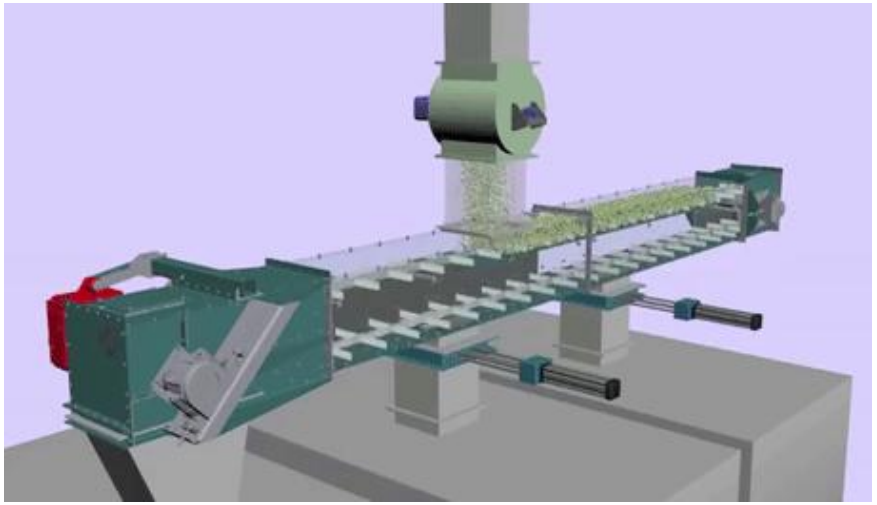
Data Structure of 3D Surface

```
typedef struct
{
        3DPoint triangle_list[][3];
        short    number_triangle;
        short    surface_type;
}
3DSurface;
```

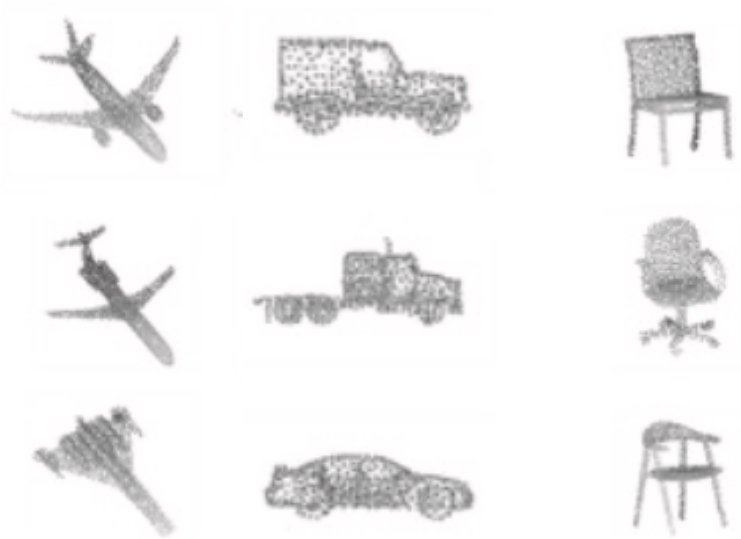
Outline of Lecture 1

- Concept of Knowledge Space
- Representation of 2D Geometry
- Computation of 2D Geometry
- Representation of 3D Geometry
- Computation of 3D Geometry

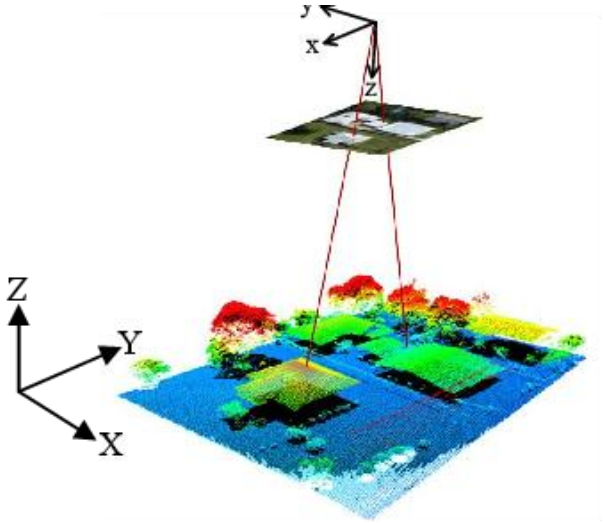




Three Questions …

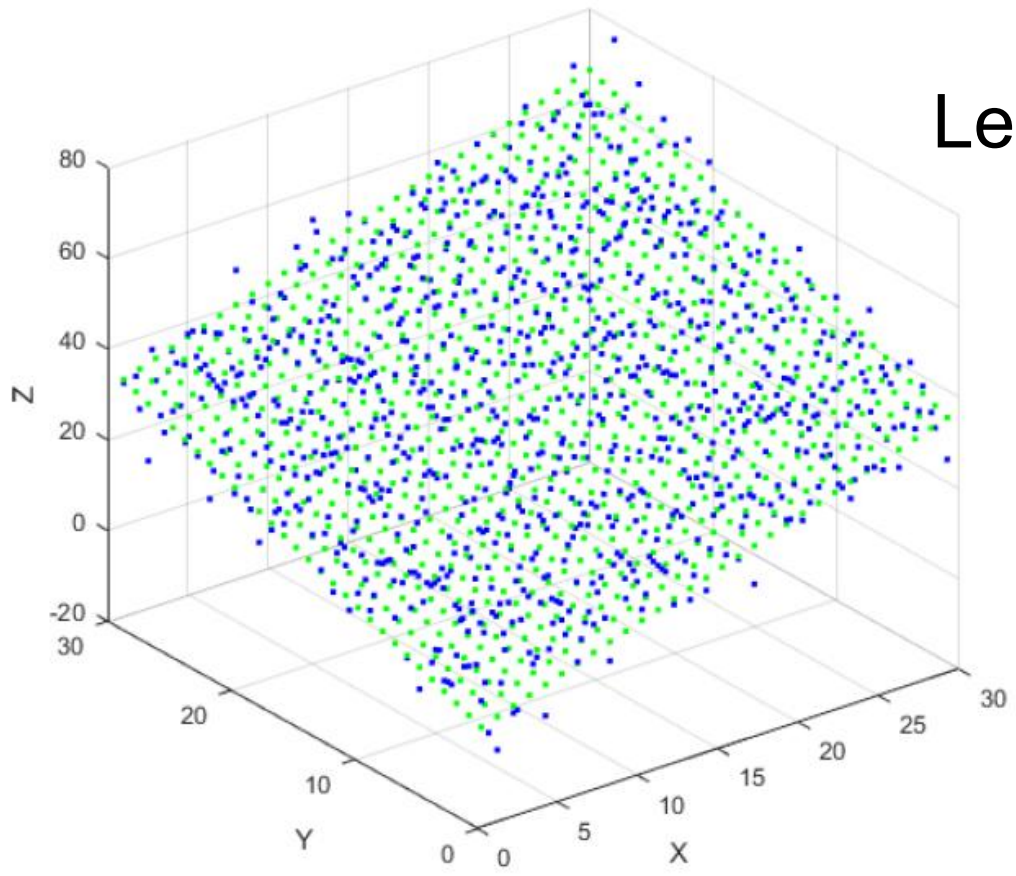


- What are the available input?
- What are the expected outcomes?
- What are the solutions which will produce the expected outcomes from the available input?



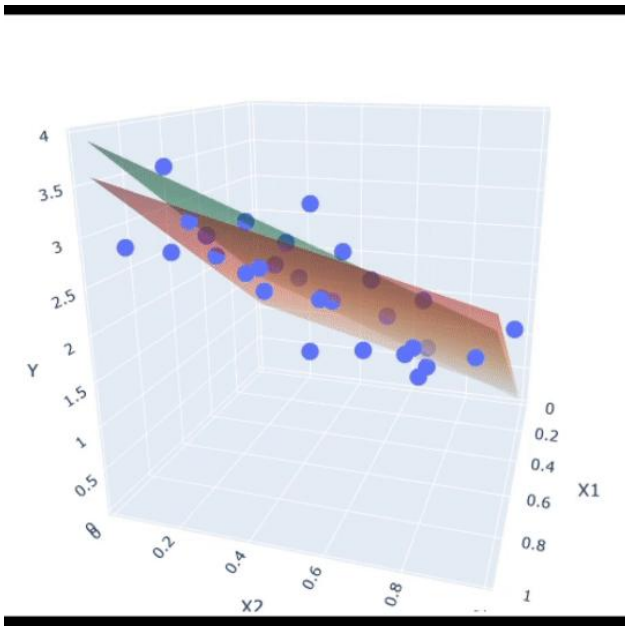
Challenge: How to obtain input?

Lecture 6



Estimation of Equation of Planar Surface

- Input:
  - A list of points as training set, which belong to a planar surface.
    $List = \{(x_i, y_i, z_i), i = 0,1,2, \dots, n\}$
- Output:
  - The coefficients of the equation of line
    $ax + by + cz + d = 1$
- Solution: Least-Squares Method



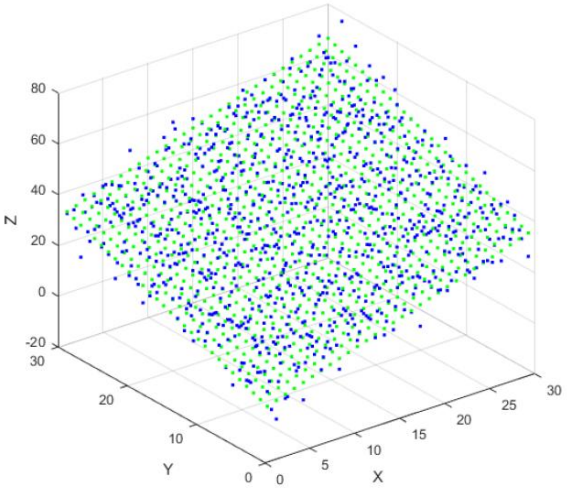
Solution

$$ax_i + by_i + cz_i + d = 1, i = 0,1,2,3, \dots n$$

$$A = \begin{bmatrix} x_0 & y_0 & z_0 & 1 \\ x_1 & y_1 & z_1 & 1 \\ \dots & \dots & \dots & \dots \\ x_n & y_n & z_n & 1 \end{bmatrix} \qquad X = \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} \qquad B = \begin{bmatrix} 1 \\ 1 \\ \dots \\ 1 \end{bmatrix}$$

$$AX = B$$

$$X = (A^t A)^{-1}(A^t B)$$



Example

- What are the parameters of the line which passes through points A(9,2,7) and B(4,8,10)?

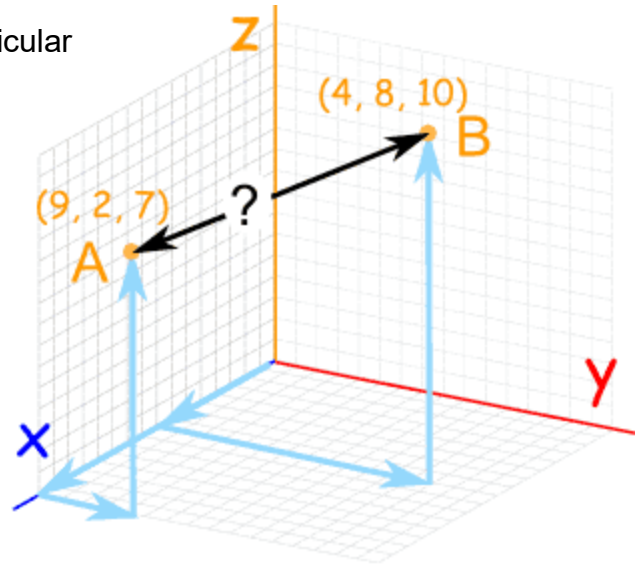
- Answer:

Planar Surface Perpendicular to OXZ Plane

$$x = az + b$$

Planar Surface Perpendicular to OYZ Plane

$$y = cz + d$$



$$9 = 7a + b \qquad 4 = 10a + b$$

$$2 = 7c + d \qquad 8 = 10c + d$$

$$\begin{pmatrix} 7 & 1 & 0 & 0 \\ 0 & 0 & 7 & 1 \\ 10 & 1 & 0 & 0 \\ 0 & 0 & 10 & 1 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} = \begin{pmatrix} 9 \\ 2 \\ 4 \\ 8 \end{pmatrix} \qquad \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} = \begin{pmatrix} 7 & 1 & 0 & 0 \\ 0 & 0 & 7 & 1 \\ 10 & 1 & 0 & 0 \\ 0 & 0 & 10 & 1 \end{pmatrix}^{-1} \begin{pmatrix} 9 \\ 2 \\ 4 \\ 8 \end{pmatrix}$$

Estimation of Equation of Spherical Surface

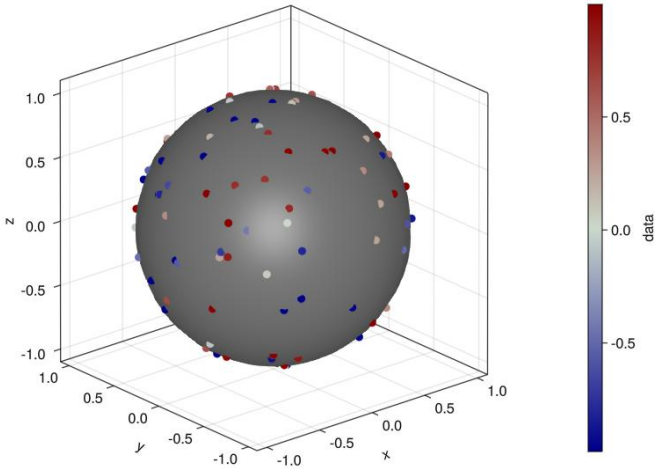
- Input:
  - A list of points as training set, which belong to a spherical surface.

    $List = \{(x_i, y_i, z_i), i = 0,1,2, \dots, n\}$

- Output:
  - The coefficients of the equation of circle

    $(x - c_x)^2 + (y - c_y)^2 + (z - c_z)^2 = r^2$

- Solution: Least-Squares Method



Solution

$$(x_i - c_x)^2 + (y_i - c_y)^2 + (z_i - c_z)^2 = r_i^2, i = 0,1,2,3, \dots n$$

$$x_i^2 - 2x_i c_x + c_x^2 + y_i^2 - 2y_i c_y + c_y^2 + z_i^2 - 2z_i c_z + c_z^2 = r_i^2, i = 0,1,2,3, \dots n$$

$$x_0^2 - 2x_0 c_x + c_x^2 + y_0^2 - 2y_0 c_y + c_y^2 + z_0^2 - 2z_0 c_z + c_z^2 = r_0^2$$

$$x_1^2 - 2x_1 c_x + c_x^2 + y_1^2 - 2y_1 c_y + c_y^2 + z_1^2 - 2z_1 c_z + c_z^2 = r_1^2$$

……

$$x_i^2 - 2x_i c_x + c_x^2 + y_i^2 - 2y_i c_y + c_y^2 + z_i^2 - 2z_i c_z + c_z^2 = r_i^2$$

……

$$x_n^2 - 2x_n c_x + c_x^2 + y_n^2 - 2y_n c_y + c_y^2 + z_n^2 - 2z_n c_z + c_z^2 = r_n^2$$

Solution (continued)

$$2(x_0 - x_1)c_x + 2(y_0 - y_1)c_y + 2(z_0 - z_1)c_z = x_0^2 - x_1^2 + y_0^2 - y_1^2 + z_0^2 - z_1^2$$

……

$$2(x_0 - x_i)c_x + 2(y_0 - y_i)c_y + 2(z_0 - z_i)c_z = x_0^2 - x_i^2 + y_0^2 - y_i^2 + z_0^2 - z_i^2$$

……

$$2(x_0 - x_n)c_x + 2(y_0 - y_n)c_y + 2(z_0 - z_n)c_z = x_0^2 - x_n^2 + y_0^2 - y_n^2 + z_0^2 - z_n^2$$

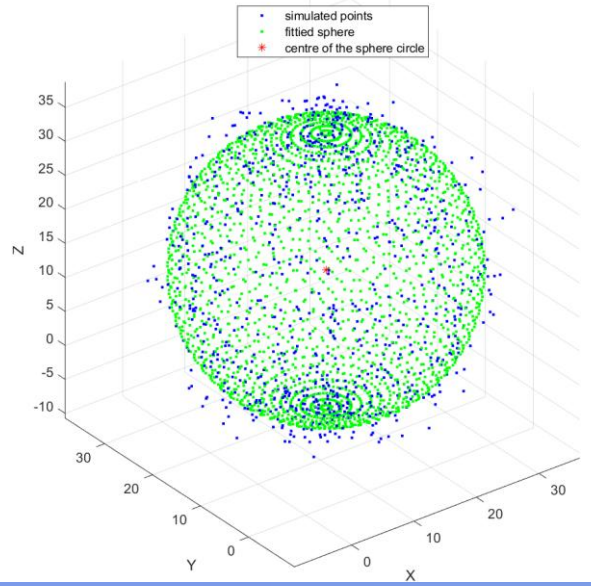
$$A = \begin{bmatrix} 2(x_0 - x_1) & 2(y_0 - y_1) & 2(z_0 - z_1) \\ \cdots & \cdots & \cdots \\ 2(x_0 - x_i) & 2(y_0 - y_i) & 2(z_0 - z_i) \\ \cdots & \cdots & \cdots \\ 2(x_0 - n) & 2(y_0 - y_n) & 2(z_0 - z_n) \end{bmatrix} \quad X = \begin{bmatrix} c_x \\ c_y \\ c_z \end{bmatrix} \quad B = \begin{bmatrix} x_0^2 - x_1^2 + y_0^2 - y_1^2 + z_0^2 - z_1^2 \\ \cdots \\ x_0^2 - x_i^2 + y_0^2 - y_i^2 + z_0^2 - z_i^2 \\ \cdots \\ x_0^2 - x_n^2 + y_0^2 - y_n^2 + z_0^2 - z_n^2 \end{bmatrix}$$

Solution (continued)

$$A = \begin{bmatrix} 2(x_0 - x_1) & 2(y_0 - y_1) & 2(z_0 - z_1) \\ \dots & \dots & \dots \\ 2(x_0 - x_i) & 2(y_0 - y_i) & 2(z_0 - z_i) \\ \dots & \dots & \dots \\ 2(x_0 - n) & 2(y_0 - y_n) & 2(z_0 - z_n) \end{bmatrix} \quad X = \begin{bmatrix} c_x \\ c_y \\ c_z \end{bmatrix} \quad B = \begin{bmatrix} x_0^2 - x_1^2 + y_0^2 - y_1^2 + z_0^2 - z_1^2 \\ \dots \\ x_0^2 - x_i^2 + y_0^2 - y_i^2 + z_0^2 - z_i^2 \\ \dots \\ x_0^2 - x_n^2 + y_0^2 - y_n^2 + z_0^2 - z_n^2 \end{bmatrix}$$

$$AX = B$$

$$X = (A^t A)^{-1}(A^t B)$$

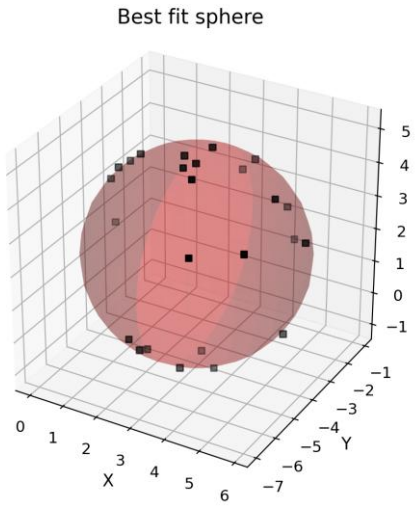


Solution (continued)

$$(x_i - c_x)^2 + (y_i - c_y)^2 + (z_i - c_z)^2 = r_i^2, i = 0,1,2,3, \dots n$$

$$r_i = \sqrt{(x_i - c_x)^2 + (y_i - c_y)^2 + (z_i - c_z)^2}, i = 0,1,2,3, \dots n$$

$$r = \frac{\sum_{i=0}^{n} r_i}{n}$$



Example

- You are given three points on a circle, what are the equations which are sufficient enough to determine the parameters of the circle. How to estimate the parameters inside the equations?

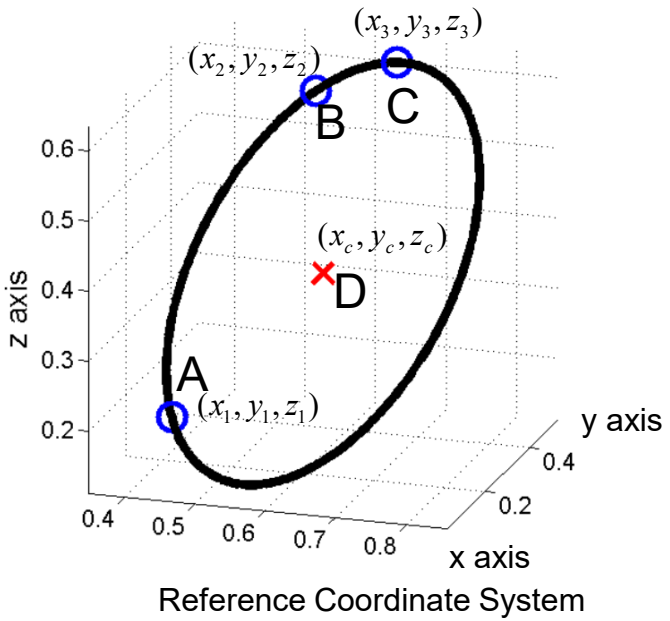
- Answer:

(1) $$(x_1 - x_c)^2 + (y_1 - y_c)^2 + (z_1 - z_c)^2 = r^2$$

(2) $$(x_2 - x_c)^2 + (y_2 - y_c)^2 + (z_2 - z_c)^2 = r^2$$

(3) $$(x_3 - x_c)^2 + (y_3 - y_c)^2 + (z_3 - z_c)^2 = r^2$$

(4) $$ax + by + cz + 1 = 0$$



Reference Coordinate System

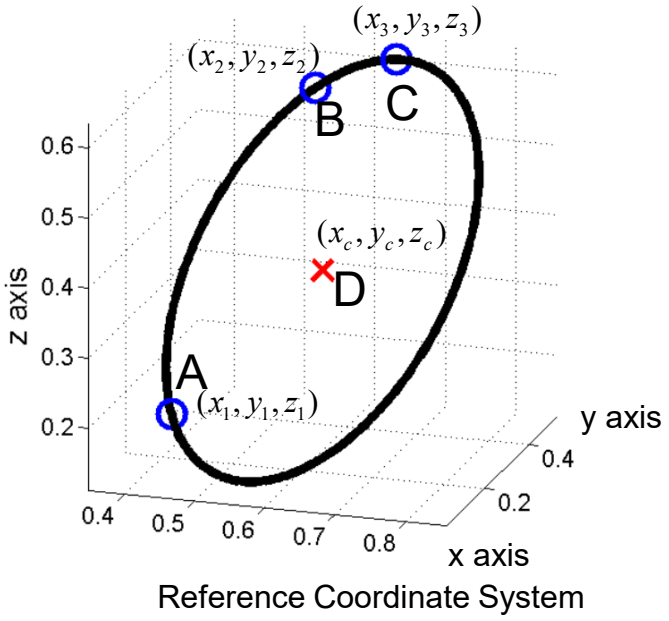
Solution

- First step is to estimate the equation of supporting plane.
- Second step is to estimate the center of the circle.
- Final step is to estimate the radius.

- Hint to first step:

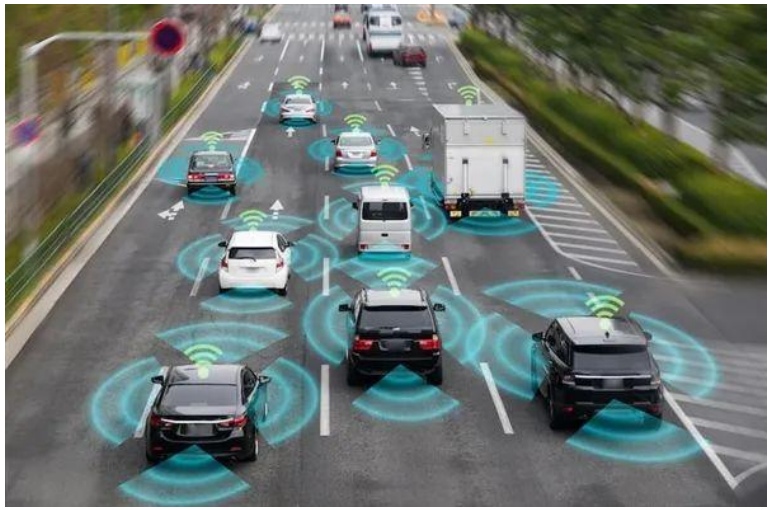
$$ax + by + cz + 1 = 0$$

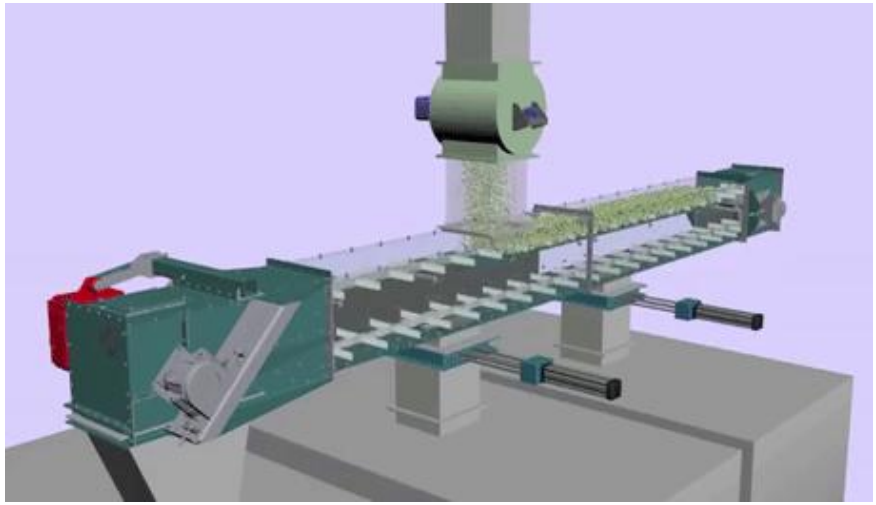
$$\begin{cases} ax_1 + by_1 + cz_1 + 1 = 0 \\ ax_2 + by_2 + cz_2 + 1 = 0 \\ ax_3 + by_3 + cz_3 + 1 = 0 \end{cases}$$



Summary of Lecture 1

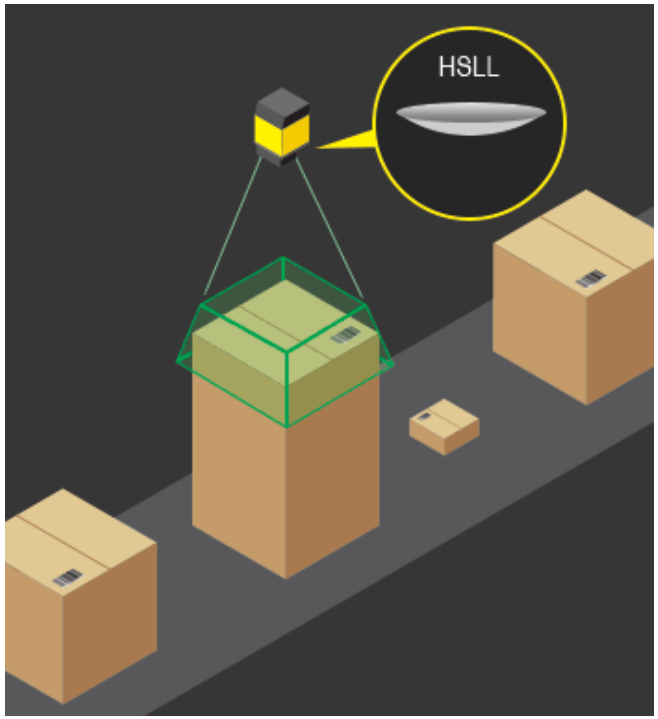
- Concept of Knowledge Space
- Representation of 2D Geometry
- Computation of 2D Geometry
- Representation of 3D Geometry
- Computation of 3D Geometry

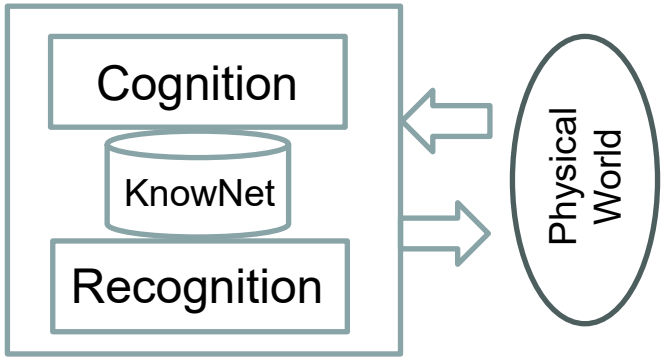




Outline of Module 4

- Lecture 1:
  - Geometry-Driven Computation
- Lecture 2:
  - Fuzziness-Driven Computation
- Lecture 3:
  - Cognition-Driven Computation
- Lecture 4:
  - Recognition-Driven Computation
- Lecture 5:
  - Computation Using Monocular Vision
- Lecture 6:
  - Computation Using Binocular Vision





NANYANG TECHNOLOGICAL UNIVERSITY

School of Mechanical & Aerospace Engineering

Design, Machine, Control, Intelligence

Lecture 2 of Module 4

AI 3.0

MA4829 Machine Intelligence

Fuzziness-Driven Computation

XIE Ming, PhD (France)

http://personal.ntu.edu.sg/mmxie

"We are living inside an ocean of signals"

What is a system with intelligence inside?

Outline of Lecture 2

- Use of Natural Languages
- Fuzziness of Natural Languages
- Concept of Belief's Fuzzy Sets
- Concept of Action's Fuzzy Sets
- Computation of Conceptual Knowledge
- Computation of Control Signals

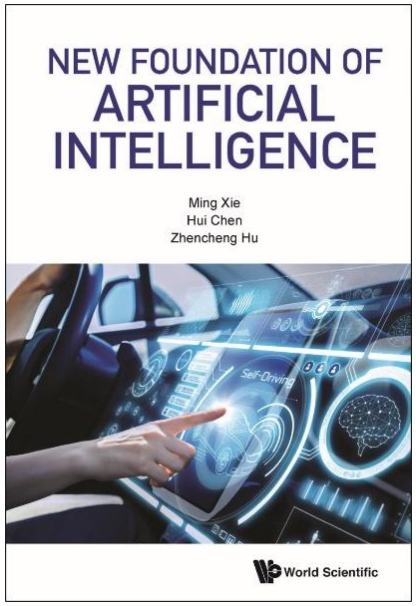
Outline of Lecture 2

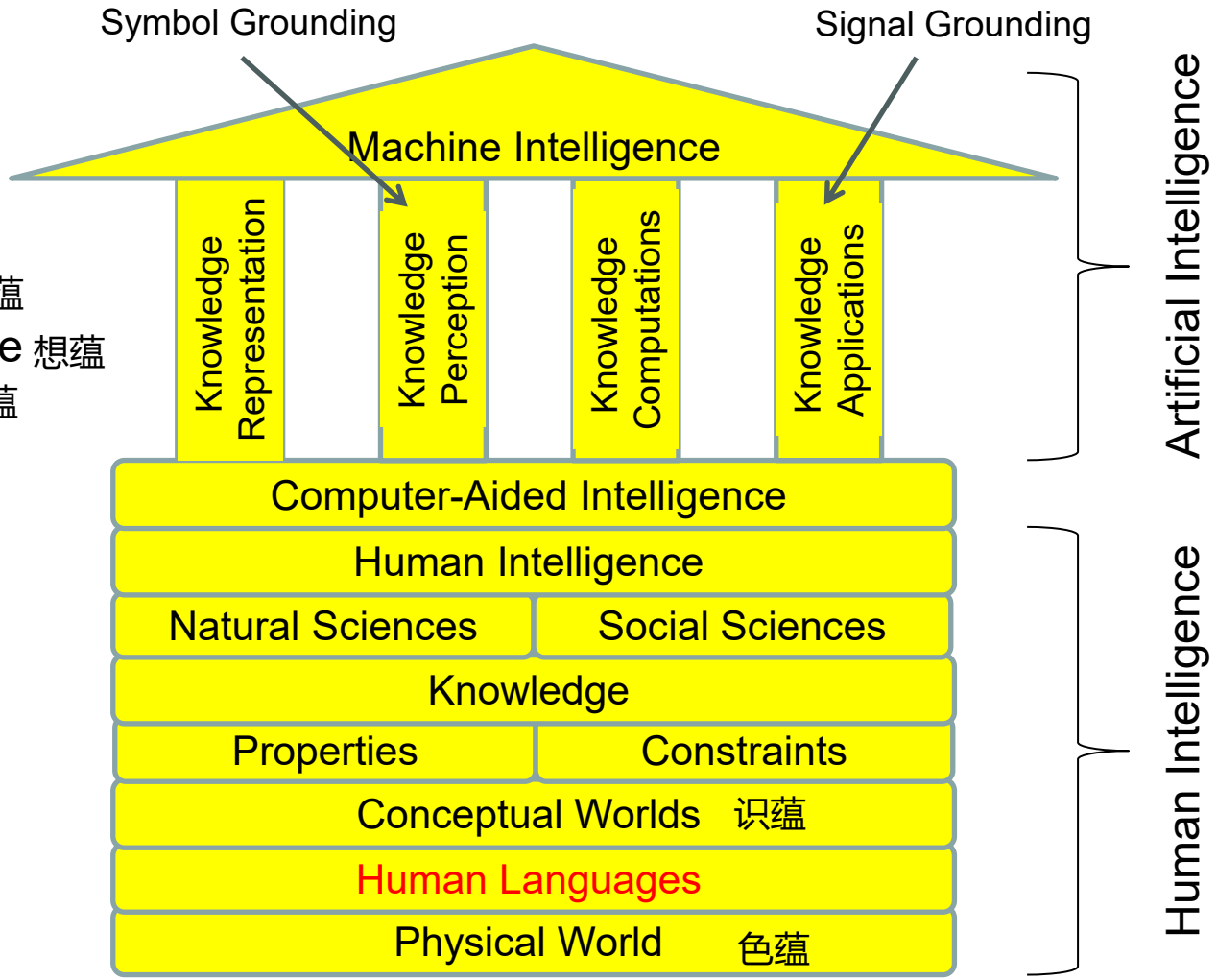
- Use of Natural Languages
- Fuzziness of Natural Languages
- Concept of Belief's Fuzzy Sets
- Concept of Action's Fuzzy Sets
- Computation of Conceptual Knowledge
- Computation of Control Signals

Landscape of Authentic AI …

- One Tool
- Two Worlds
- Three Intelligences
- Four Pillars

- Signal to Knowledge 受蕴
- Knowledge to Knowledge 想蕴
- Knowledge to Signal 行蕴





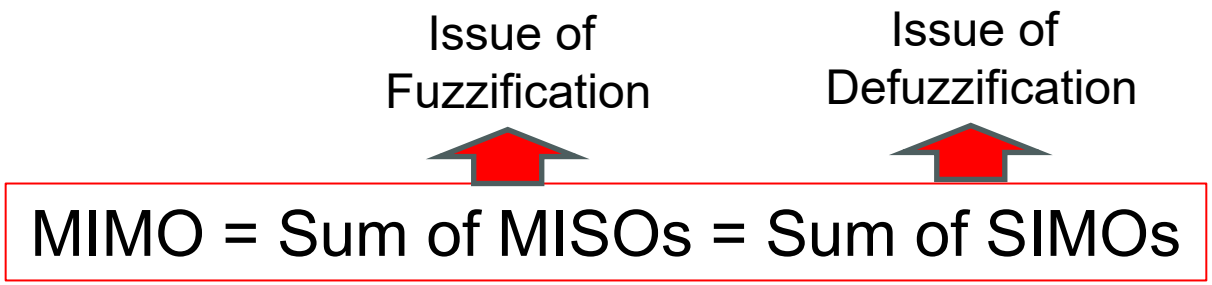
What is a human language?

- Human languages are tools for our sensory systems to encode knowledge perceived from the physical world.
- The outcome of such encoding are texts which form the so-called conceptual worlds.
- Texts in conceptual worlds enable us to reconstruct (i.e., decoding) the knowledge in the physical world.

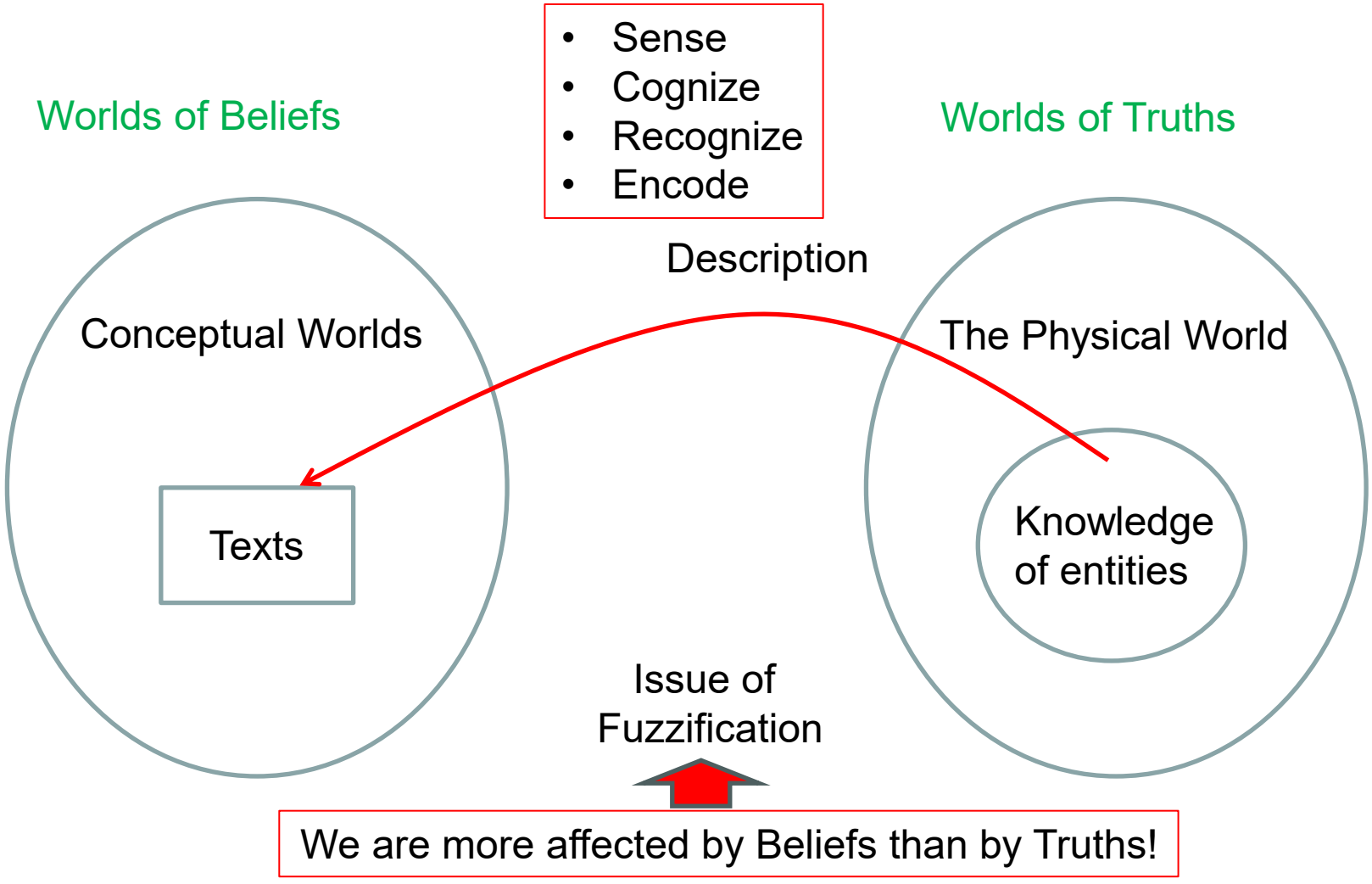
There Are Two Purposes for Human Beings to Use Human Languages

- Knowledge Description (via Perception)
  - From physical world to conceptual worlds

- Knowledge Understanding (via Reconstruction)
  - From conceptual worlds to physical world



Knowledge Description: Illustration

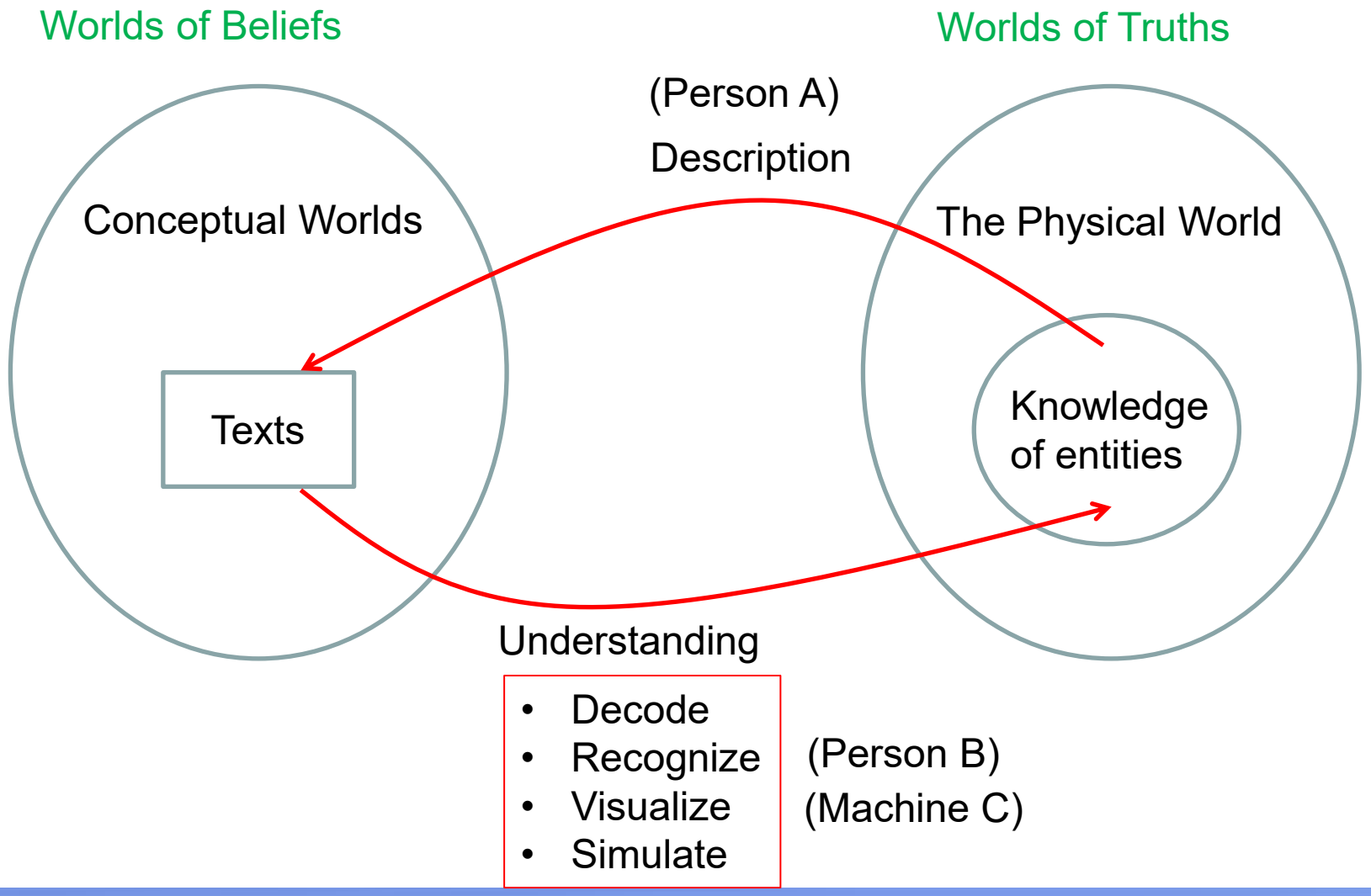


Process of Knowledge Description

It involves two steps:

- Use of sensory systems to sense, cognize, and recognize, the knowledge.
- Use of words, phrases and sentences to encode the cognized, or recognized, knowledge.

Knowledge Understanding: Illustration



Process of Knowledge Understanding

It involves two steps:

- To decode, or recognize, knowledge from texts.
- To visualize, or simulate, the recognized knowledge in the physical world.

Outline of Lecture 2

- Use of Natural Languages
- Fuzziness of Natural Languages
- Concept of Belief's Fuzzy Sets
- Concept of Action's Fuzzy Sets
- Computation of Conceptual Knowledge
- Computation of Control Signals

Example of Human-Robot Interaction Using Natural Language with Fuzziness …

Clearly, the sensory systems of human beings could output accurate data to a certain extent (this is an advantage). For example, …

- A car is moving.
- A car is following another car.
- A car stops at a junction.
- There is a pedestrian.

Car's Size?

Car's Speed?

Car's Weight?

It seems that our sensory systems intentionally produce fuzzy outputs (this is an advantage). For example, ...

Example 1

- A car is moving very fast.
- A car is following another car closely.
- That car is beautiful and large.
- This car is powerful.

Example 2

- That truck is heavily loaded.
- That car is very noisy.
- The street is very crowded.
- The street-light is too dim.

Discussion: Is fuzziness about belief or vagueness?

Challenges Faced by Authentic AI

- All knowledge in the physical world are crisp.
- Not all knowledge in conceptual worlds are crisp (NOTE: this is an advantage).

- Challenge 1: (Pillar No.2 of AI)
  - How to transform crisp sensory data into fuzzy sensory beliefs or conceptual knowledge?

→ Use of Belief's Fuzzy Sets

- Challenge 2: (Pillar No.4 of AI)
  - How to transform fuzzy linguistic instructions into crisp signals for action-taking or behavior-control?

→ Use of Action's Fuzzy Sets

Example of Fuzzy Set Associated with Belief "Red Color"

- The sensory data (255, 0, 0) could trigger the belief of red color.
- The sensory data (255, 1, 0) could also trigger the belief of red color.
- The sensory data (255, 1, 1) could also trigger the belief of red color.

Example of Fuzzy Set Associated with Belief "Tall Man"

- A tall man may refer to a man of the height of 1.80 m.
- A tall man may refer to a man of the height of 1.85 m
- A tall man may refer to a man of the height of 1.90 m

{A tall man} ⟺ {Infinite Number of Instances of Height}

This is an advantage of doing knowledge compression!

Outline of Lecture 2

- Use of Natural Languages
- Fuzziness of Natural Languages
- Concept of Belief's Fuzzy Sets
- Concept of Action's Fuzzy Sets
- Computation of Conceptual Knowledge
- Computation of Control Signals

Important Remarks

- The sensory systems of human beings could not output crisp data.

- The sensory systems of human beings output sensory beliefs.

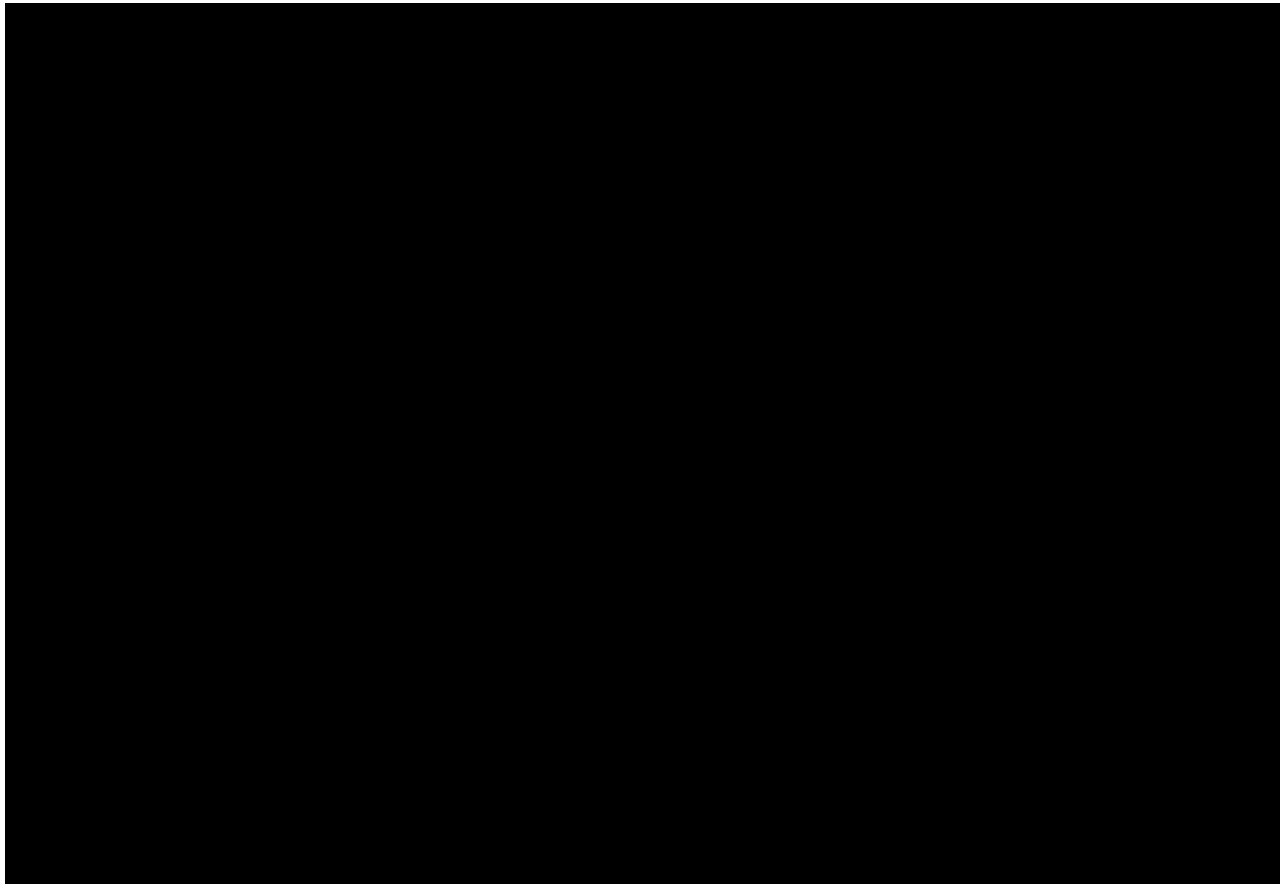
How is the traffic condition?

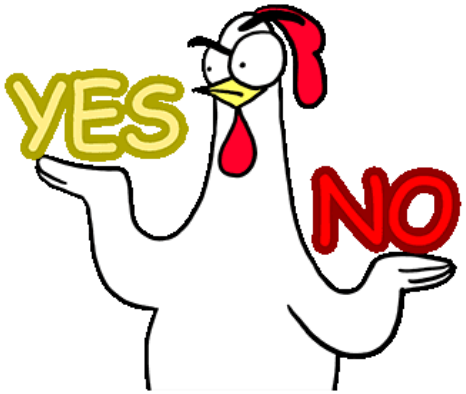
Issue of Fuzzification

Issue of Defuzzification

MIMO = Sum of MISO

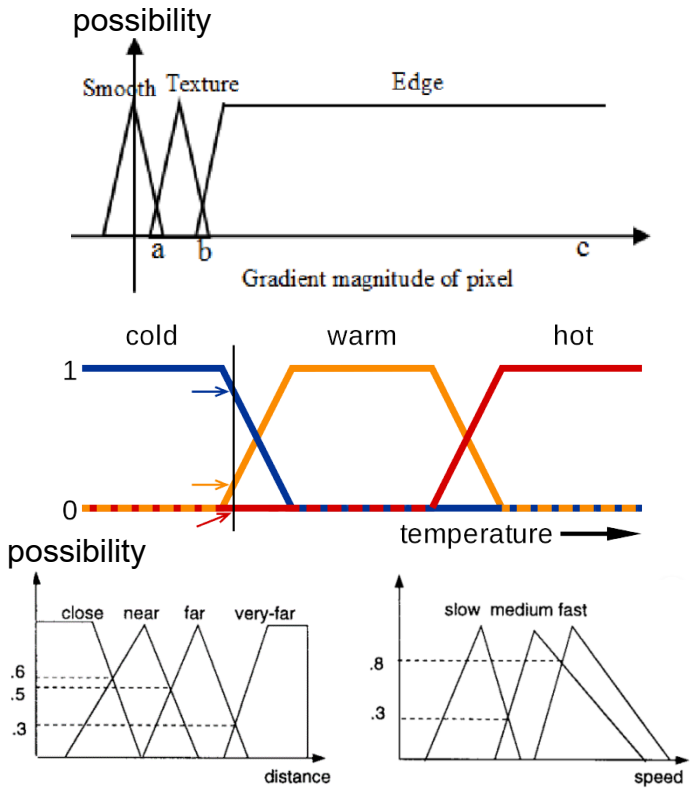
Belief will affect us more than truth





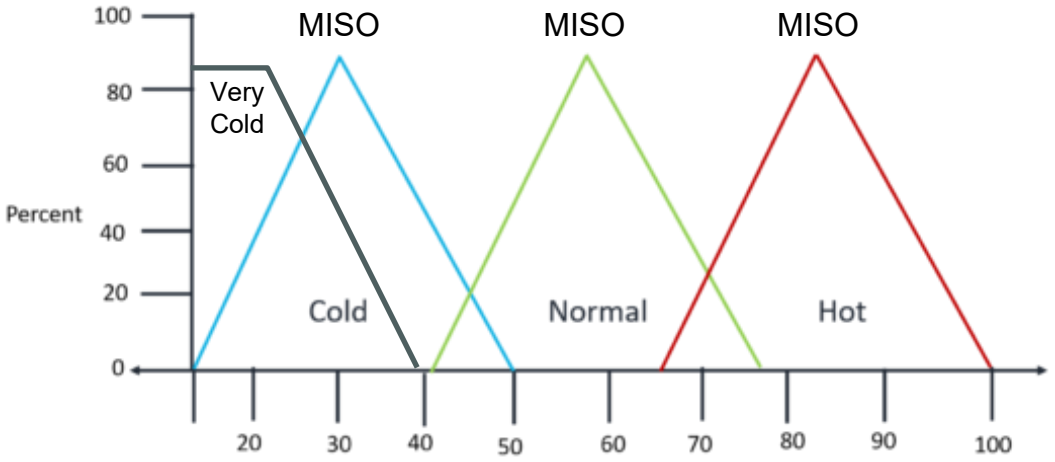
What is a sensory belief which is fuzzy?

- It refers to a specific knowledge in terms of properties, constraints or behaviors, which is associated with a set of sensory data with certain degrees of belief. Such association is one-to-many association. One belief normally correspond to a set of sensory data (i.e., MISO).



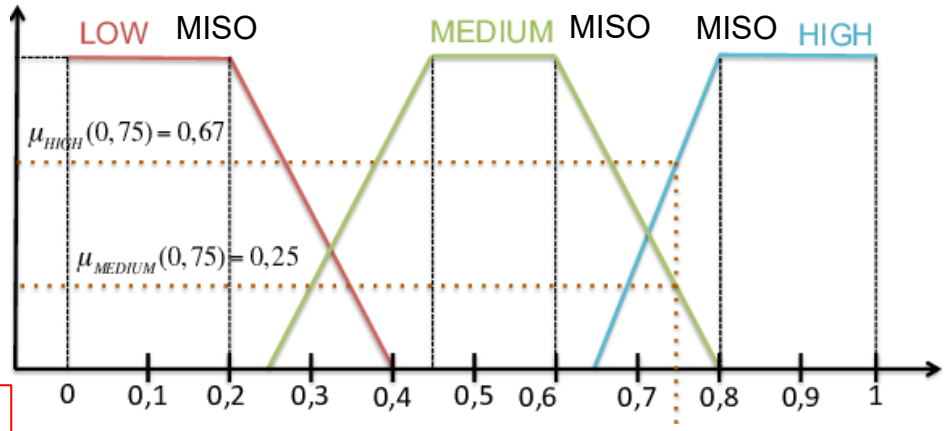
What is a belief's fuzzy set?

- All the sensory data, which could trigger a same belief with different levels of possibility, consist of a belief's fuzzy set.



Crisp Values of Sensory Data

- Hence, a belief's fuzzy set corresponds to a set of sensory data associated with possibility values.

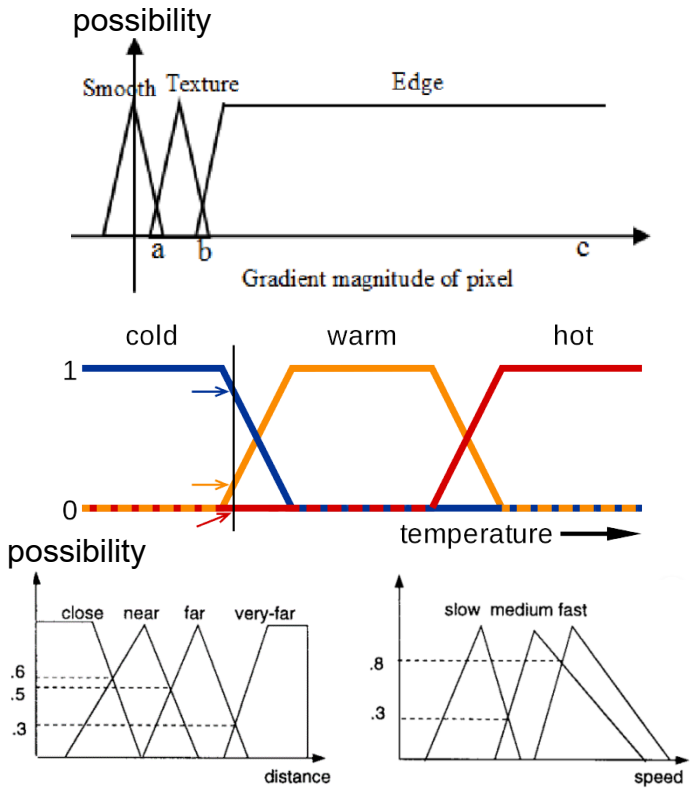


Crisp Values of Sensory Data

MIMO = Sum of MISO

Definition of Possibility Function

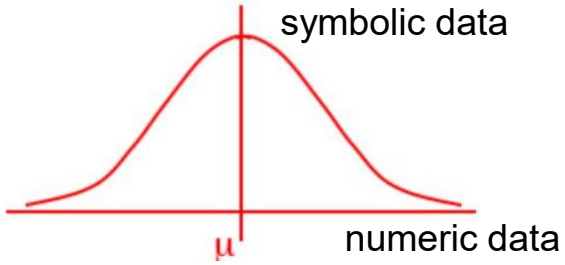
- It is a function which takes values between 0 and 1.
- In fuzzification, such values indicate the possibilities for a set of sensory data to trigger a fuzzy belief about specific knowledge (1st type of output from AI systems).
- In defuzzification, such values reveal the possibilities for a fuzzy action to produce a set of crisp values for controlling specific behaviors (2nd type of output from AI systems).



How to determine possibility functions?

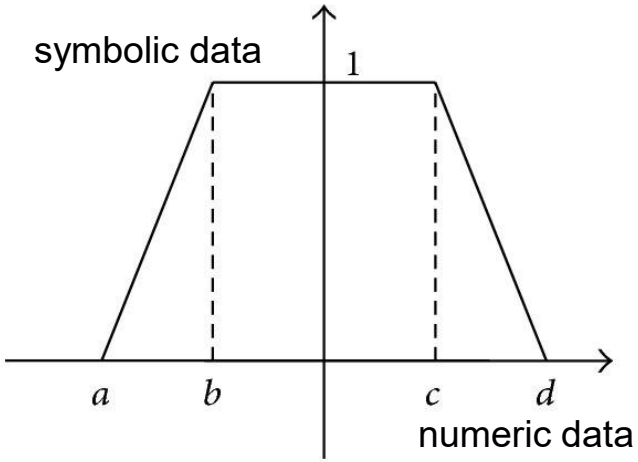
– Possibility functions such as normalized Gaussian functions

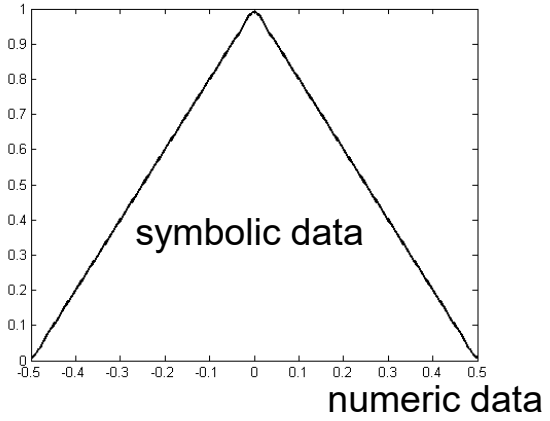
$$P(x) = e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$



– Possibility functions such as trapezoidal functions

– Possibility functions such as triangular functions





Representation of Belief's Fuzzy Set

- Given a set of n crisp sensory data (input):

$$\mathrm{X} = \{x_i, i = 1,2,3, \dots, n\}$$

- Given a set of m fuzzy beliefs (output) and their possibility functions:

$$B = \{(b_j, \mu_j(x_i)), i = 1,2,3, \dots, n;\ \ j = 1,2,3, \dots, m\}$$

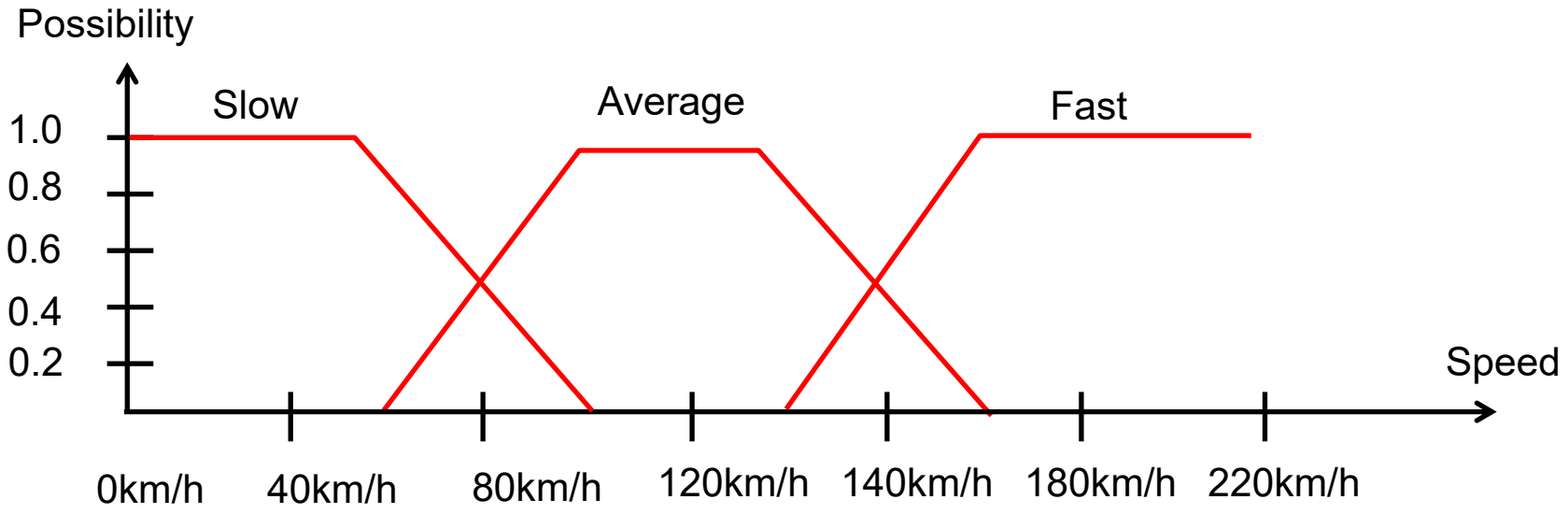
- Then, the fuzzy set associated with j-th fuzzy belief is represented as:

$$F_B(b_j) = \{(x_i, \mu_j(x_i)), i = 1,2, \dots, n\},\ \mathrm{j} = 1, 2, 3, \dots, \mathrm{m}$$

MIMO = Sum of MISO

Example of Fuzzy Sets Associated with Beliefs

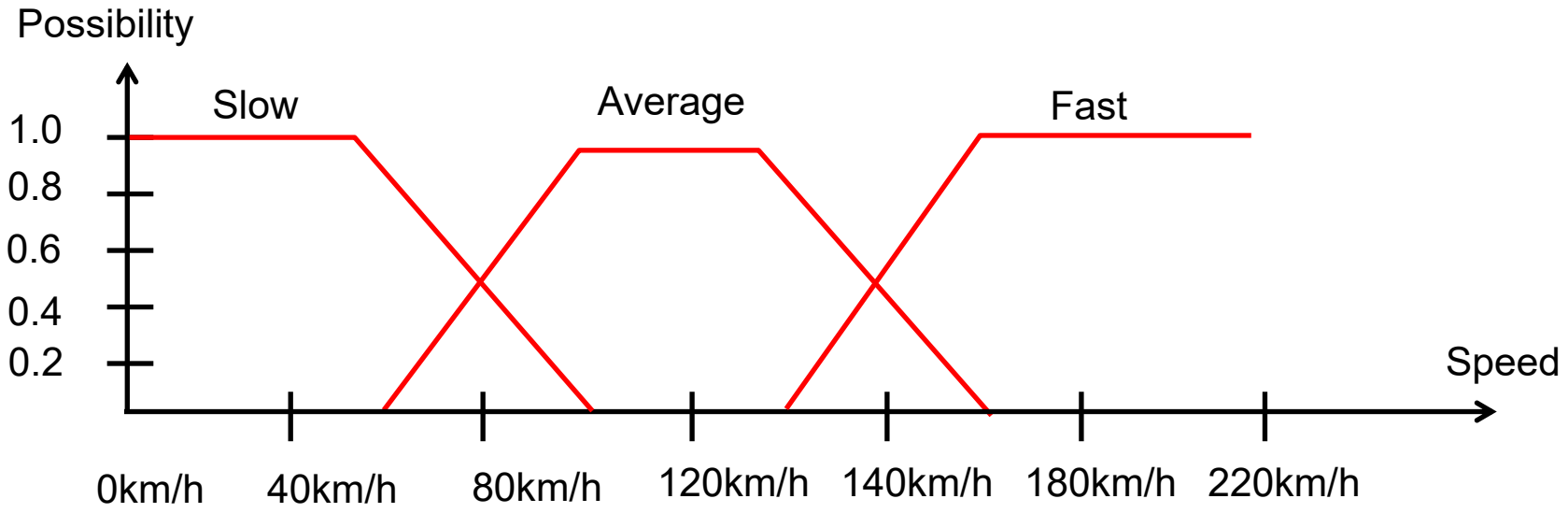
- As shown in the figure below, what are the three fuzzy sets associated with the three fuzzy beliefs if the speed's sensory data set is {40, 80, 120, 140, 180, 220} (km/h)?



MIMO = Sum of MISO

Answer

- $F_B(Slow)$ = {(40, 1.0), (80, 0.5), (120, 0), (140, 0), (180, 0), (220, 0)}
- $F_B(Normal)$ = {(40, 0), (80, 0.5), (120, 1.0), (140, 0.5), (180, 0), (220, 0)}
- $F_B(Fast)$ = {(40, 0), (80, 0), (120, 0), (140, 0.5), (180, 1.0), (220, 1.0)}



Outline of Lecture 2

- Use of Natural Languages
- Fuzziness of Natural Languages
- Concept of Belief's Fuzzy Sets
- Concept of Action's Fuzzy Sets
- Computation of Conceptual Knowledge
- Computation of Control Signals

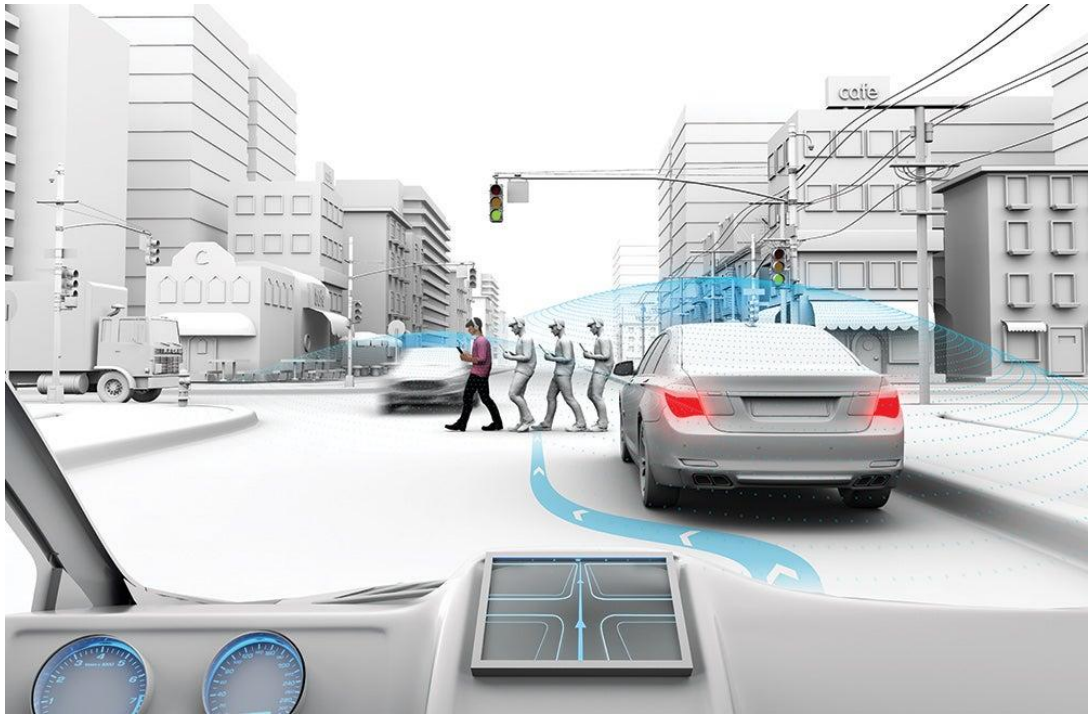
Important Remarks

- Human beings are skillful with the use of sensory beliefs in order to discover knowledge.

- Human beings are also skillful with the use of sensory beliefs in order to determine crisp values of control signal for action-taking.

Fuzzy Decision of Action

If pedestrians are present, stop car immediately



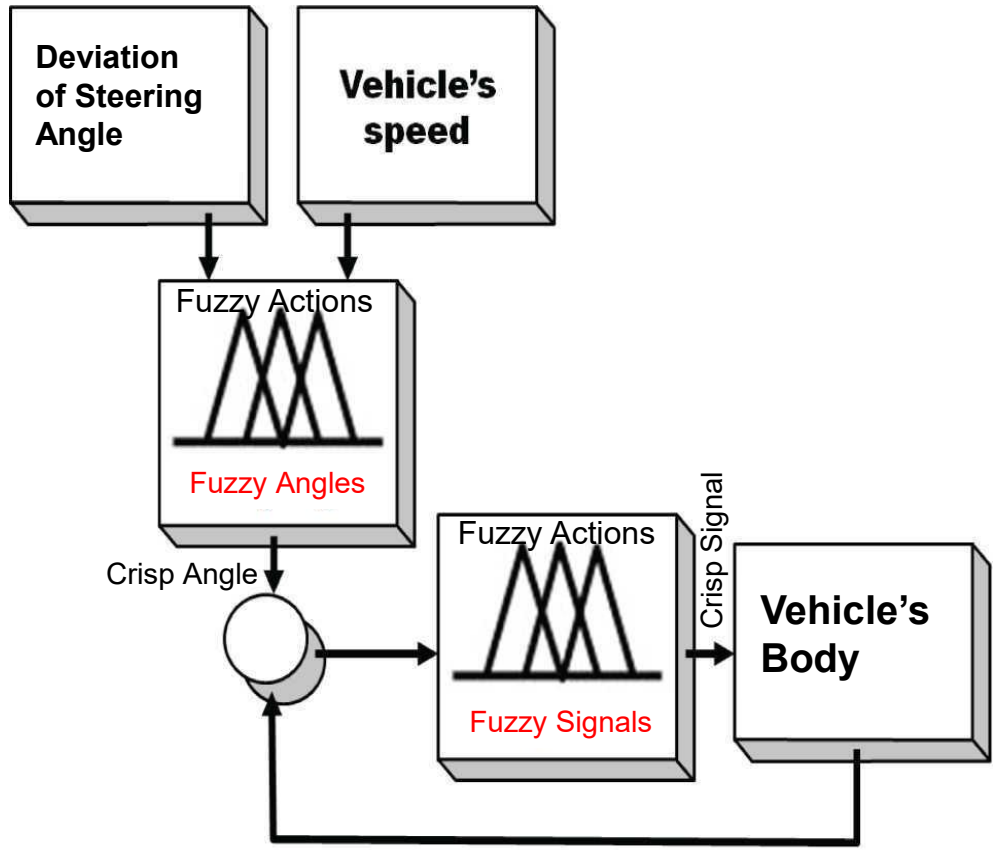
If fuzzy conditions, then fuzzy decisions of actions.

If fuzzy decisions, then crisp values for action-taking.

Question to Focus on …

- How to transform fuzzy decisions of actions into crisp values of actions which will enable signal-driven controls?

- Answer:
  - Use of Action's Fuzzy Sets

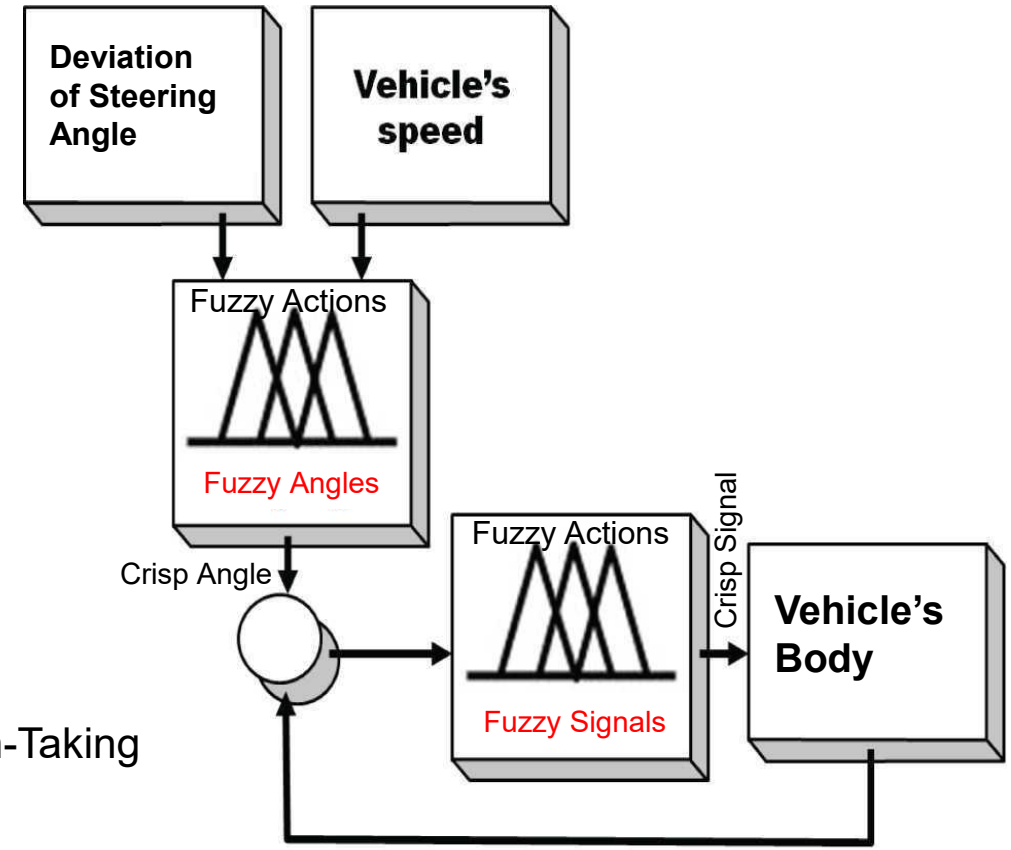


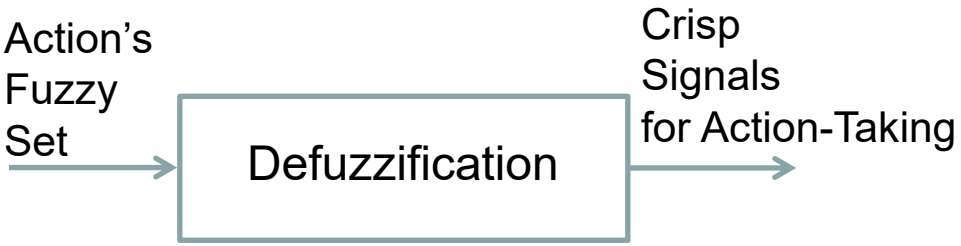
From Fuzzy Decisions of Actions to Fuzzy Sets

From Fuzzy Sets to Crisp Values for Action-Taking

What is an action's fuzzy set?

- An action's fuzzy set is a set of fuzzy decisions associated with a single crisp value which enable action-taking or behavior-control.





MIMO = Sum of MISO

From Fuzzy Decisions of Actions to Fuzzy Sets

From Fuzzy Sets to Crisp Values for Action-Taking

Example of Guessing an Action's Fuzzy Set ...

- A car's speed is measured to be 140 km/h. Then, this crisp data of action may reveal fuzzy decisions of action such as:
  - Person A may say: "I am 60% certain that the car has received the fuzzy command which asks it to move at a normal speed".
  - Person B may say: "I am 90% certain that the car has received the fuzzy command which asks it to move a quite fast speed".

- A car's speed is measured to be 50 km/h. Then, this crisp data of action may reveal fuzzy decisions of action such as:
  - Person C may say: "I am 40% certain that the car has received the fuzzy command which asks it to move slowly".
  - Person D may say: "I am 70% certain that the car has received the fuzzy command which asks it to move at a normal speed".

Representation of Action's Fuzzy Set

- Given a set of n crisp data (output) and m fuzzy decisions (input):

$$Y = \{y_i, i = 1,2,3, \dots, n\} \qquad D = \{d_j, j = 1,2,3, \dots, m\}$$

- The possibility function associated with j-th fuzzy decision is: $\mu_j(y_i)$

- The possibility value for j-th fuzzy decision to output i-th crisp data is:

$$P(d_j) = \mu_j(y_i)$$

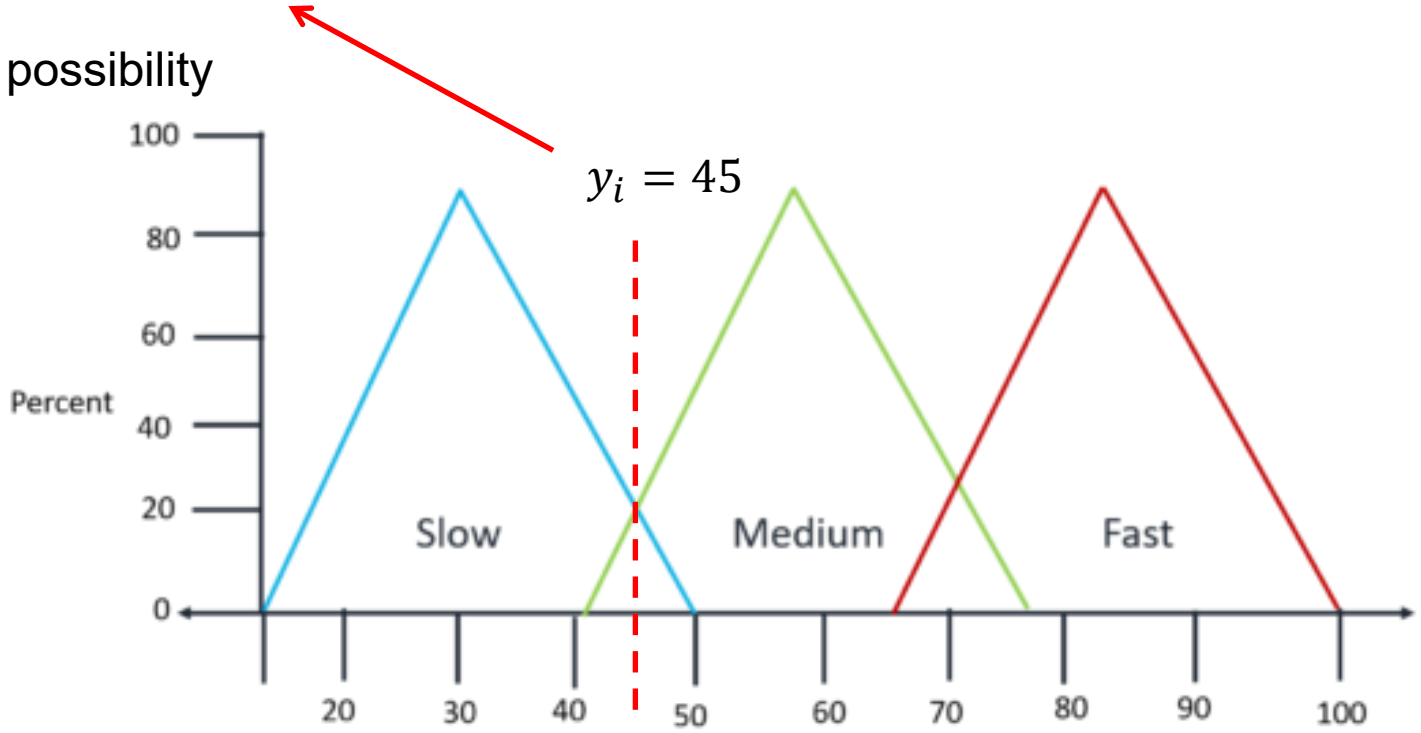
- Then, the fuzzy set associated with i-th crisp data is represented as:

$$F_A(y_i) = \{(d_j, \mu_j(y_i)), j = 1,2,3, \dots, m\}, \text{ i = 1, 2, 3, …, n}$$

MIMO = Sum of MISO

Example of Action's Fuzzy Set for Vehicle's Speed Control …

$$F_A(45) = \{(drive\ slowly, 0.2), (drive\ mediumly, 0.2), (drive\ fast, 0.0)\}$$

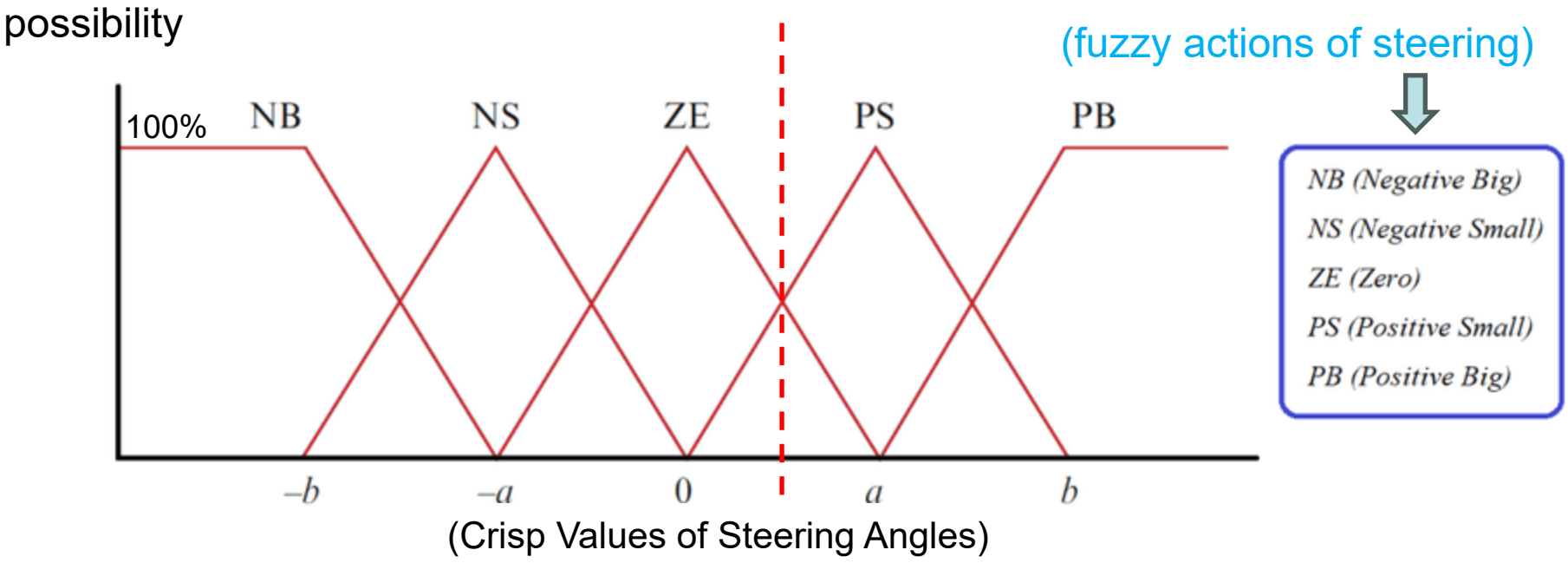


Crisp Values of Speed Control Data

Example of Action's Fuzzy Set for Steering Angle Control …

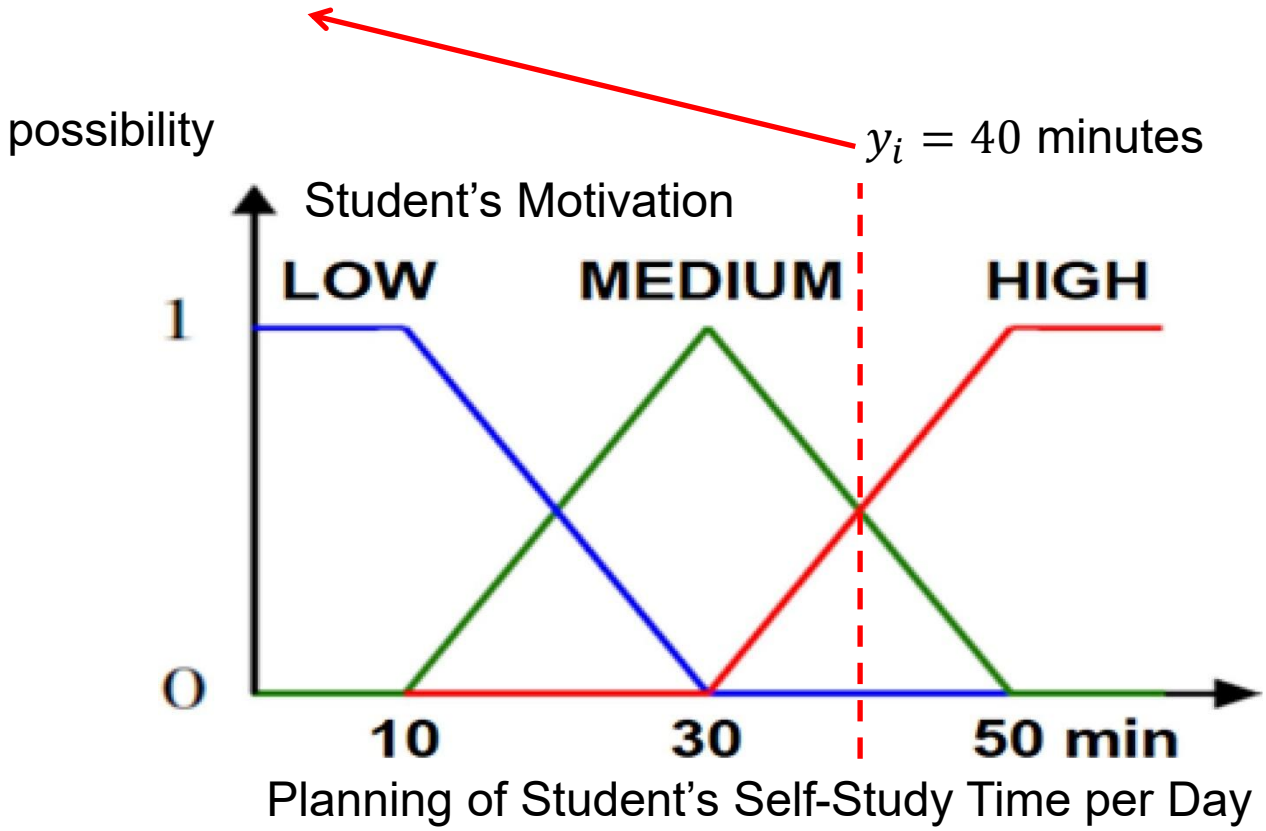
$$F_A(a/2) = \{(NB, 0.0), (NS\ 0.0), (ZE, 0.5), (PS, 0.5), (PB, 0.0)\}$$

$$y_i = \frac{a}{2}$$



Example of Action's Fuzzy Set (i.e., Planning) for Daily Self-Study Time ...

$$F_A(40) = \{(low\ motivation, 0.0), (medium\ motivation, 0.5), (high\ motivation, 0.5)\}$$

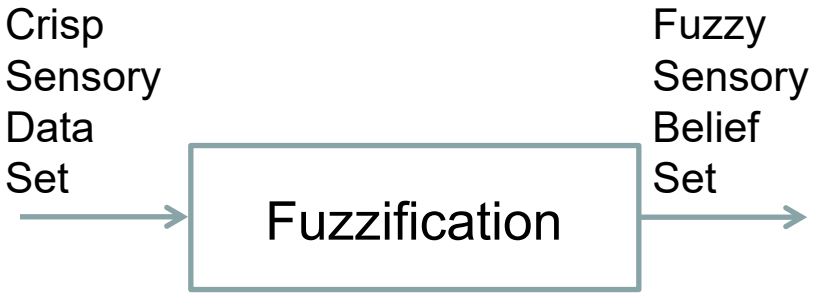


Outline of Lecture 2

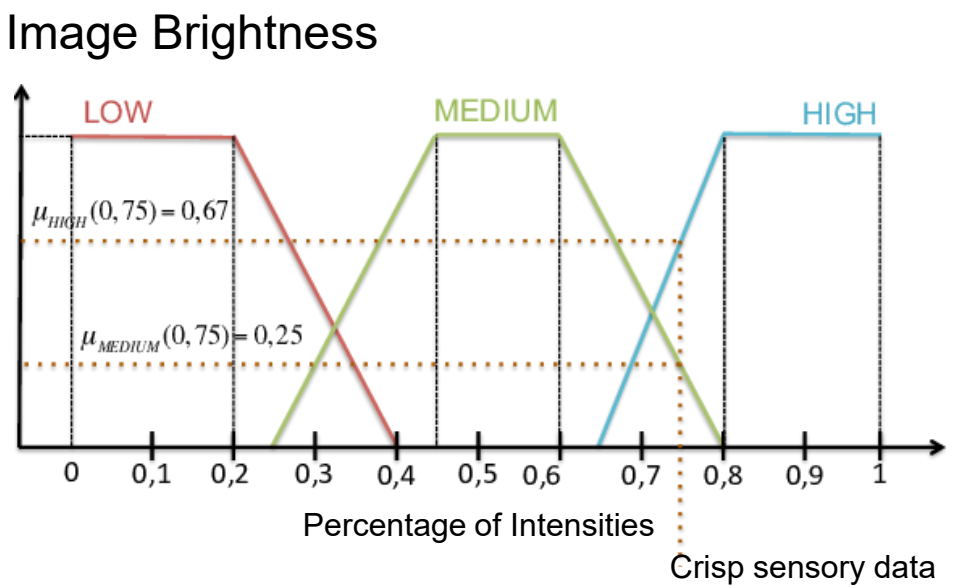
- Use of Natural Languages
- Fuzziness of Natural Languages
- Concept of Belief's Fuzzy Sets
- Concept of Action's Fuzzy Sets
- Computation of Conceptual Knowledge
- Computation of Control Signals

Applications of Fuzzy Sets Associated with Beliefs

- Transformation of sensory data into sensory beliefs.
- Solution to the Symbol Grounding Problem in AI.



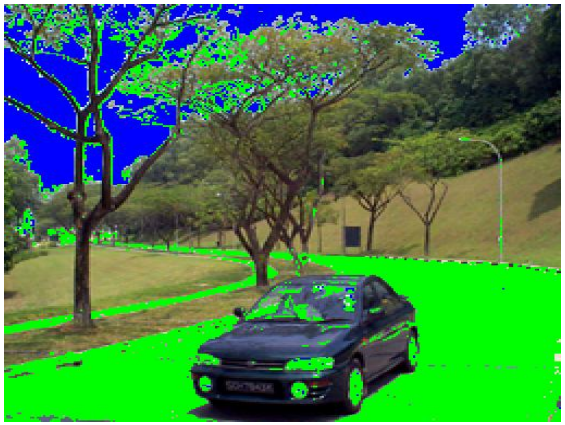
MIMO = Sum of MISO



Example 1: Use of Colour Images to Do Pixel Groupings

(a). An Input Image from NTU

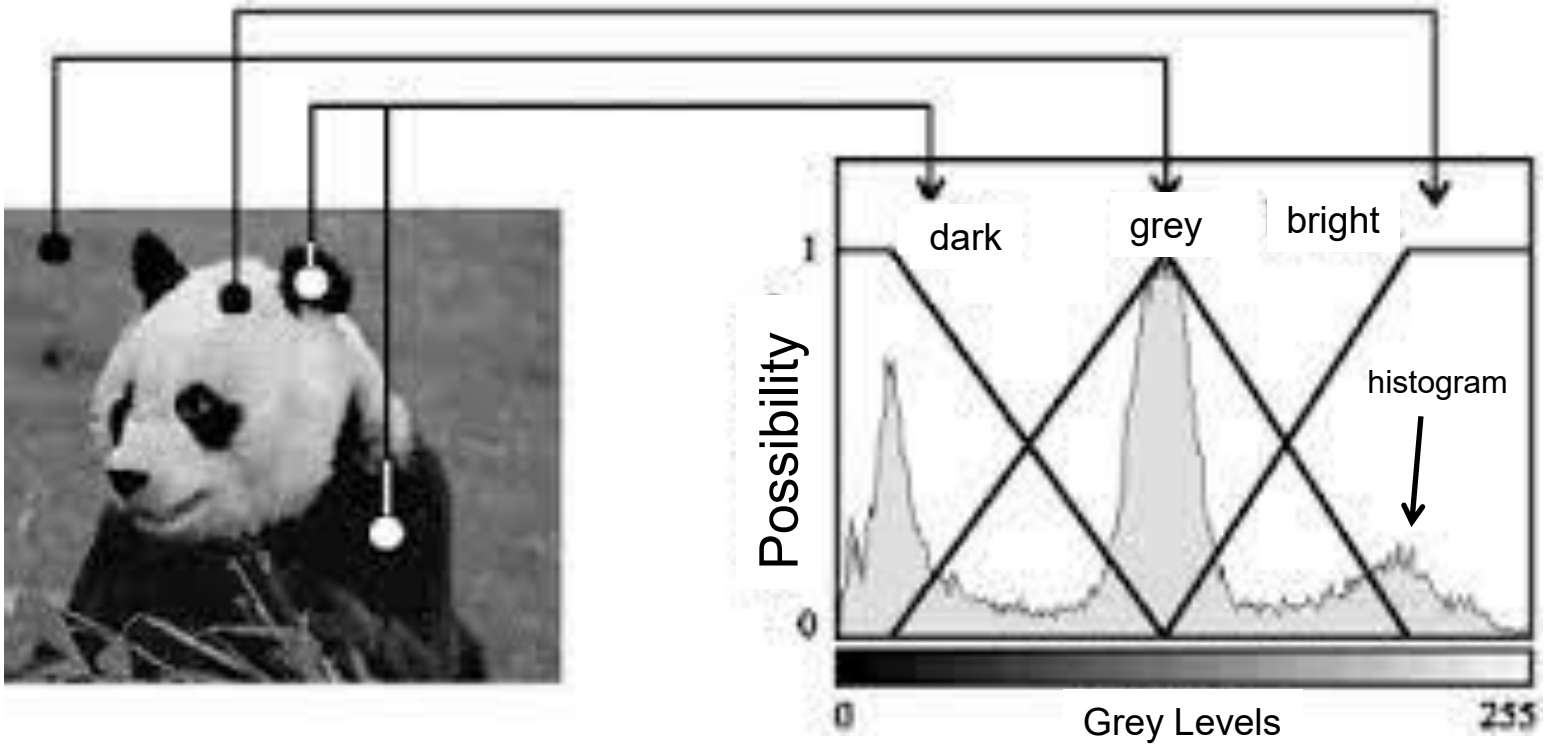
(b). Grouping of Pixels from the Trees



(c). Groupings of Pixels from the Road and the Sky

(d). Combined Result of Pixel Groupings

Example 2: Use of Intensities to Do Pixel Groupings



Crisp Values of Sensory Data

Outline of Lecture 2

- Use of Natural Languages
- Fuzziness of Natural Languages
- Concept of Belief's Fuzzy Sets
- Concept of Action's Fuzzy Sets
- Computation of Conceptual Knowledge
- Computation of Control Signals

Applications of Fuzzy Sets Associated with Beliefs and Actions …

AI's Pillar No. 3: knowledge to Knowledge

- To imitate human-like decision-making with fuzzy sensory data:
  - To transform crisp sensory data into fuzzy sets associated with beliefs
  - To apply fuzzy rules to determine fuzzy decisions
  - To transform fuzzy decisions into crisp decisions

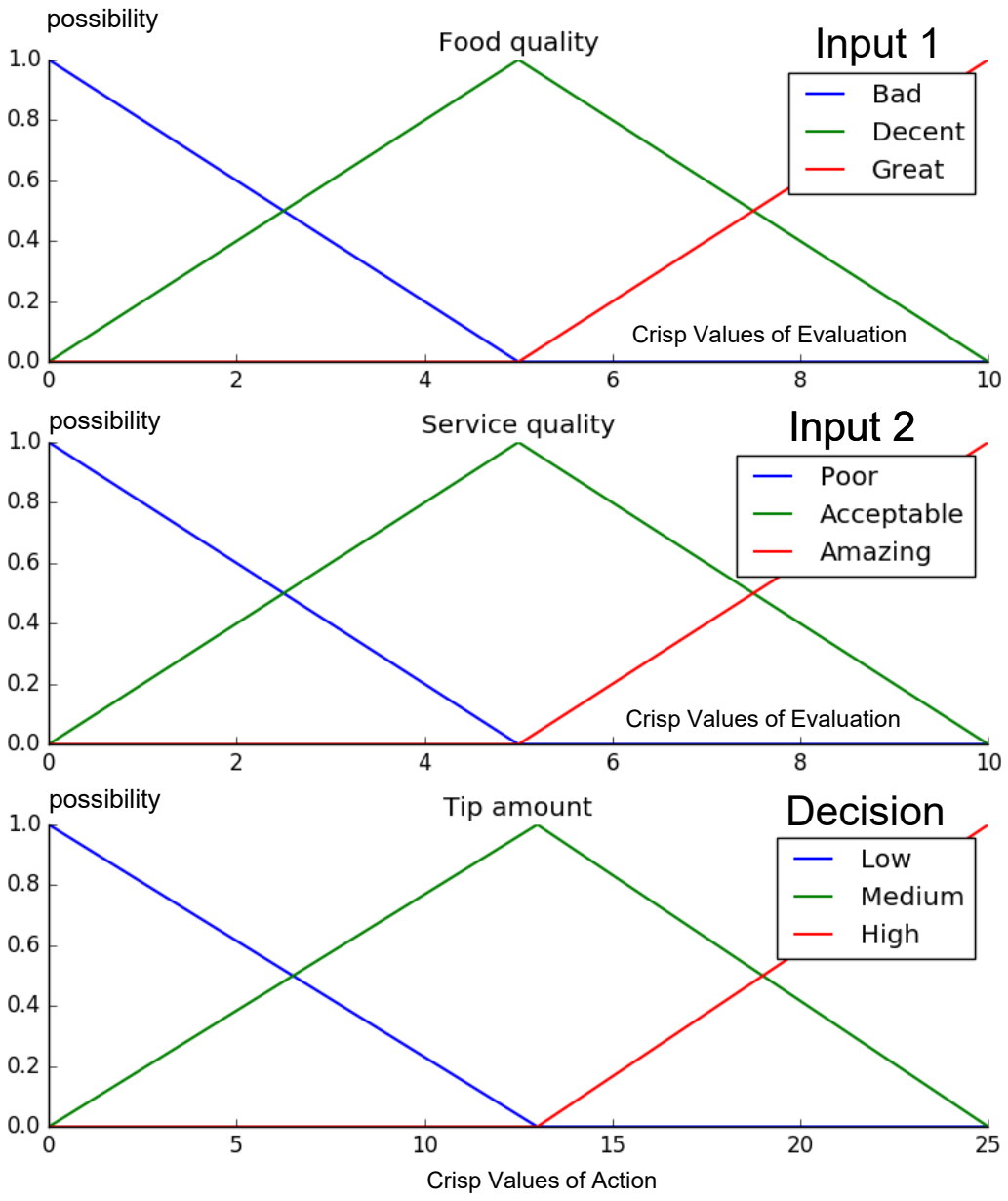
- To imitate human-like action-taking with fuzzy sensory data:
  - To transform crisp sensory data into fuzzy sets associated with beliefs
  - To apply fuzzy rules to determine fuzzy decisions of actions
  - To transform fuzzy decisions of actions into crisp values for action-taking or control

MIMO = Sum of MISO

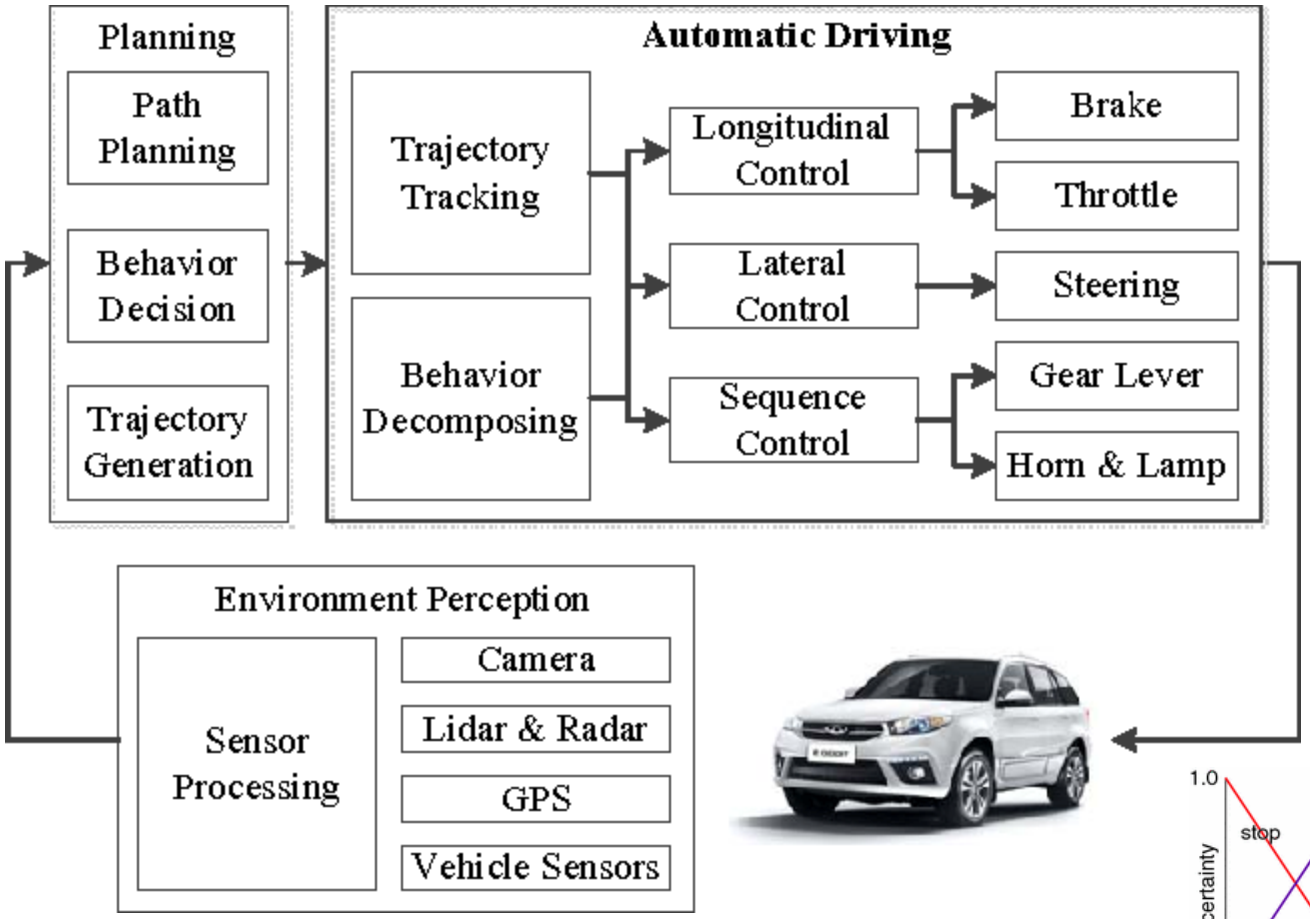
AI's Pillar No. 4: knowledge to Signal

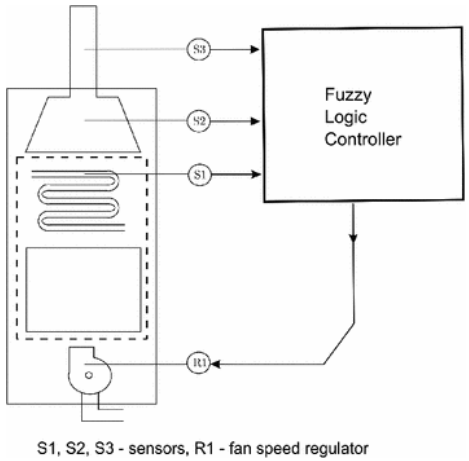
Example of Imitating Human-Like Decision-Making in Restaurant …

- Get evaluation of food quality
- Get evaluation of service quality
- Fuzzify both evaluation results
- If food quality is …, then tip is …
- If service quality is …, then tip is …
- Defuzzify the fuzzy decisions.
- Determine the crisp value of action.

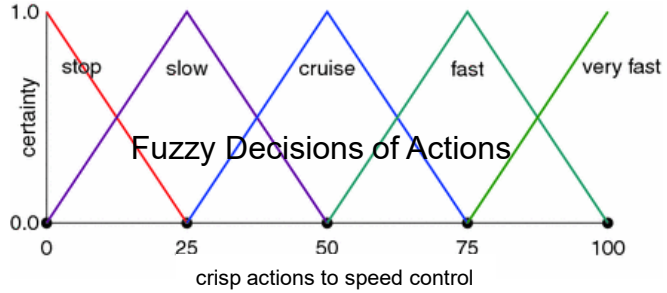


Examples of Implementing Fuzzy Controllers ...





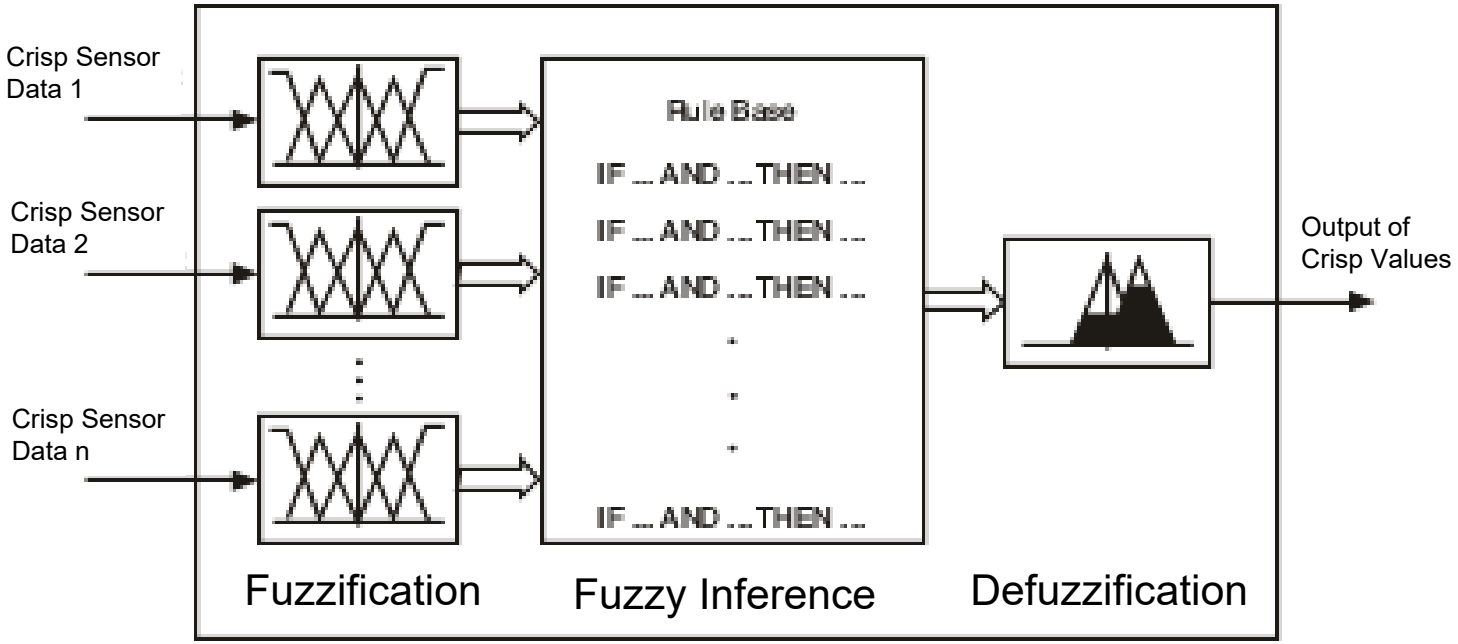
S1, S2, S3 - sensors, R1 - fan speed regulator



Fuzzy Decisions of Actions

Procedure of Implementing Fuzzy Controllers …

- Step 1: Transform crisp sensory data into fuzzy conditions (i.e., beliefs).
- Step 2: Transform fuzzy conditions into fuzzy decisions (i.e., also beliefs).
- Step 3: Transform fuzzy decisions into crisp values for action-taking.

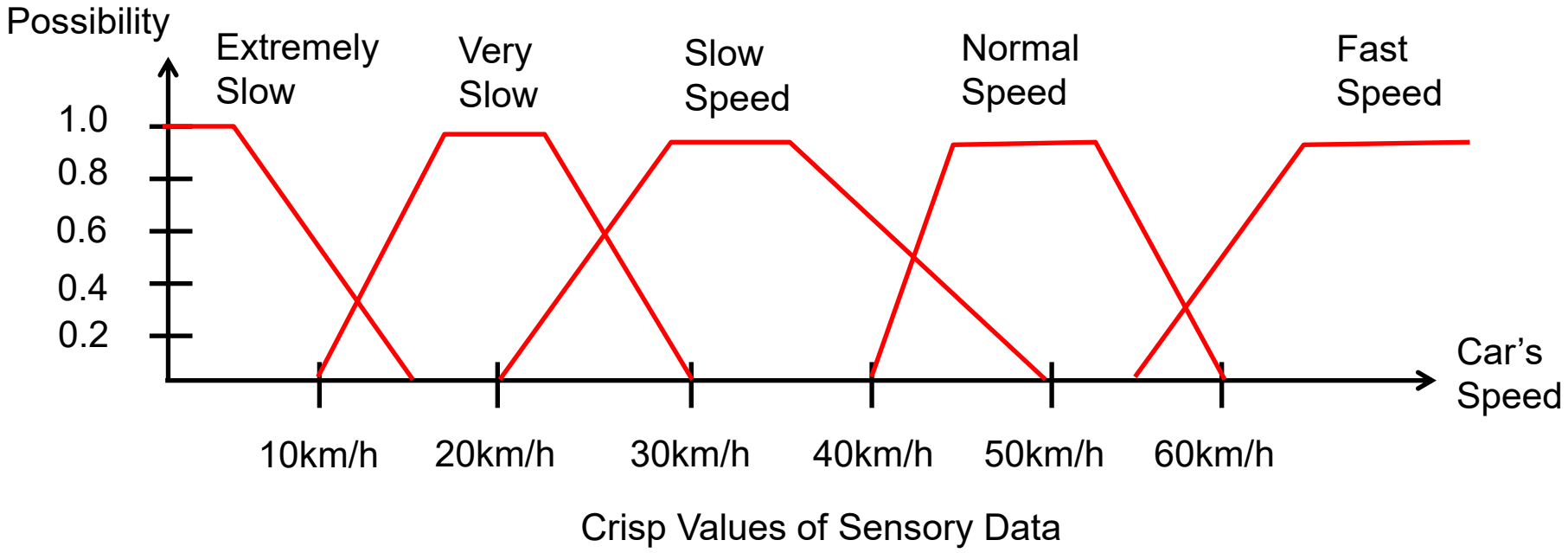


MIMO = Sum of MISO

Step 1: To convert crisp sensory data into fuzzy conditions (i.e., beliefs)

- To use fuzzy belief sets which include possibility functions

MIMO = Sum of MISO



Step 2: To transform fuzzy conditions into fuzzy decisions (i.e., beliefs)

- To use fuzzy rules such as: if fuzzy conditions, then fuzzy decisions of actions.

For example,

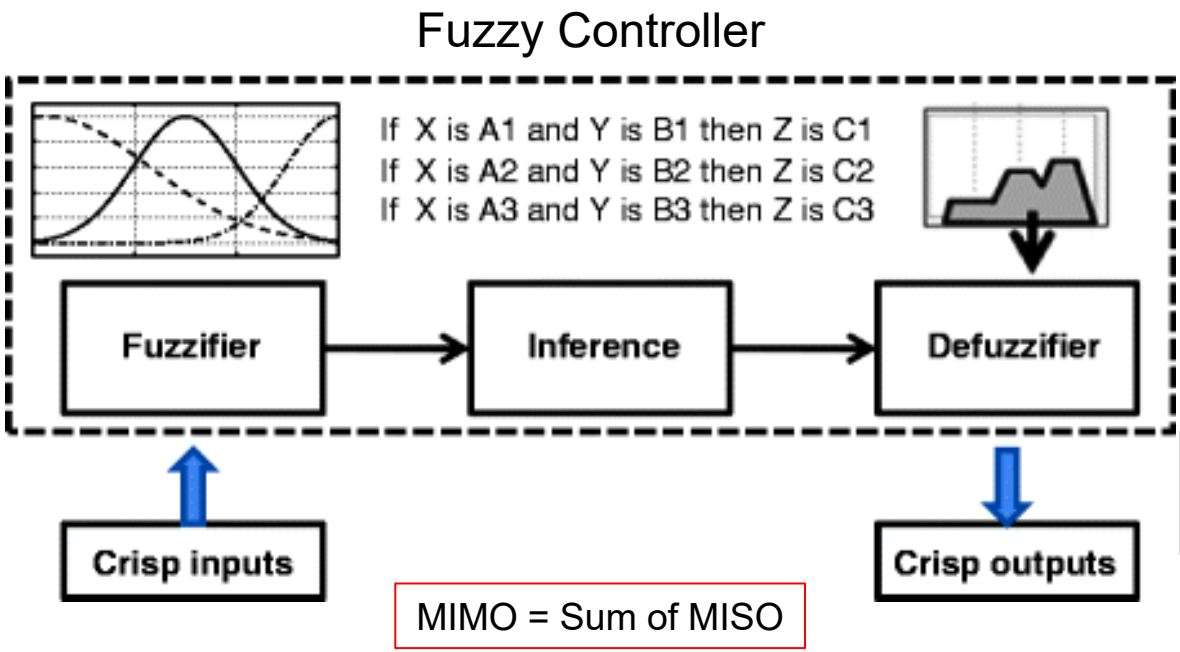
- If the speed of a car is too fast, strongly increase the braking force.
- If the temperature in a room is too warm, largely lower the value of temperature setting.
- If the distance to a frontal vehicle is too far, largely increase the speed of our vehicle.

More Examples to Illustrate Step 2 …

- If a car is moving too close to the left landmark, then rotate the steering wheel clockwise in a large amount of angle.
- If a car is following a fast-moving car in front, largely increase the distance to the leading car.
- If the weather becomes very foggy, largely reduce a car's speed.
- If error signal becomes too big, largely increase control signal.
- If the payload becomes too big, largely increase a car's output power.

Step 3: To transform fuzzy decisions into crisp values for action-taking

- To represent fuzzy decisions of actions.
- To compute the weighed sum of the possibility distributions of all the permissible fuzzy decisions of actions.
- To use the weighted sum as the output of crisp value.



$$V_o = \frac{\sum m(a_i)P(a_i)}{\sum P(a_i)}$$

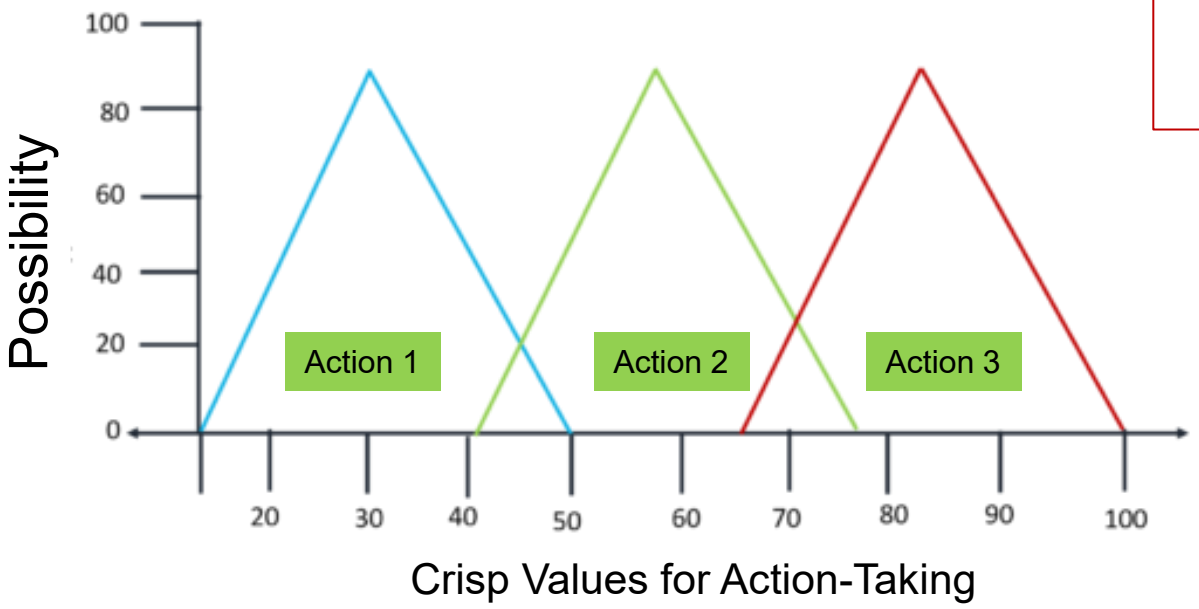
Weighted Sum of Fuzzy Actions

$m(a_i)$: Mean value of fuzzy decision of action i's possibility distribution.

$P(a_i)$: Possibility of making fuzzy decision of action i.

$$V_o = \frac{\sum m(a_i)P(a_i)}{\sum P(a_i)}$$

Fuzzy Action
=
Fuzzy Decision
+
Crisp Value

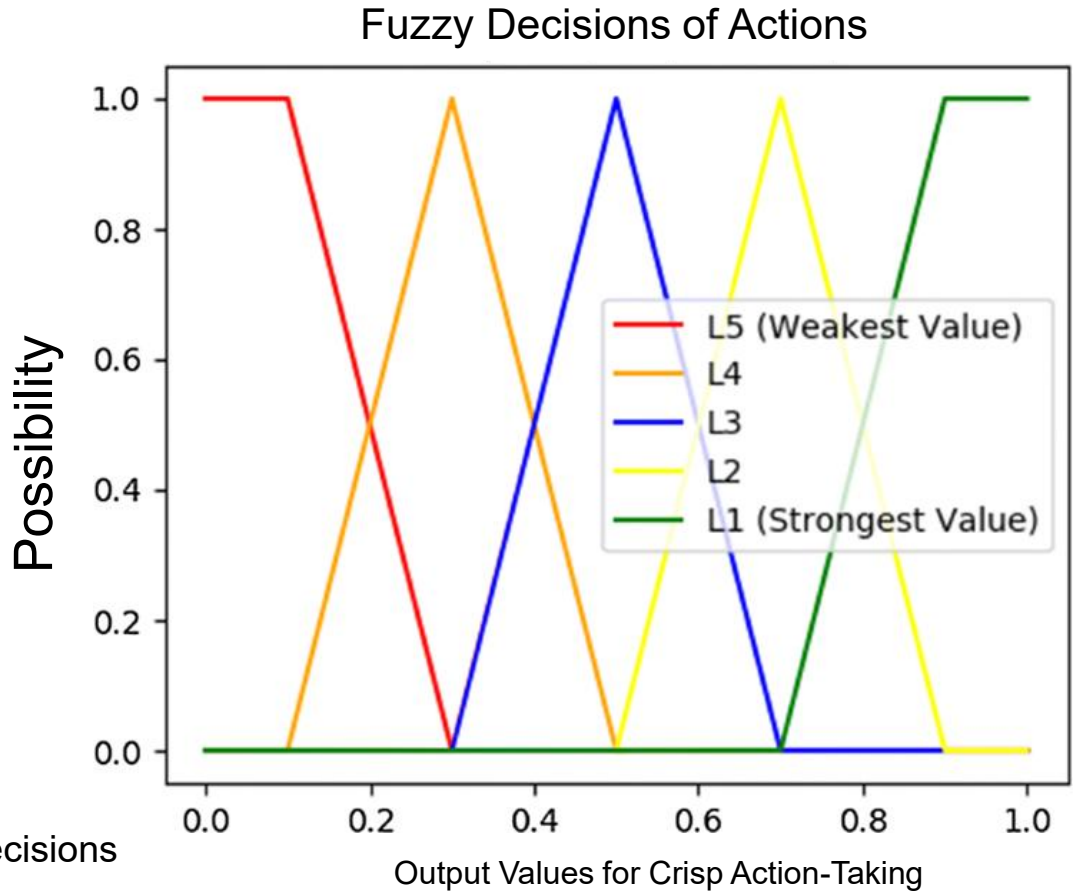


Representation of Fuzzy Decisions of Actions ...

- To associate a full-scale possibility function with each possible decision of action, which is also a fuzzy belief.

Fuzzy Decisions of:
- Action L1
- Action L2
- Action L3
- Action L4
- Action L5

MIMO = Sum of MISO



A single sensory data may trigger multiple decisions

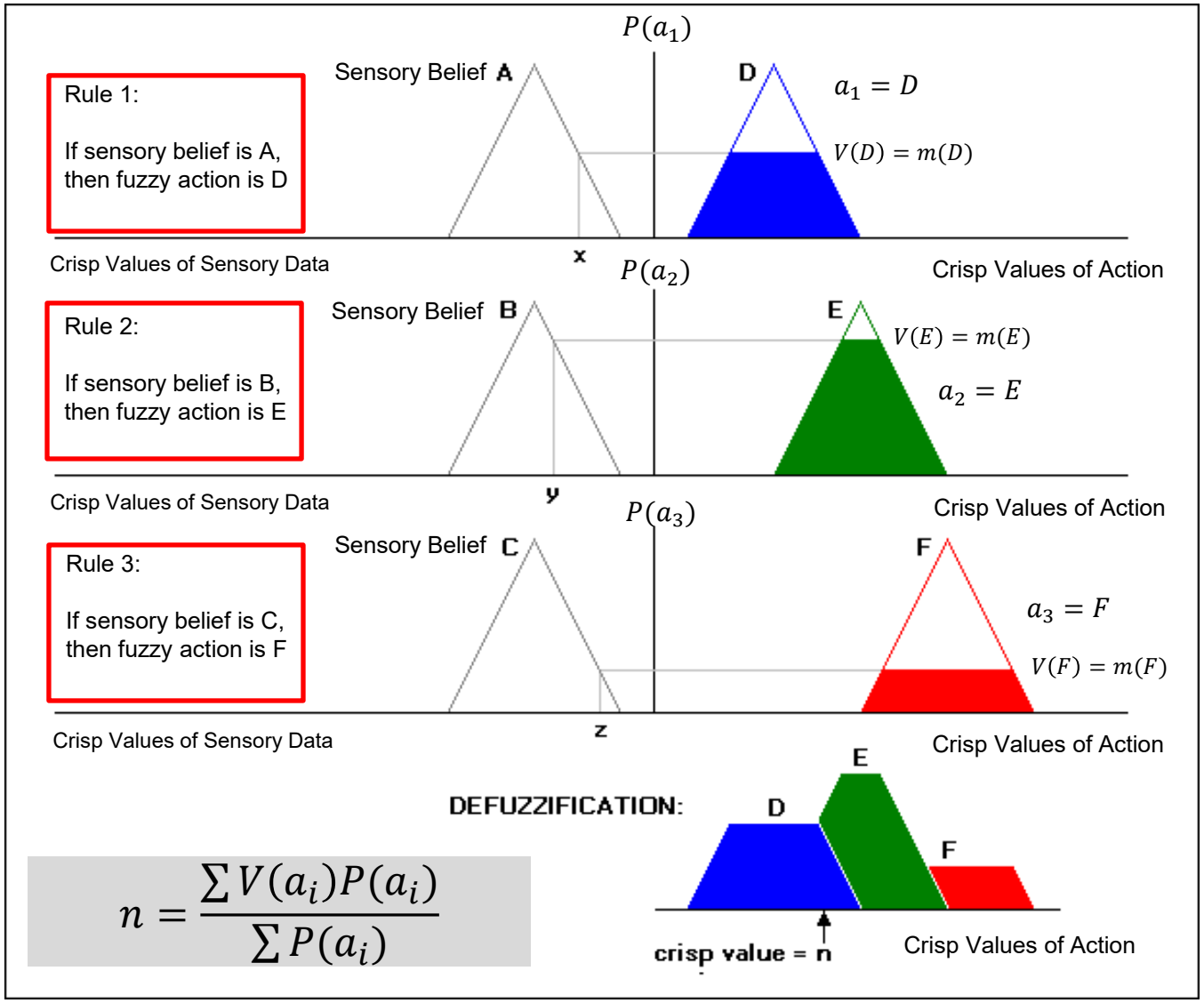
Example of Computing Weighted Sum …

Sensor Data x triggers action D

Sensor Data y triggers action E

Sensor Data z triggers action F

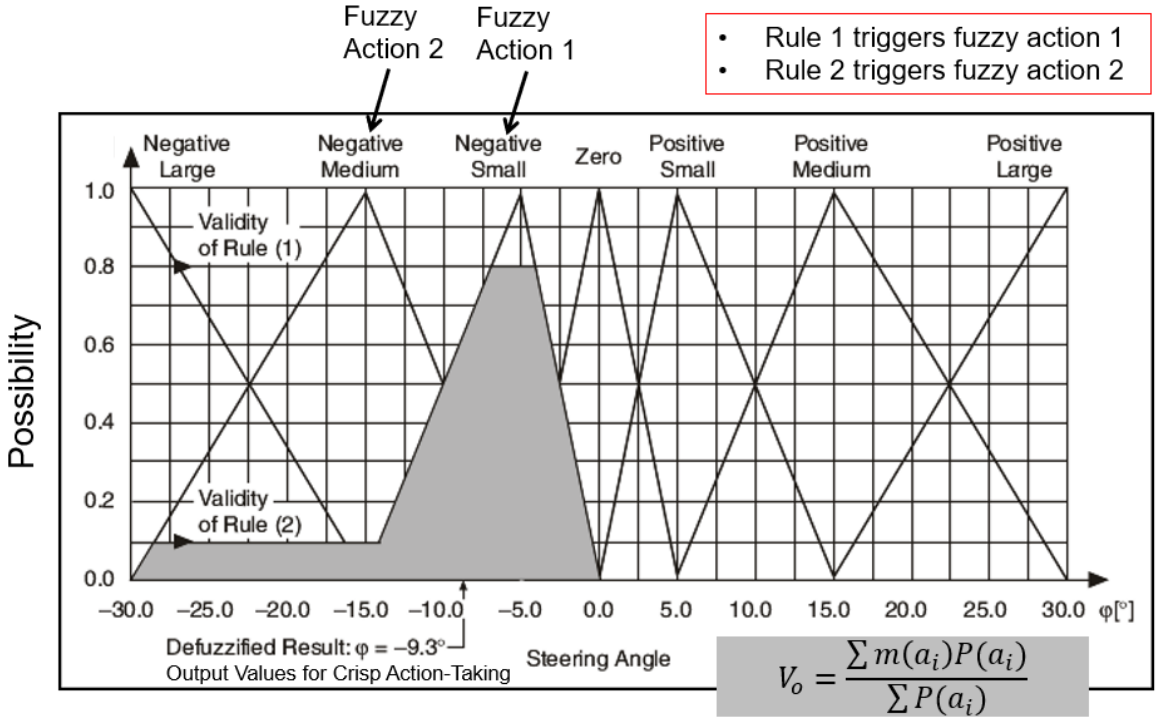
Fuzzy Action
=
Fuzzy Decision
+
Crisp Value



$$n = \frac{\sum V(a_i)P(a_i)}{\sum P(a_i)}$$

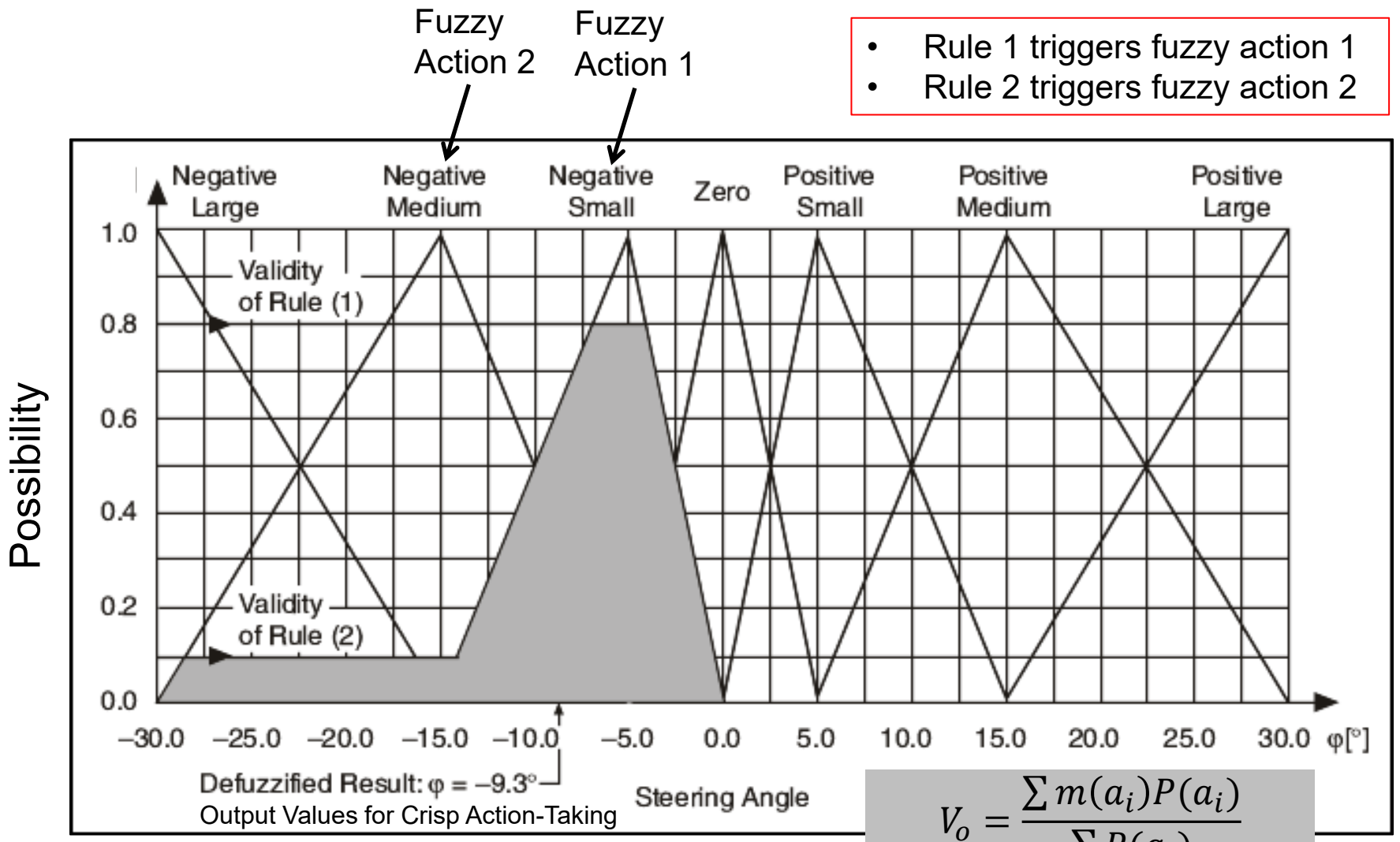
Another Example of Computing Weighted Sum …

- Inside a smart car, a fuzzy sensory belief triggers action 1 with the possibility value of 0.8 while a fuzzy sensory belief triggers action 2 with the possibility value of 0.1. If the mean value of fuzzy action 1's possibility distribution is -8.2 degrees and the mean value of fuzzy action 2's possibility distribution is -18.2 degrees, what should be the crisp value of steering angle?



Answer: $V_0 = (-18.2 * 0.1 - 8.2 * 0.8)/0.9 = -9.31$

- Rule 1 triggers fuzzy action 1
- Rule 2 triggers fuzzy action 2

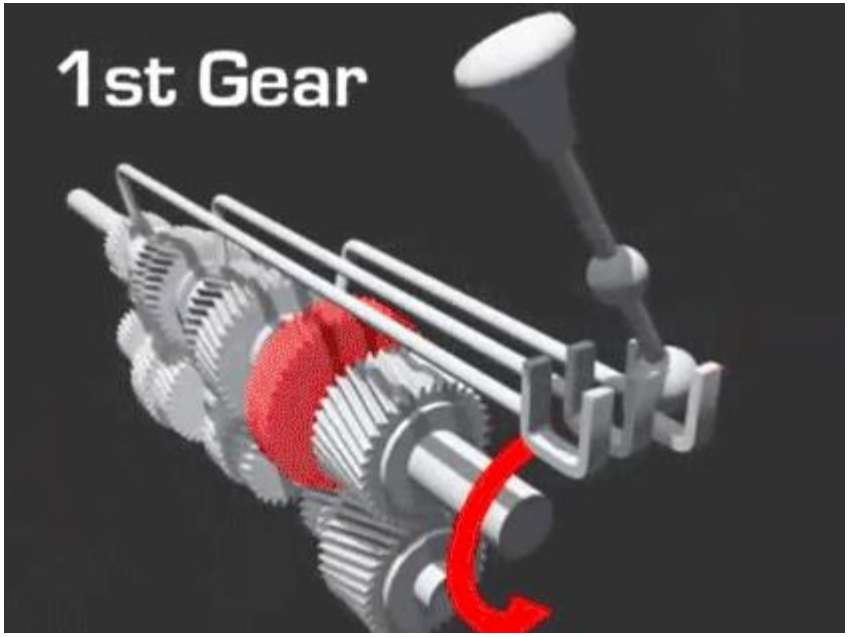


$$V_o = \frac{\sum m(a_i)P(a_i)}{\sum P(a_i)}$$

One More Example …

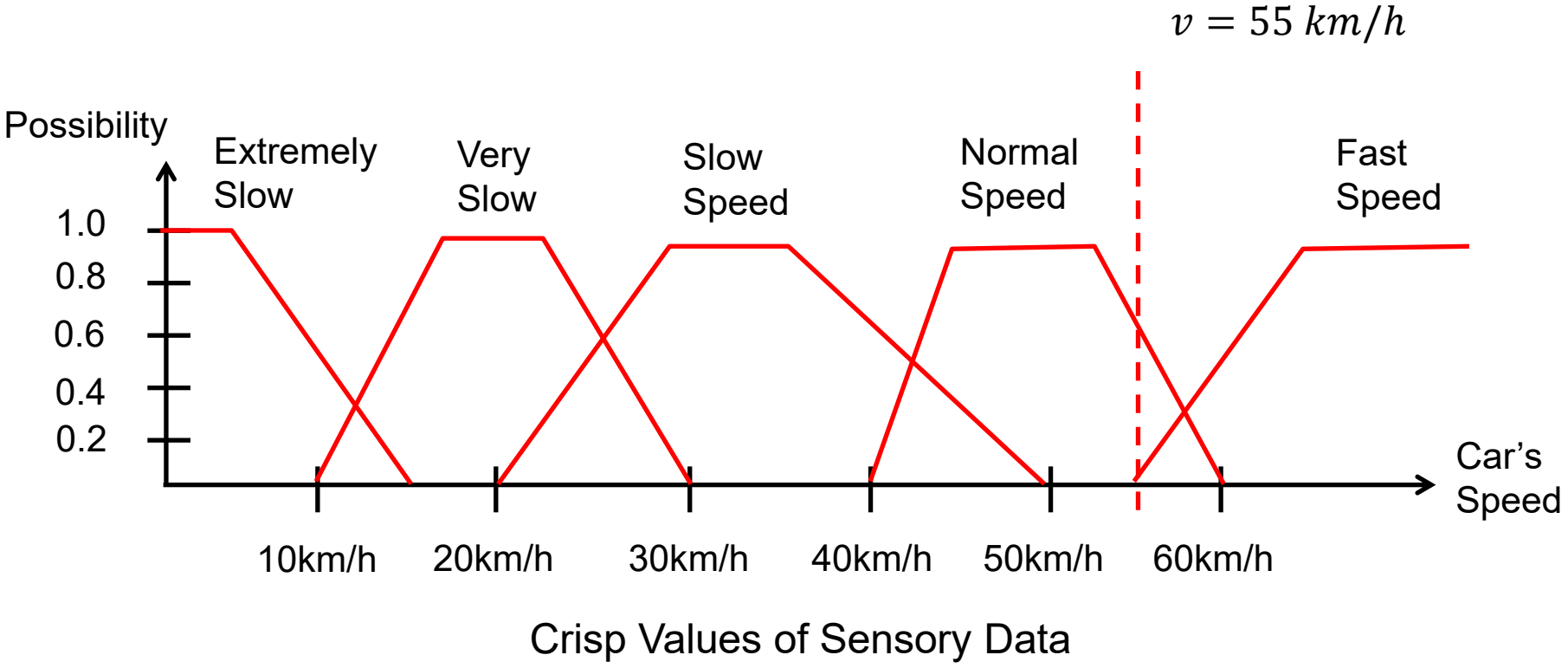
- You works in an automobile industry. You are responsible of designing a fuzzy controller for automatic gearboxes with five speeds or gear ratios.

- What will be your design solution?



Question

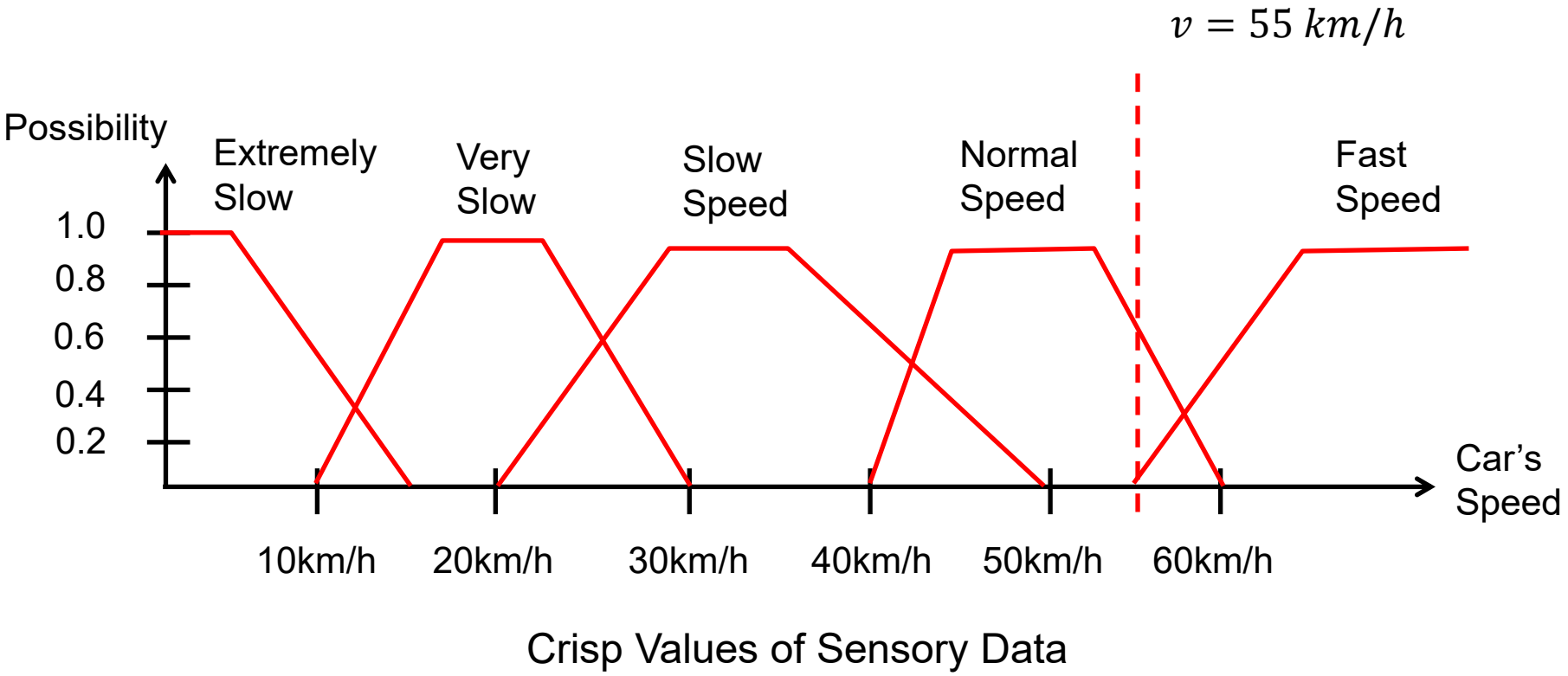
- If the car's speed is 55 km/h, what should be the gearbox's ratio?



Answer

- Step 1: Do Fuzzification

The sensor's output is 55 km/h. The car is at normal speed.



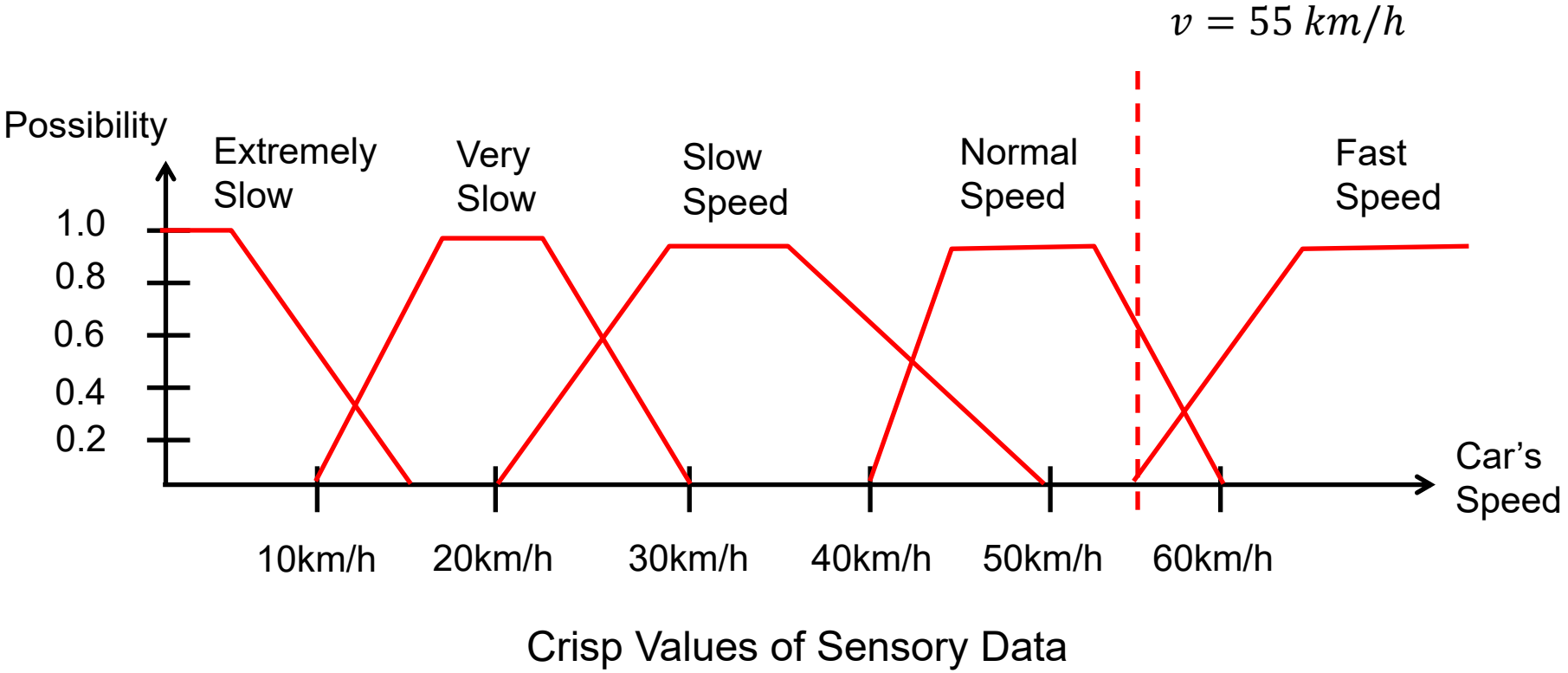
Answer (continued)

- Step 2: Apply Fuzzy Inference Rules …
  - If a car's speed is extremely slow, choose speed gear ratio 1.
  - If a car's speed is very slow, choose speed gear ratio 2.
  - If a car's speed is slow, choose speed gear ratio 3.
  - If a car's speed is normal, choose speed gear ratio 4
  - If a car's speed is fast, choose speed gear ratio 5.

Answer (continued)

- Step 3: Output Knowledge for Action-Taking

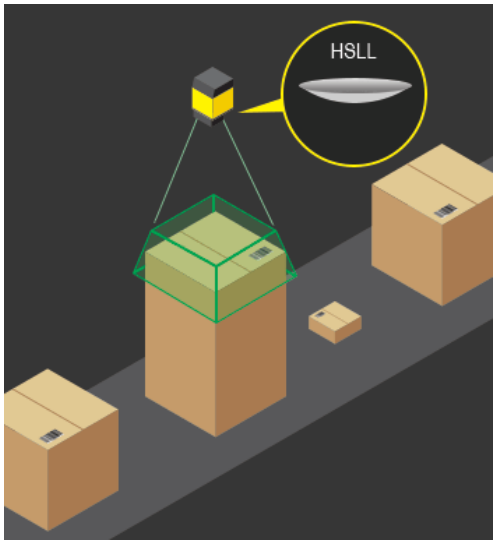
If a car's speed is normal, set speed gearbox ratio to 4.



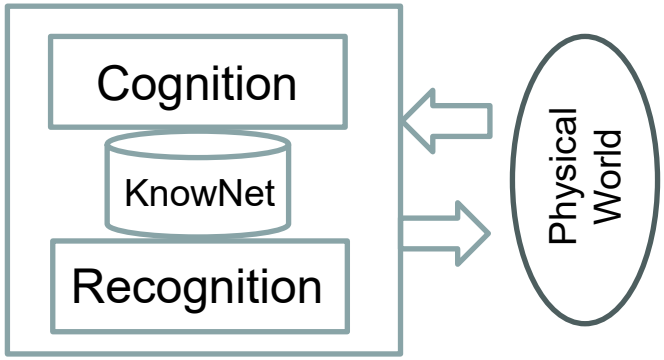
Summary of Lecture 2

- Use of Natural Languages
- Fuzziness of Natural Languages
- Concept of Belief's Fuzzy Sets
- Concept of Action's Fuzzy Sets
- Computation of Conceptual Knowledge
- Computation of Control Signals

Outline of Module 4



- Lecture 1:
  - Geometry-Driven Computation
- Lecture 2:
  - Fuzziness-Driven Computation
- Lecture 3:
  - Cognition-Driven Computation
- Lecture 4:
  - Recognition-Driven Computation
- Lecture 5:
  - Computation Using Monocular Vision
- Lecture 6:
  - Computation Using Binocular Vision



NANYANG TECHNOLOGICAL UNIVERSITY

School of Mechanical & Aerospace Engineering

Design, Machine, Control, Intelligence

Lecture 3 of Module 4

AI 3.0

MA4829 Machine Intelligence

Cognition-Driven Computation

XIE Ming, PhD (France)

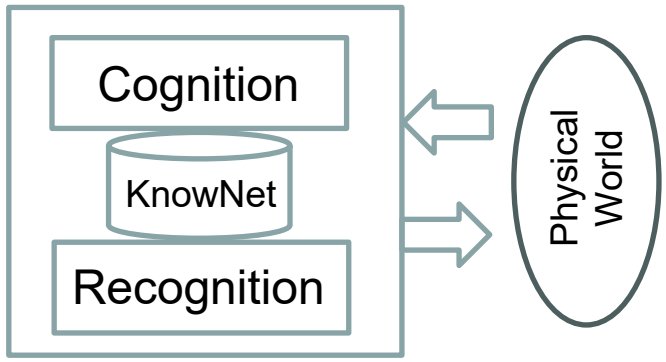
http://personal.ntu.edu.sg/mmxie

"We are living inside an ocean of signals"

What is a system with intelligence inside?

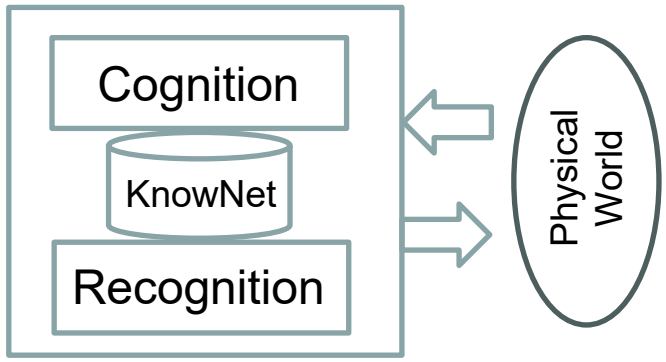
Outline of Lecture 3

- Principle of Cognition (Learning)
- Principle of Artificial Neural Network
- Principle of RCE Neural Network
- Cognition of Colors
- Cognition of Curves
- Cognition of Pattern



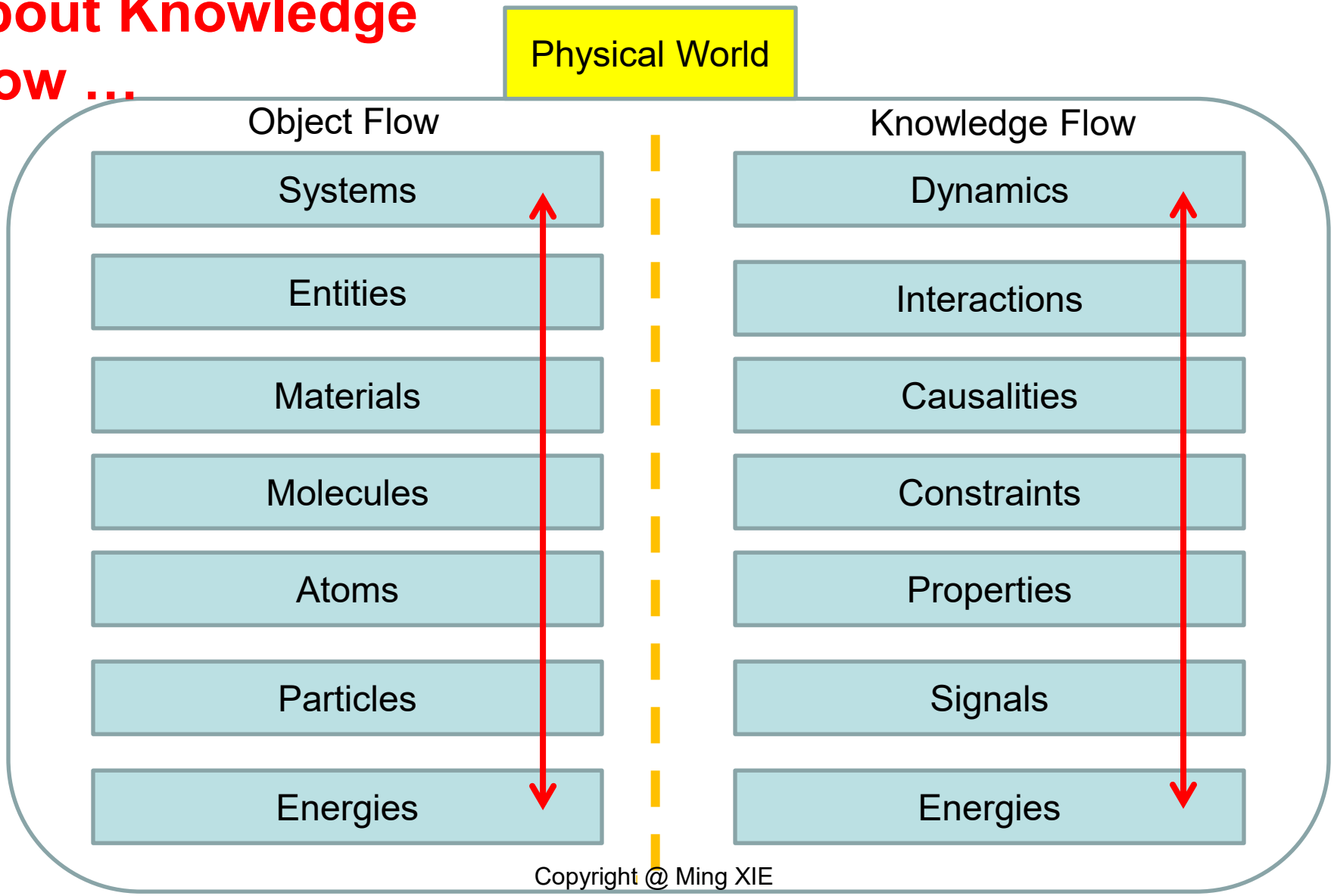
Outline of Lecture 3

- Principle of Cognition (Learning)
- Principle of Artificial Neural Network
- Principle of RCE Neural Network
- Cognition of Colors
- Cognition of Curves
- Cognition of Pattern



Cognition enables autonomy …

About Knowledge Flow …

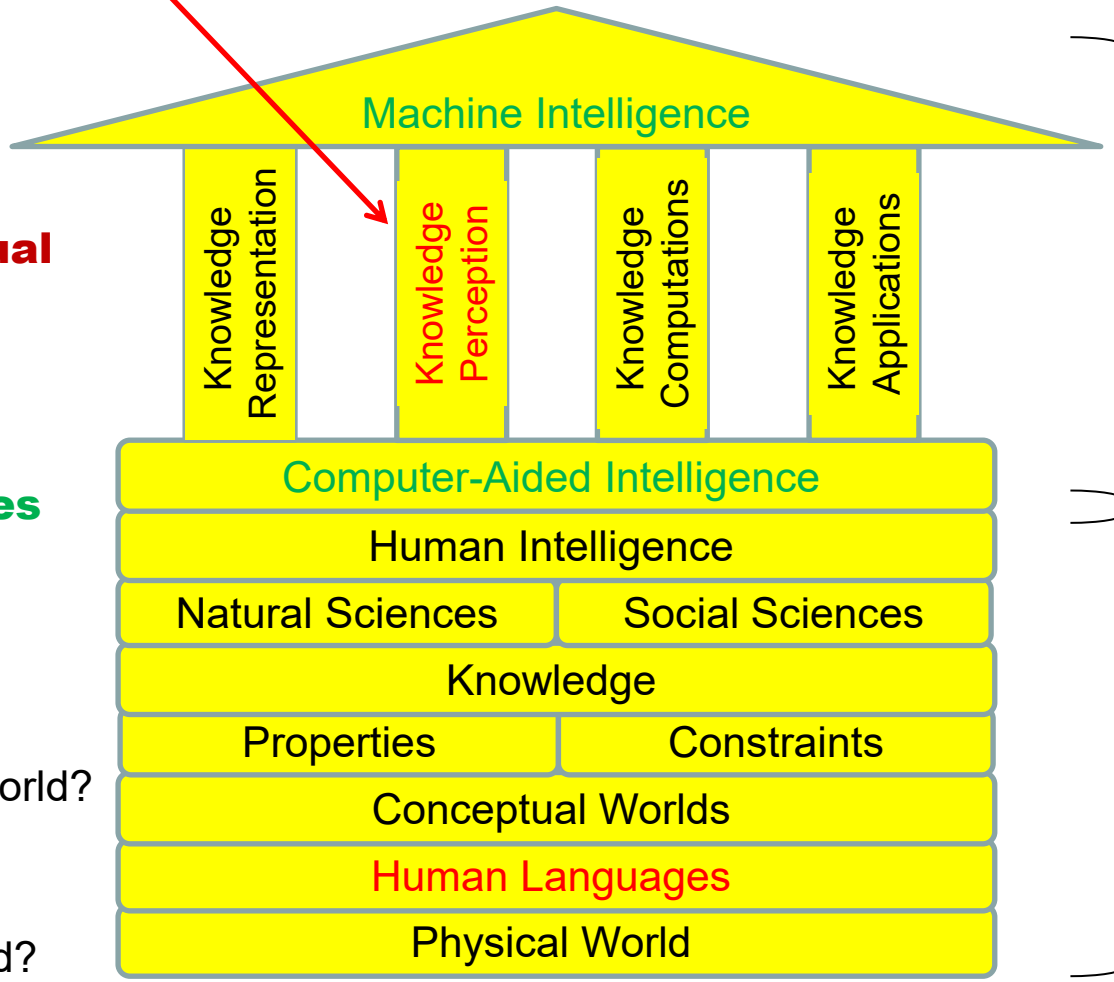


How to cognize or learn conceptual knowledge?

Answer:

The mental capability of transforming visual signals into knowledge.

Such mental capability includes visual cognition and recognition.



What is conceptual world?

?

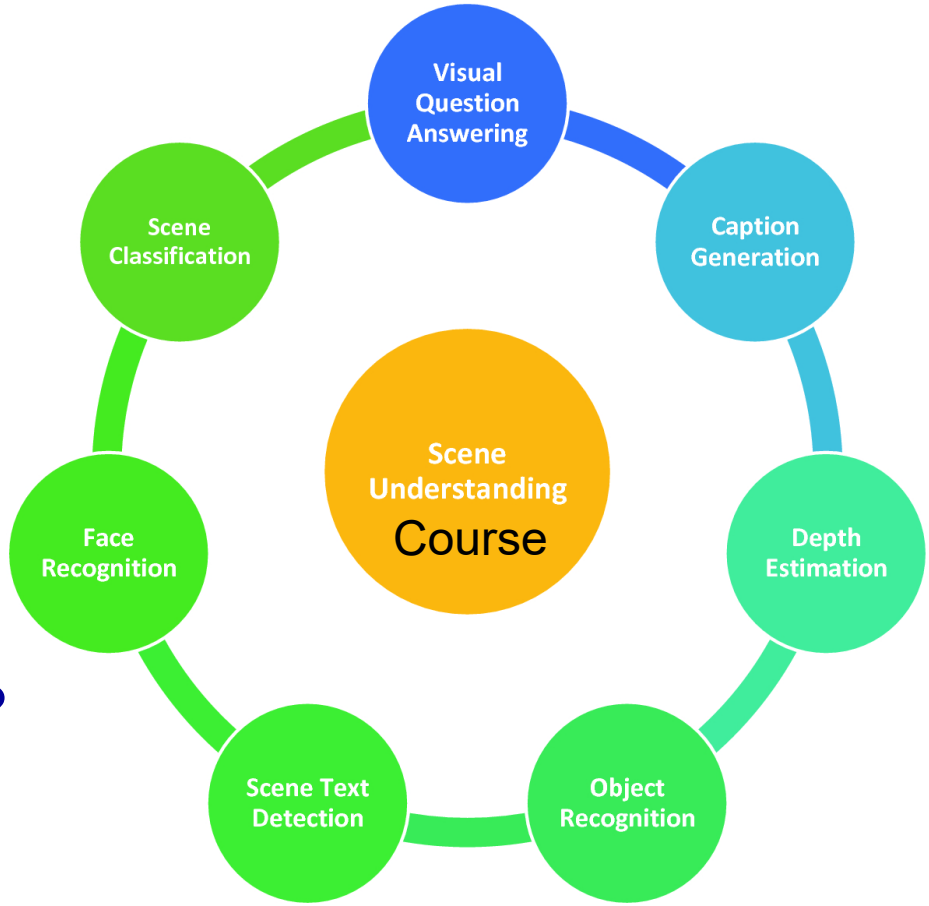
What is physical world?

Discussion 1 (warm up …)

- Could learning be unsupervised?
- Is unsupervised computation equal to unsupervised learning?
- Could supervision come from oneself?
- What are the roles of teachers?

Discussion 2 (warm up …)

- Could learning be statistical (i.e. randomness)?
- Are curriculums well organized in schools?
- Is statistical computation equal to statistical learning?



Discussion 3 (warm up …)

- Could learning be deep, or be not deep?
- Is recursive computation equal to deep learning?
- Learning is an incremental process, isn't it?

Discussion 4 (warm up ...)

- Why does a student spend many years to do supervised learning in schools and universities?

Scenario of Visual Cognition

Cognition

Teacher: This is a blue color.
Machine: Got it. It is a blue color.

Re-cognition

Teacher: What is this color?
Machine: It is a blue color.

What is cognition in general?

- Cognition is to cognize or learn anything new or anything which is not known before. Cognition or learning has to be supervised.

- What is this place?
- What is the building in the foreground?
- What is the building in the middle of the background?

Definition of Cognition

Definition

- It is a mental process of cumulatively learning new knowledge about new entities or new knowledge about existing entities.

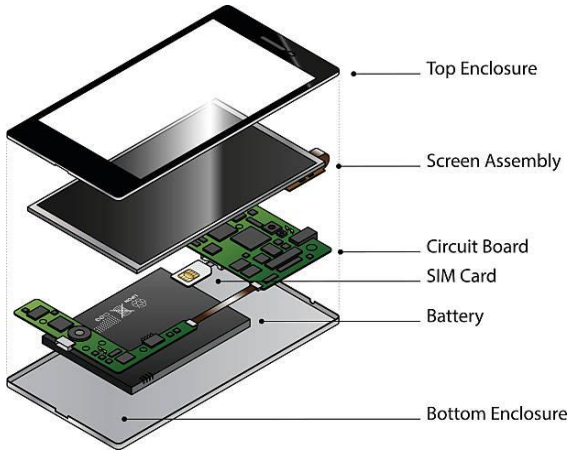
Similar Terms

- Supervised Learning
- Supervised Training
- Education
- Discovery

What to cognize from visual signals?

- Photometric Knowledge of Entities from the Physical World
  - Luminance-related Appearances
  - Chrominance-related Appearances

- Geometrical Knowledge of Entities from the Physical World
  - Points
  - Lines
  - Curves
  - Shapes
  - Surfaces
  - Objects
  - Exploded Drawings, etc.



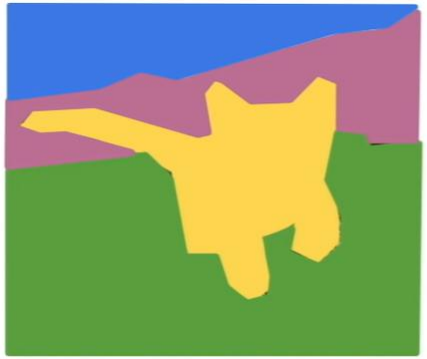
Terminology Alerts

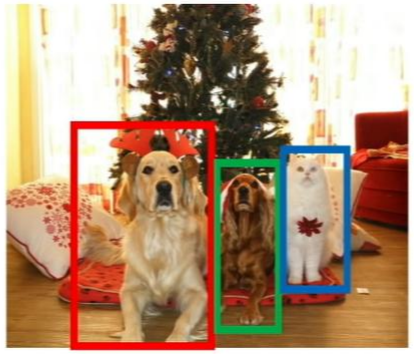
- Objects
  - Objects are the entities inside the physical world.
- Images
  - Images are the projections of scenes onto cameras.
- Colors
  - Colors are the coordinates inside a color space.
- Curves
  - Curves are the linked edges inside a digital image.
- Patterns
  - Patterns are the shapes or outlooks of individual objects.

How to cognize new and meaningful features from visual signals?

- Step 1: Detection of New and Meaningful Features
- Step 2: Grouping of New and Meaningful Features into Networks
- Step 3: Supervised Training of Feature Networks

Our Research Around 1993



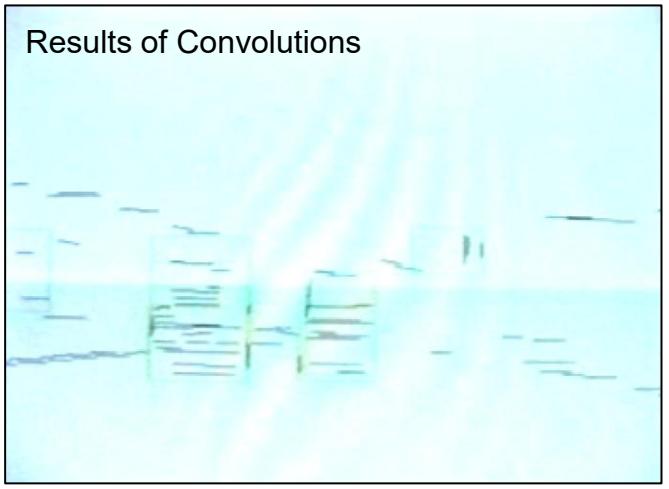


YOLO Algorithm Since 2015

How to detect new and meaningful features from digital images?

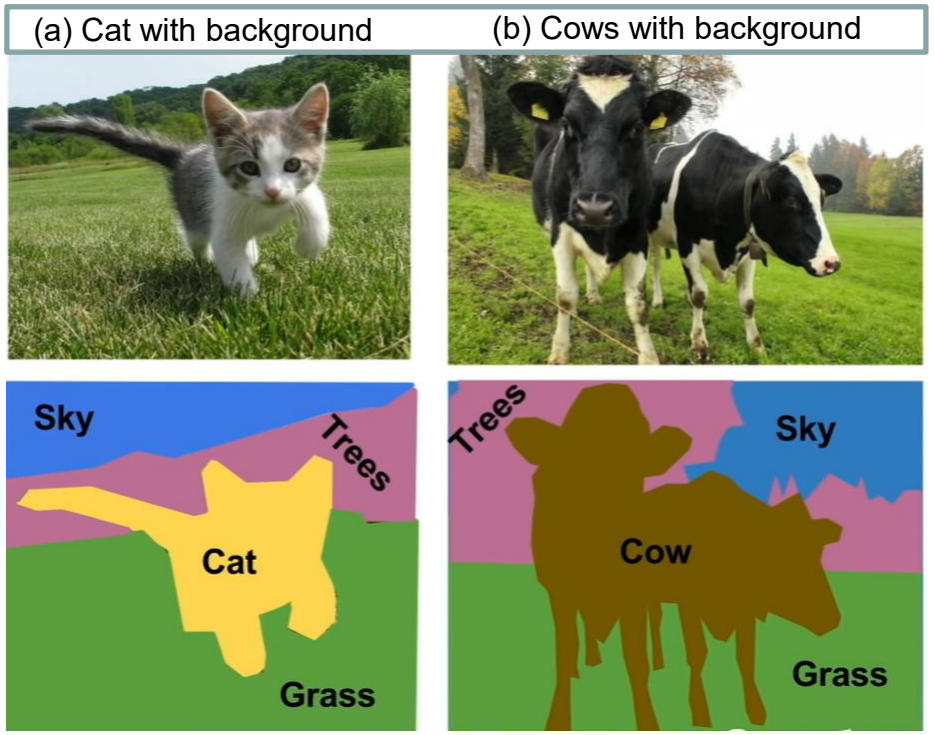
- Region-Based Detection and Grouping
  - Arithmetic and statistical computation of continuities in digital images
- Edge-Based Detection and Grouping
  - Arithmetic and statistical computation of discontinuities in digital images
- Texture-Based Detection and Grouping
  - Arithmetic and statistical computation of periodic transitions in digital images
- Pattern-Based Detection and Grouping
  - Arithmetic and statistical computation of occurrences of shapes or outlooks

Our Research Around 1993



What is statistical computation from digital images?

- To compute statistics in the forms of:
  - Histogram of Intensities
  - Histogram of Colors
  - Histogram of Gradients
  - Histogram of Textures (i.e. repeated mini-patterns), etc.

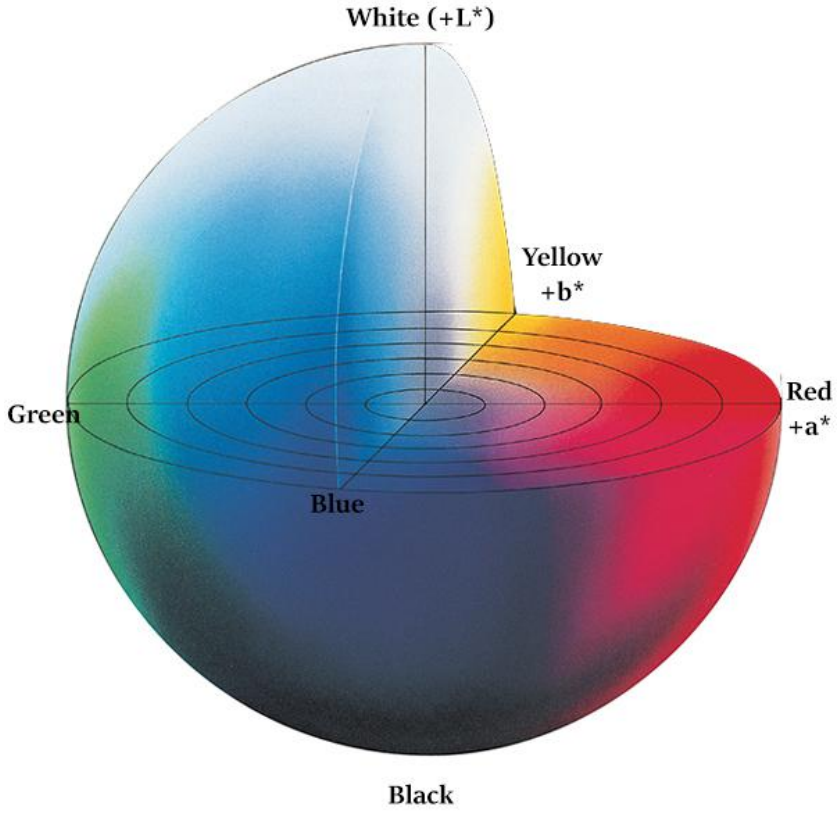


Foreground = Image - Background

How to represent new and meaningful features detected from images?

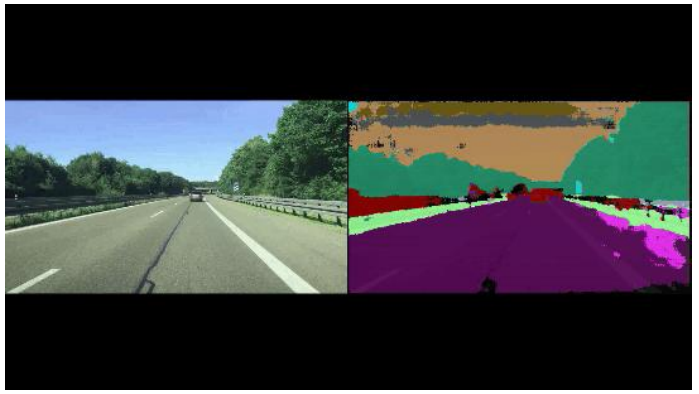
- To use feature vectors which define feature spaces.

- Examples:
  - Color Space
  - Line Space
  - Curve Space
  - Shape Space
  - Surface Space
  - Texture Space, etc



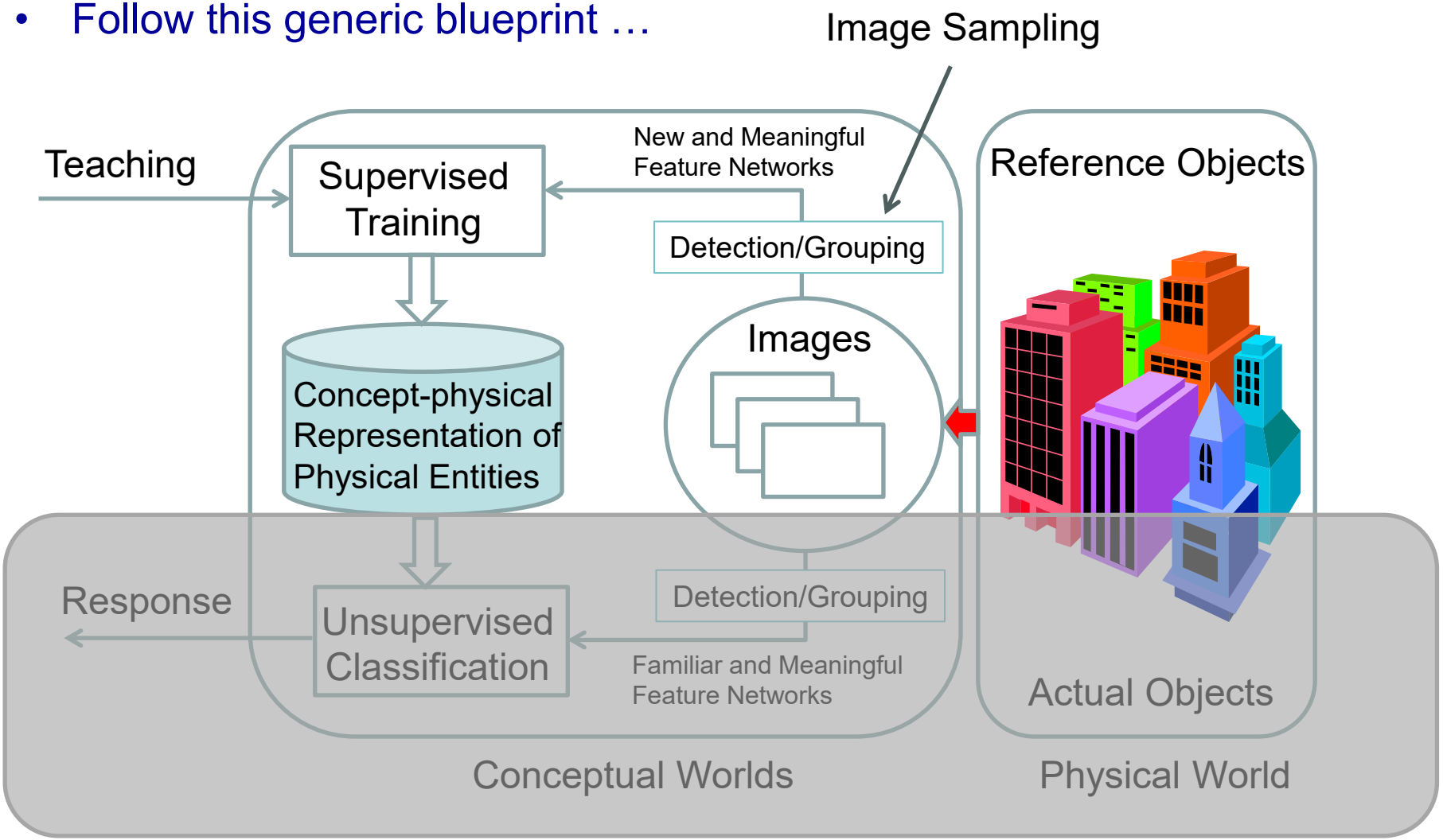
What is feature (i.e. meaningful data) network (i.e. interconnection)?

- A digital image is originally a set of unrelated pixels.
- A feature is a meaningful data inside a digital image.
- A feature network is a set of similar features which are interconnected together.
- Examples:
  - A network of interconnected pixels of a similar color
  - A network of interconnected edges
  - A network of interconnected textures (e.g. textures of background)
  - A network of interconnected patterns (e.g. patterns inside a face)



How to Design Visual Cognition Systems?

- Follow this generic blueprint …



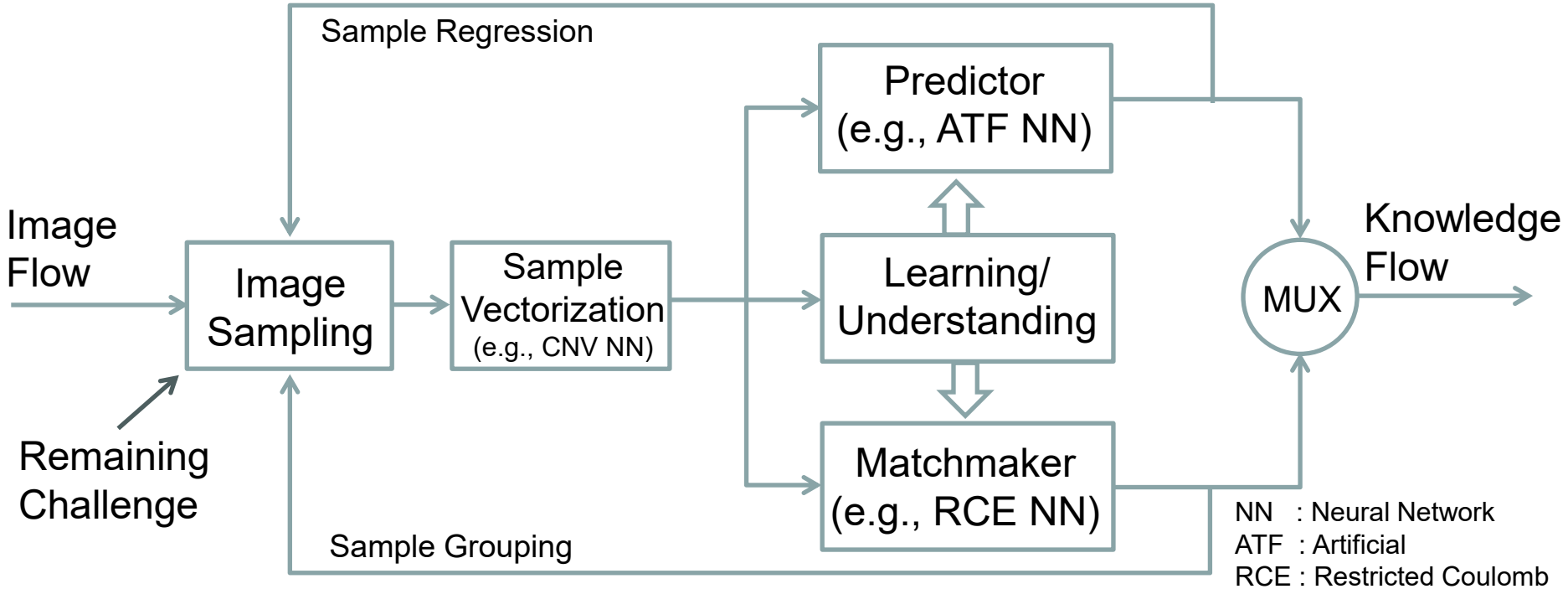
What are solutions underlying cognition of …

- Colors
- Curves
- Patterns
- Products
- etc …




Answer: to connect physical world to conceptual world (Feature Extraction → Symbol Grounding)

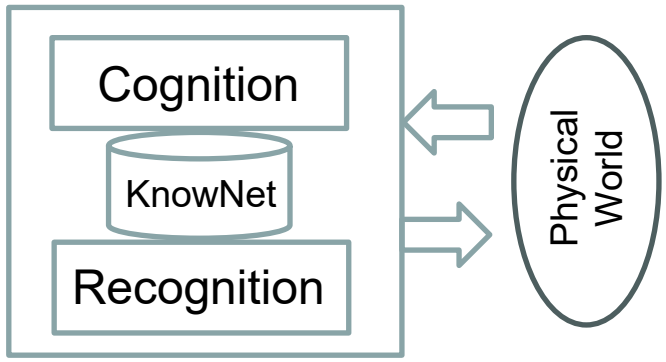
- Key Steps Involve Use of 2D-Model-Based Cognition and Recognition:
  - Pattern/Object Detection Followed by Categorization
  - Pattern/Object Detection Following by Identification



Note: We advocate KnowNet which is a knowledge network.

Outline of Lecture 3

- Principle of Cognition (Learning)
- Principle of Artificial Neural Network
- Principle of RCE Neural Network
- Cognition of Colors
- Cognition of Curves
- Cognition of Pattern

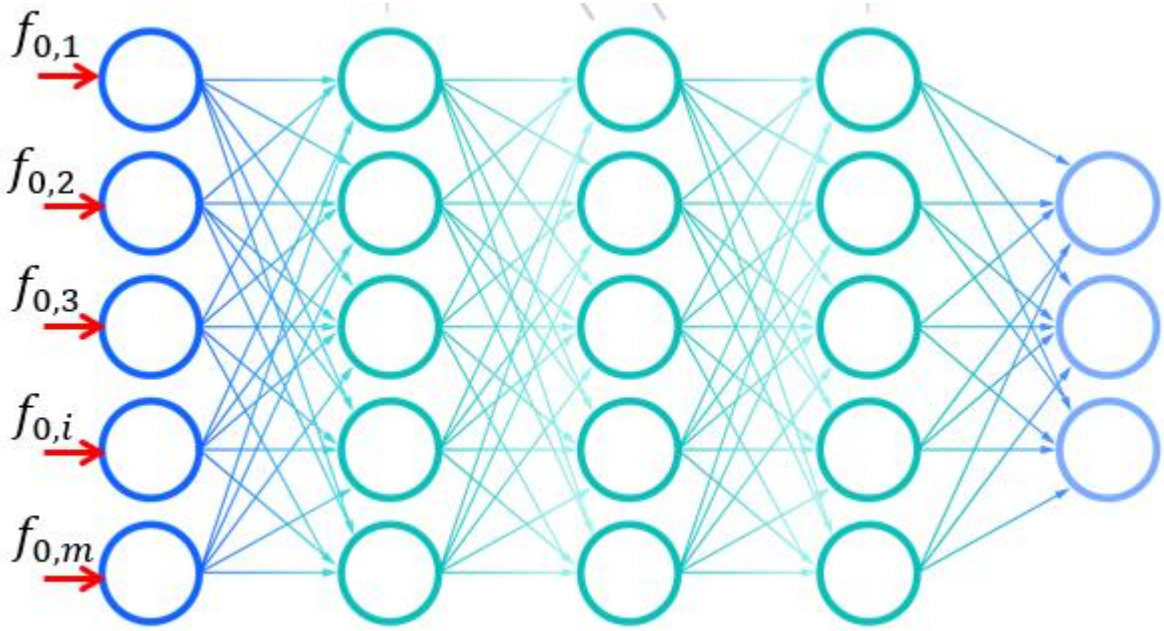


History of Artificial Neural Network (ANN)

- **Warren McCulloch and Walter Pitts (1943):** Often cited as the original inventors of the first computational model for neural networks. They published "A Logical Calculus of the Ideas Immanent in Nervous Activity," which proposed a simplified mathematical model of a biological neuron (the **McCulloch-Pitts neuron**) capable of performing logical functions.
- Frank Rosenblatt[1] (1958) created the perceptron, an algorithm for pattern recognition.
- The first deep learning multilayer perceptron trained by stochastic gradient descent was published in 1967 by Shun'ichi Amari.
- Backpropagation is an efficient application of the chain rule derived by Gottfried Wilhelm Leibniz in 1673.
- The origin of the Convolutional Neural Network (CNN) architecture is the "neocognitron" introduced by Kunihiko Fukushima in 1980.

Definition of Artificial Neural Network

- An artificial neural network is an associative memory of values for the representation of a system of linear equations.



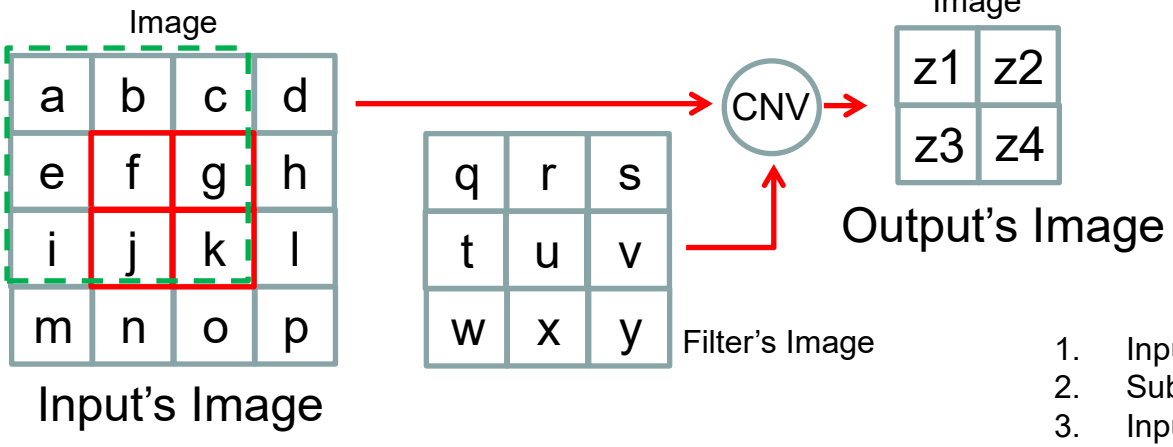
Property 1 of Artificial Neural Network

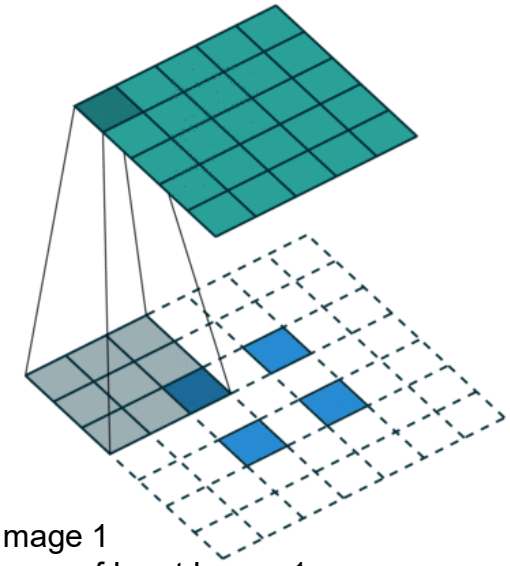
- An artificial neural network is a solution for the implementation of the convolution between an input and a filter.

- What is a filter?
  - A filter is a selector of feature or knowledge.
  - A filter is better to be the outcome of top-down design.
  - A filter could be the outcome of bottom-up optimization.

Example of Image Convolution

- Convolutional Neural Network (i.e. CNN)

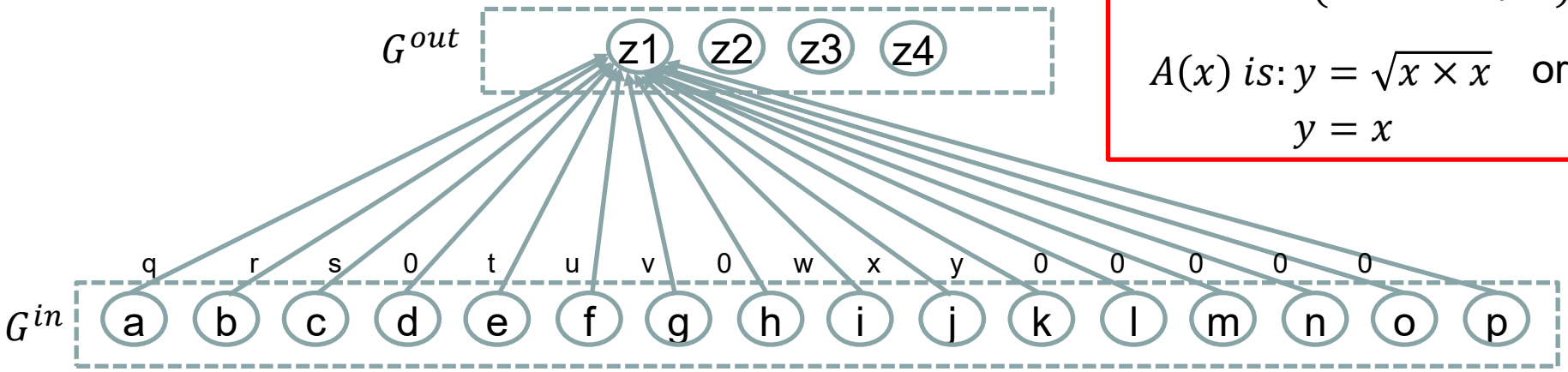




1. Input Image 1
2. Sub-image of Input Image 1
3. Input Image 2

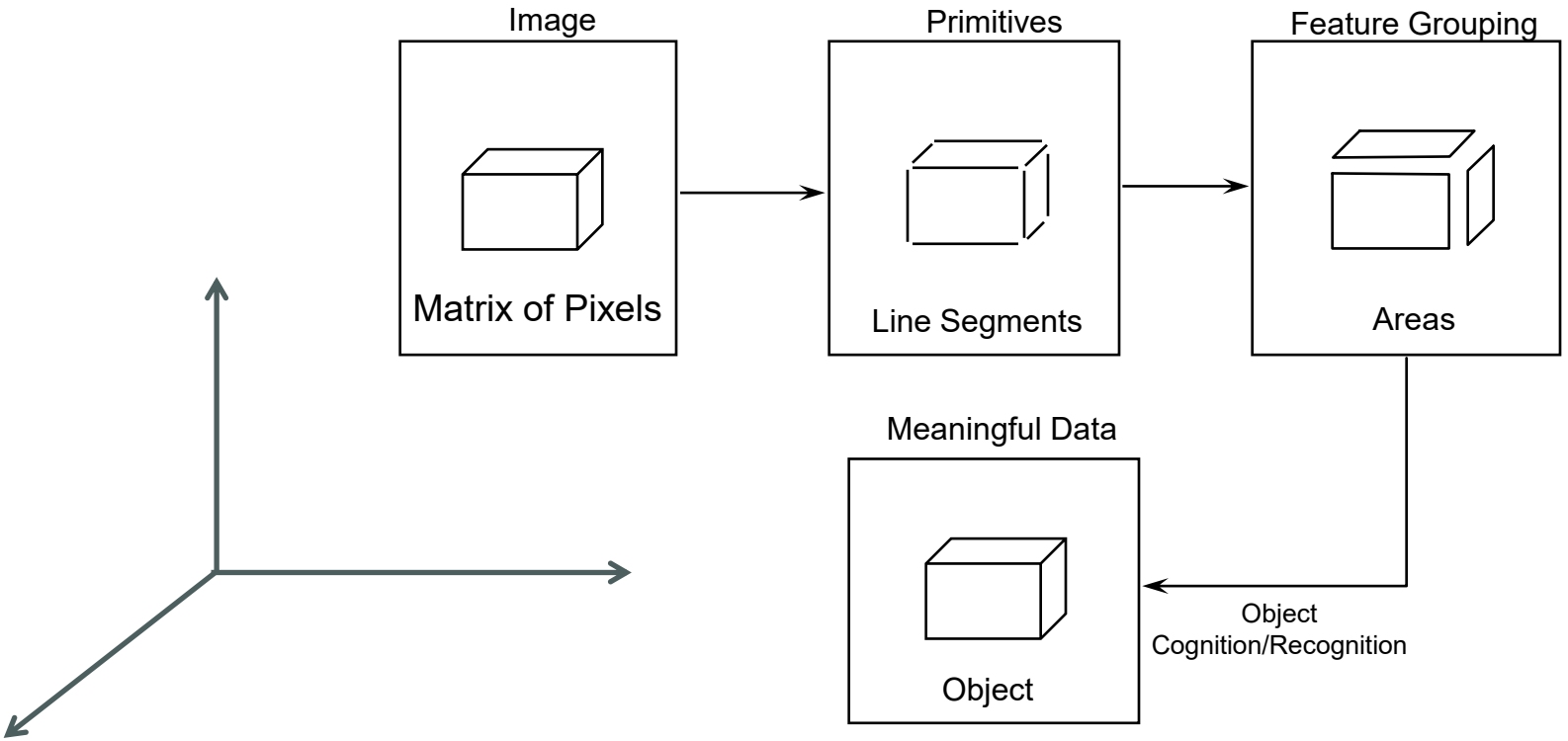
$$G^{out} = A(W \times G^{in} + B)$$

$$A(x)\ is: y = \sqrt{x \times x} \quad \text{or} \quad y = x$$



Property 2 of Artificial Neural Network

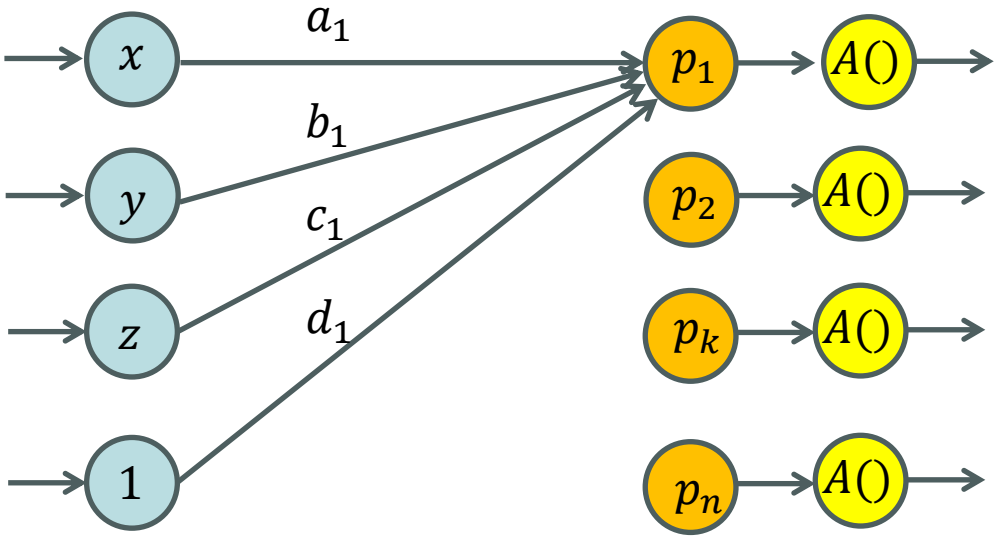
- An artificial neural network is a solution for the implementation of a system of equations which represent the boundaries inside a knowledge space.
- An artificial neural network is better to be the outcome of top-down design.
- An artificial neural network could be the outcome of bottom-up optimization.



Example of Boundary Representation

- Representational Neural Network (i.e. RNN)

$$\begin{cases} p_1 = a_1 x + b_1 y + c_1 z + d_1 \\ p_2 = a_2 x + b_2 y + c_2 z + d_2 \\ \cdots \\ p_n = a_n x + b_n y + c_n z + d_n \end{cases}$$



MIMO = Sum of MISO

A(): Activation Function

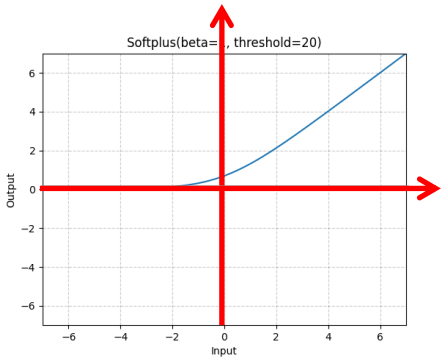
$output = A(input)$

Popular Activation Functions:

- SoftPlus Function:

$$\mathrm{X} = \{x_i, i = 1,2, \dots, n\}$$

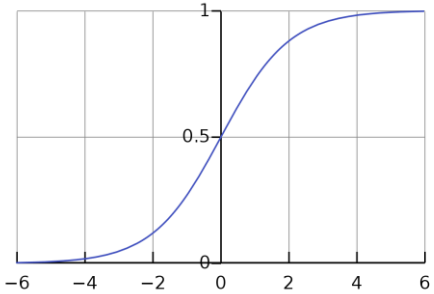
$$A(X) = \{\ln(1 + e^{x_i}), i = 1,2, \dots, n\}$$



- SoftMax Function:

$$\mathrm{X} = \{x_i, i = 1,2, \dots, n\}$$

$$A(X) = \{\frac{e^{x_i}}{\sum_{i=1}^{n} e^{x_i}}, i = 1,2, \dots n\}$$



- Sigmoid Function:

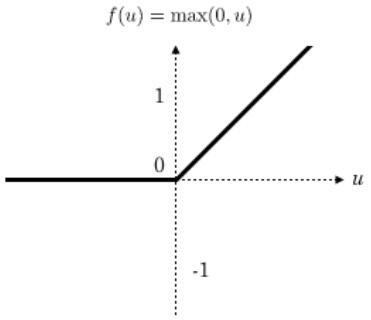
$$\mathrm{X} = \{x_i, i = 1,2, \dots, n\}$$

$$A(X) = \{\frac{1}{1 + e^{-x_i}}, i = 1,2, \dots n\}$$

- ReLU (Rectified Linear Unit):

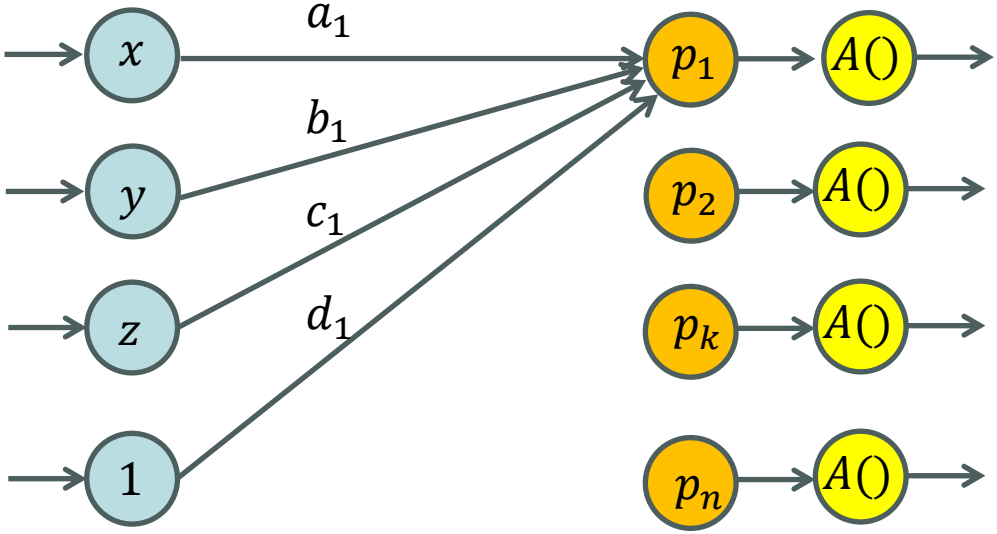
$$\mathrm{X} = \{x_i, i = 1,2, \dots, n\}$$

$$A(X) = \{\max(0, x_i), i = 1,2, \dots, n\}$$



Property 3 of Artificial Neural Network

- An artificial neural network is a predictor or classifier which could tell the identity and/or category of given physical knowledge.
- An artificial neural network is a solution to the symbol-grounding problem in AI.



MIMO = Sum of MISO

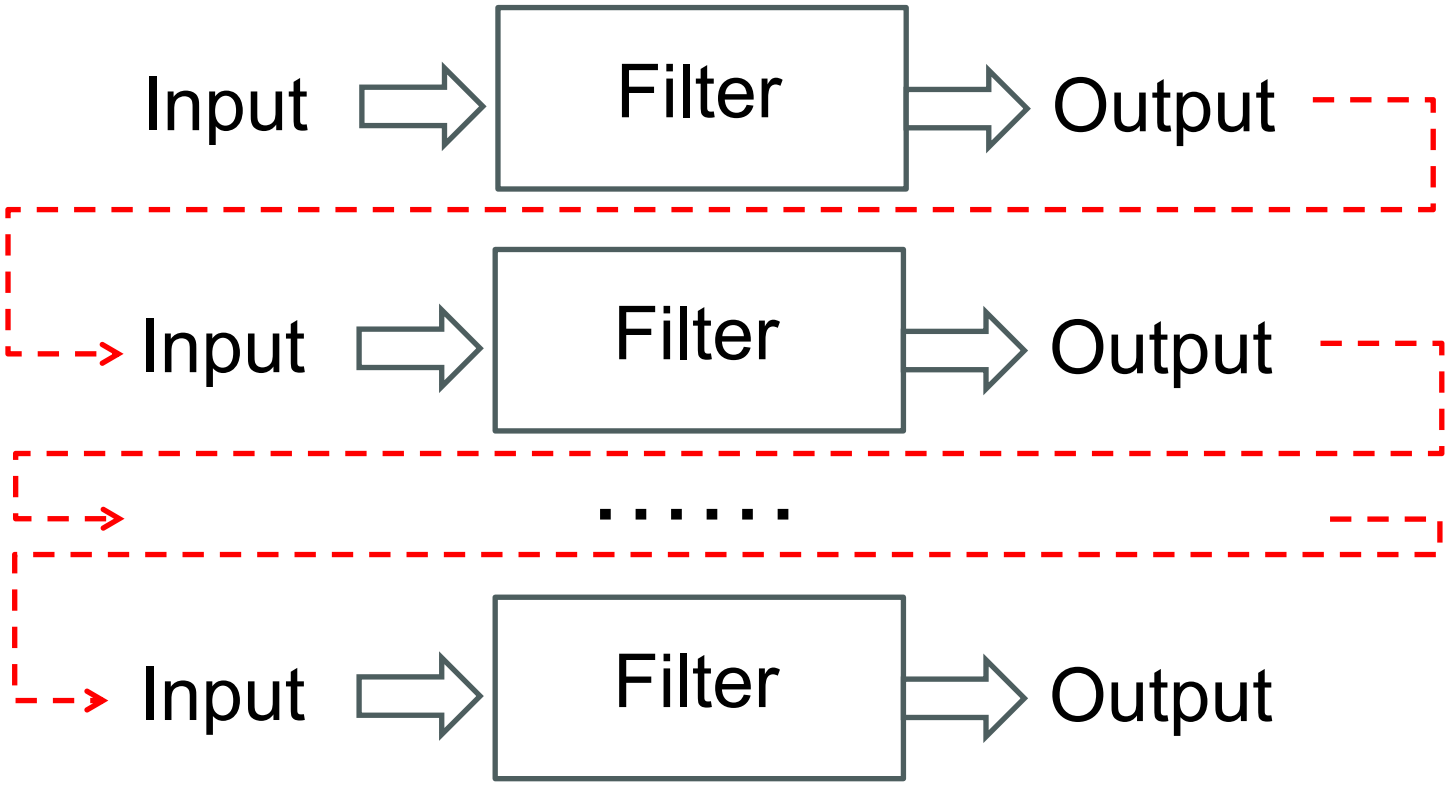
A(): Activation Function

$$A(input) = \begin{cases} -1, if\ input < 0 \\ 0, if\ input = 0 \\ +1, if\ input > 0 \end{cases}$$

{belong, not belong}

What is deep neural network?

- It is a cascaded artificial neural network which integrates artificial neural network's computational power and prediction capability together.



Details of Deep Neural Network

(output = symbol with belief) $A(X)$: $SoftPlus, SoftMax, Sigmoid, etc$

Feature Vector at Layer 0:

Input = $F_0 = \{f_{0,i}, i = 1,2, \dots, m\}$

Feature Vector at Layer 1:

$F_1 = A(\mathrm{W}_0 \times F_0 + \mathrm{B}_0)$

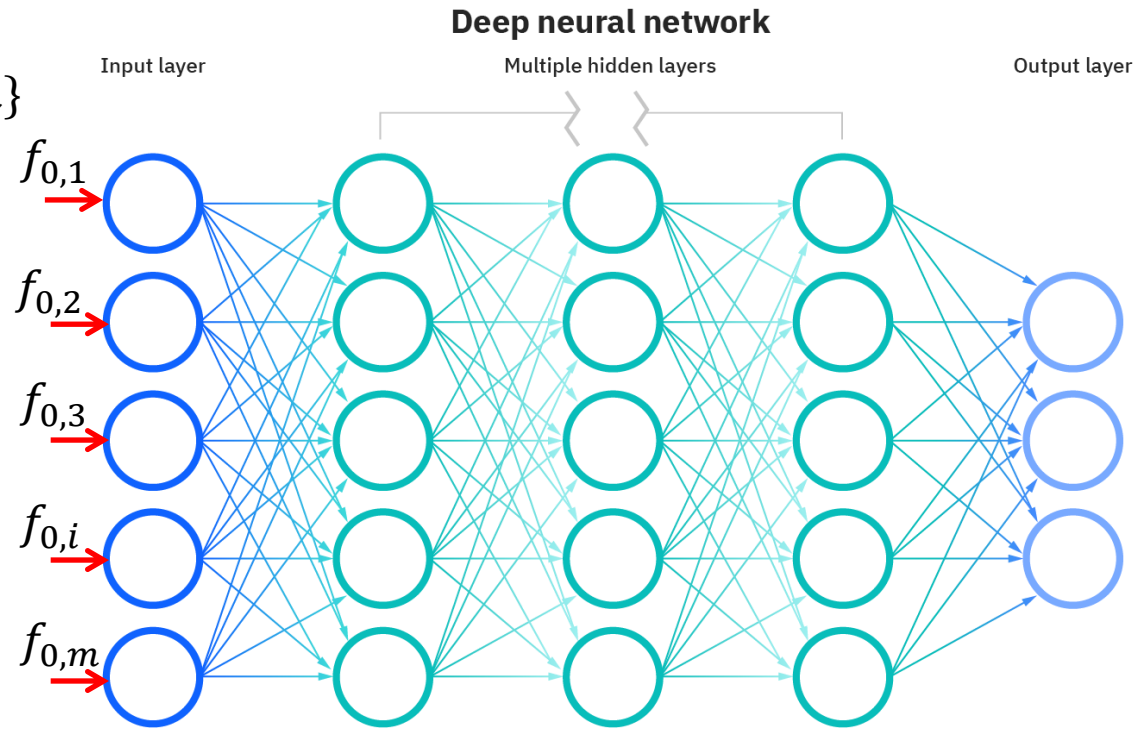
Feature Vector at Layer k:

$F_k = A(\mathrm{W}_{k-1} \times F_{k-1} + \mathrm{B}_{k-1})$

.....

Feature Vector at Layer n:

$F_n = A(\mathrm{W}_{n-1} \times F_{n-1} + \mathrm{B}_{n-1})$



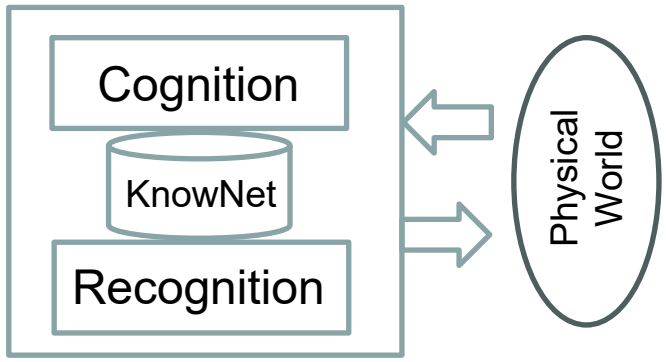
Conceptual Meanings at Output:

$Belief = \{[belief_j, possibility_j(F_n)], j = 1,2, \dots, L\}$

MIMO

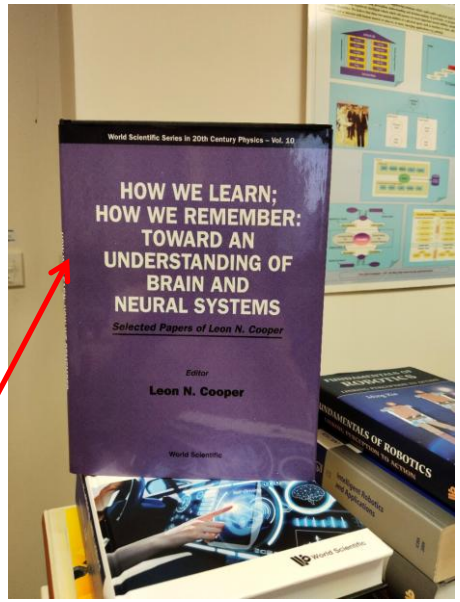
Outline of Lecture 3

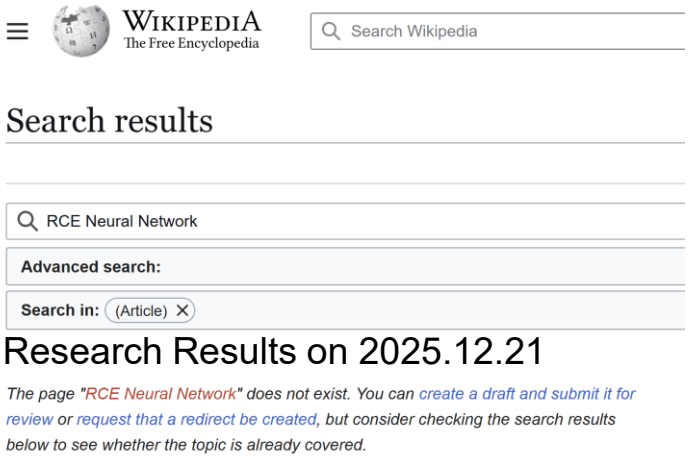
- Principle of Cognition (Learning)
- Principle of Artificial Neural Network
- Principle of RCE Neural Network
- Cognition of Colors
- Cognition of Curves
- Cognition of Pattern



History of RCE Neural Network (RNN)

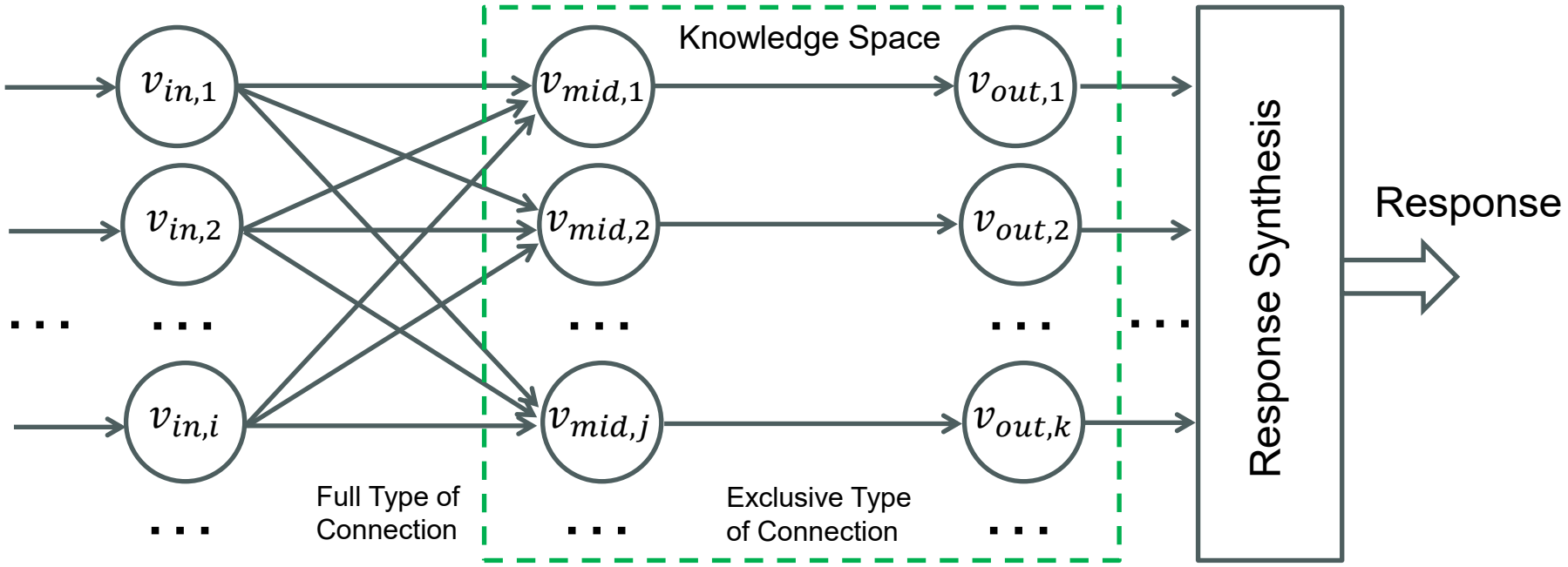
- **Leon N. Cooper** (né **Kupchik**; February 28, 1930 – October 23, 2024) was an American theoretical physicist and neuroscientist. He won the Nobel Prize in Physics for his work on superconductivity. Cooper developed the concept of Cooper pairs and collaborated with John Bardeen and John Robert Schrieffer to develop the BCS theory of conventional superconductivity. In neuroscience, Cooper co-developed the BCM theory of synaptic plasticity.
- Restricted Coulomb RCE Neural Network was developed by a team led by Leon N. Cooper in the late 1970s.
- Some modified versions of RCE Neural Network was developed by T. Olmez et al (1993) and M. Xie et al (2003).





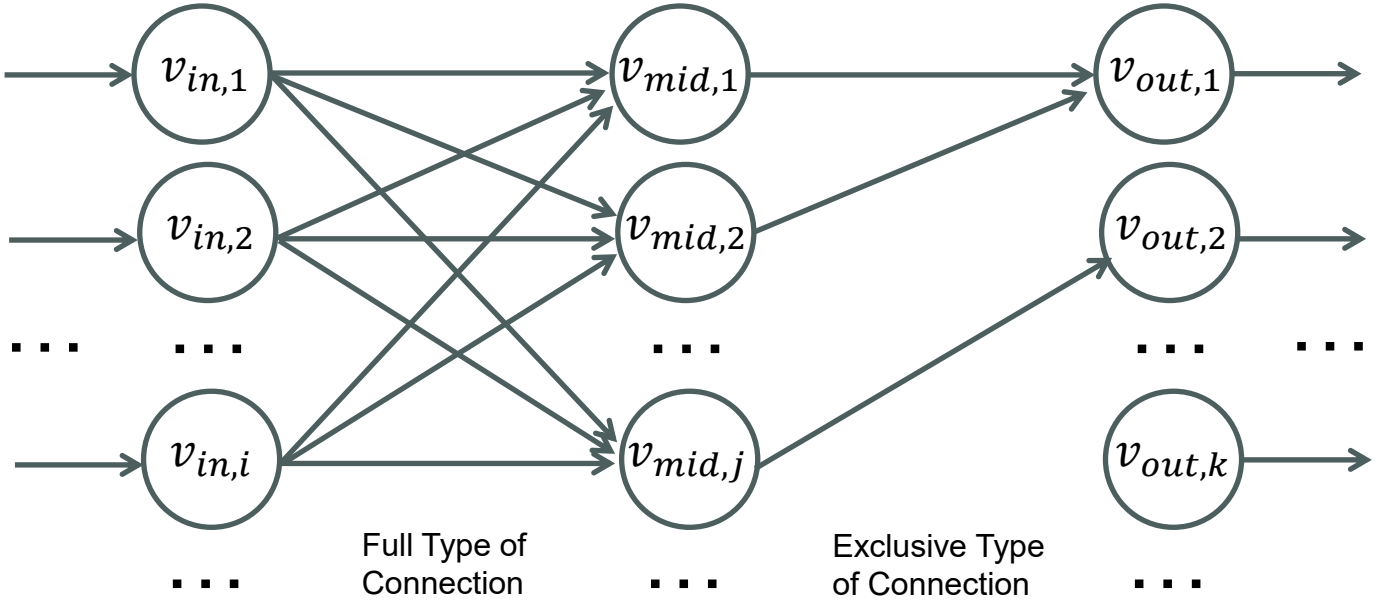
Definition of RCE Neural Network

- A RCE neural network is an associative memory of vectors for the representation of a knowledge space.



Property 1 of RCE Neural Network

- RCE neural network is a vectorized neural network which has three layers.
- Each layer consists of a list of knowledge vectors. The weighting coefficient at each link is 1.
- Between the input layer and middle layer (also called prototype layer), the connection among the node is full-type of connection.
- Between the middle layer and output layer, the connection among the nodes is exclusive type of connection.



Property 2 of RCE Neural Network

- RCE neural network is a MIMO system which could be represented as:
  - Sum of MISO
  - Sum of SIMO
  - Sum of SISO
- Hence, RCE neural network represents a knowledge space which supports incremental learning.

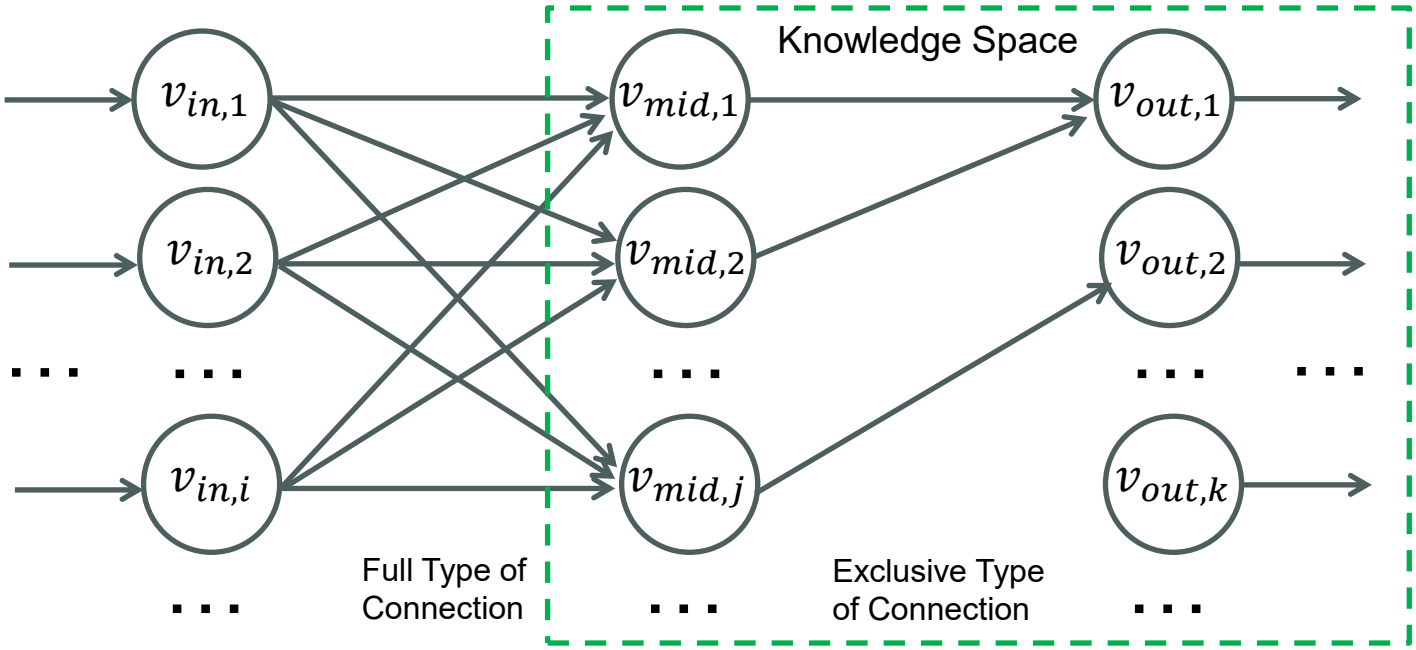
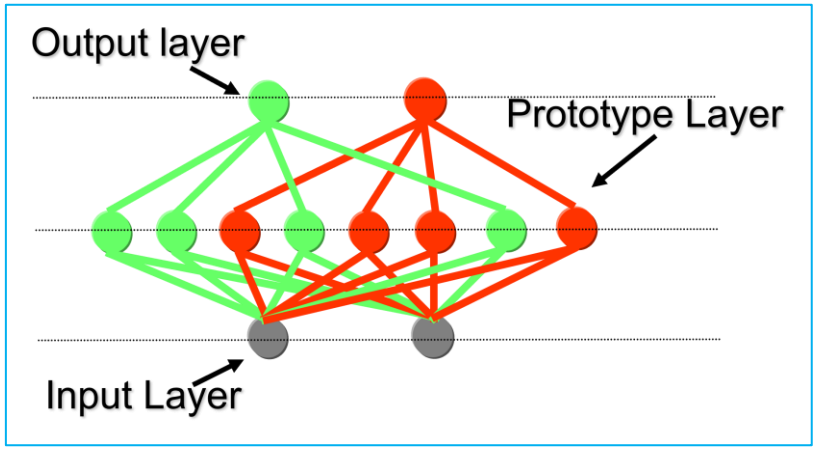


Illustration of Property 2:

- MIMO = Sum of SISO
- RCE NN is an ideal system!

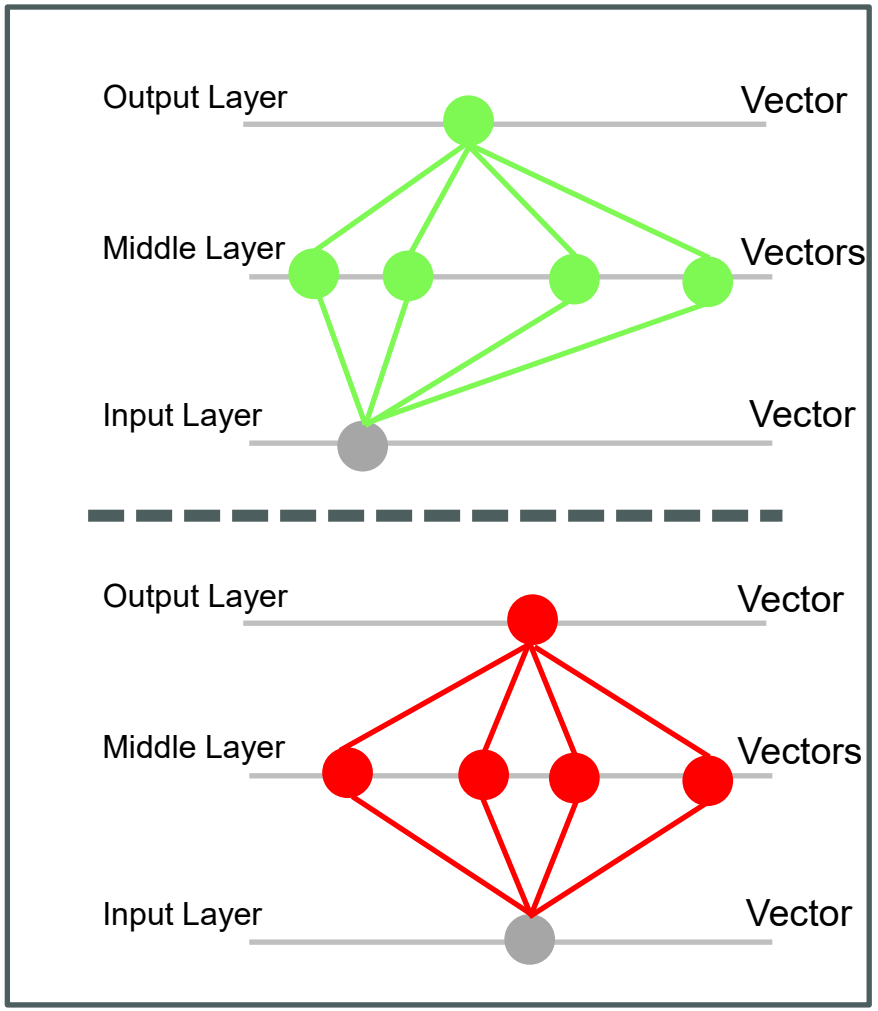


MIMO = Sum of MISO

MIMO = Sum of SISO

=

Incremental Learning



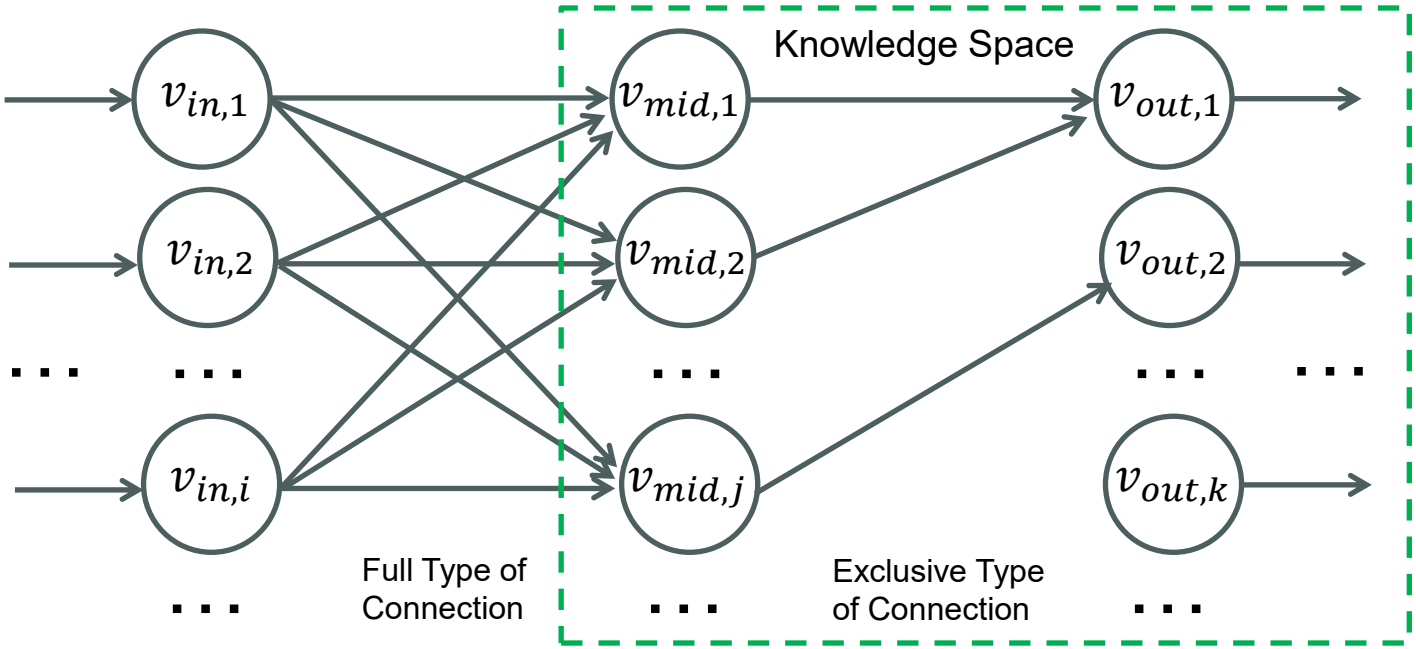
Property 3 of RCE Neural Network

- Each vector at the input layer is a feature vector which could be represented as:

$$v_{in,i} = \{v_{in,i,l}, l = 1,2, \ldots\}$$

Could be incrementally increased

- Output from each node of input layer is its feature vector.



Property 4 of RCE Neural Network

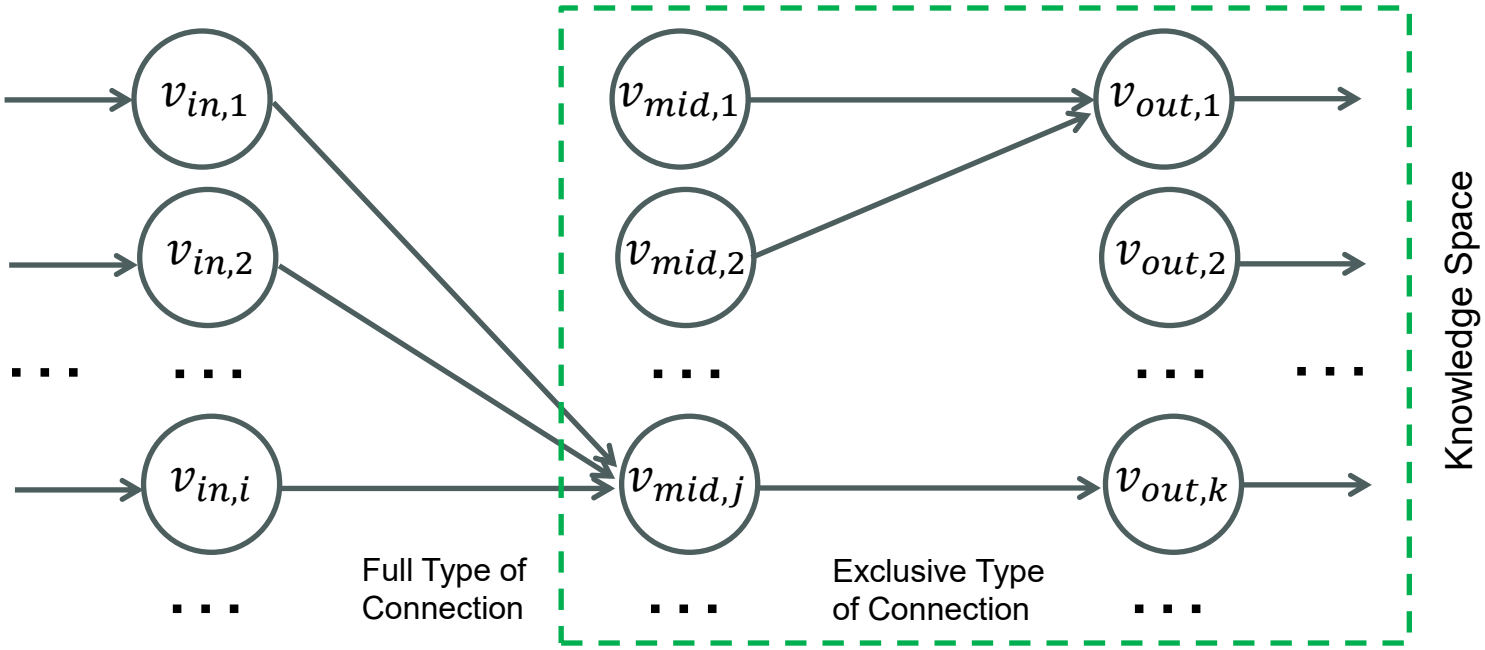
- Each vector at the middle layer (also called prototype layer) is knowledge vector which could be represented as:

$$v_{mid,j} = \{v_{mid,j,l}, l = 1,2, \ldots\} = \{\mu_j, \Sigma_j, p_j, conceptual\ labels, others\}$$

μ_j: mean of training inputs
Σ_j: co-variance of training inputs

$j = 1,2, \ldots$

From Teachers/Trainers



Property 5 of RCE Neural Network

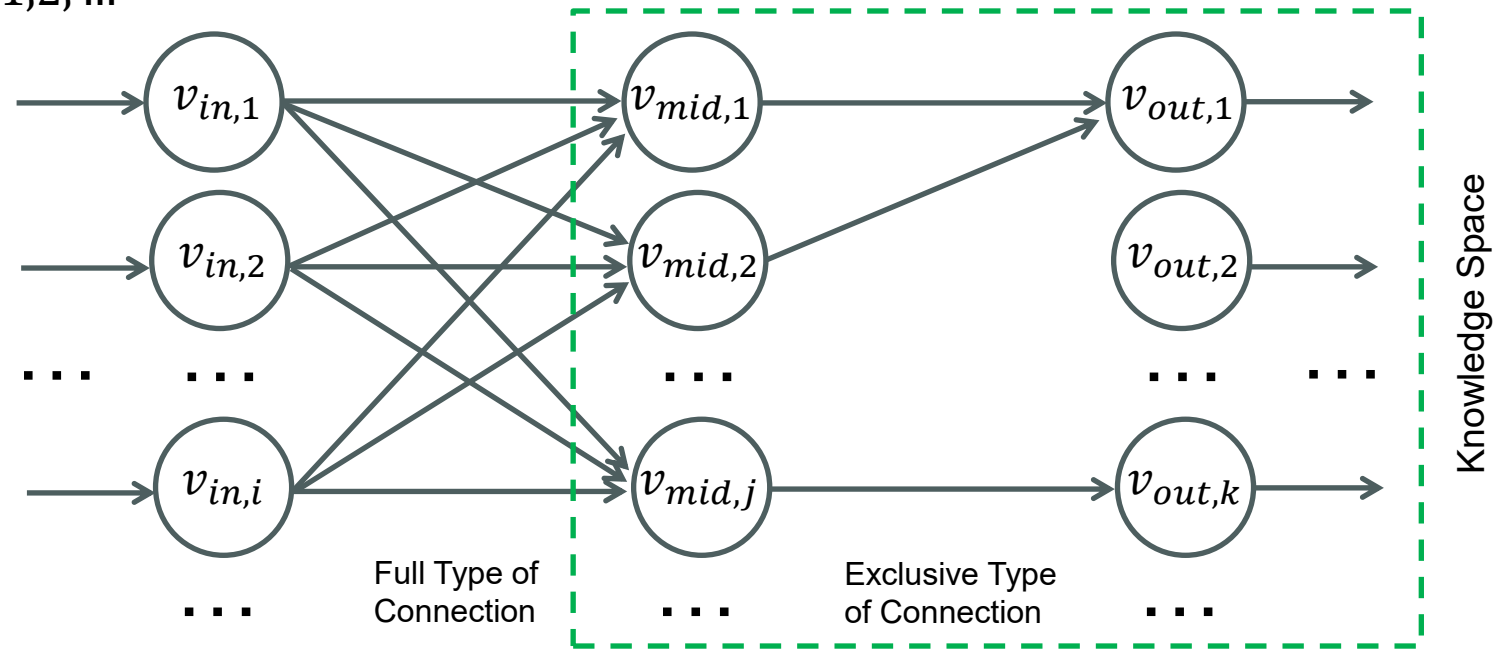
- Output from each node of middle layer is a possibility value:

$$v_{mid,j} = \{v_{mid,j,l}, l = 1,2, \ldots\} = \{\mu_j, \Sigma_j, p_j, conceptual\ labels, others\}$$

μ_j: means of training inputs
Σ_j: co-variance of training inputs

$$p_j = e^{-\frac{1}{2}\left(v_{in,i}-\mu_j\right)^t \Sigma_j^{-1}\left(v_{in,i}-\mu_j\right)}$$

$j = 1,2, \ldots$



Property 6 of RCE Neural Network

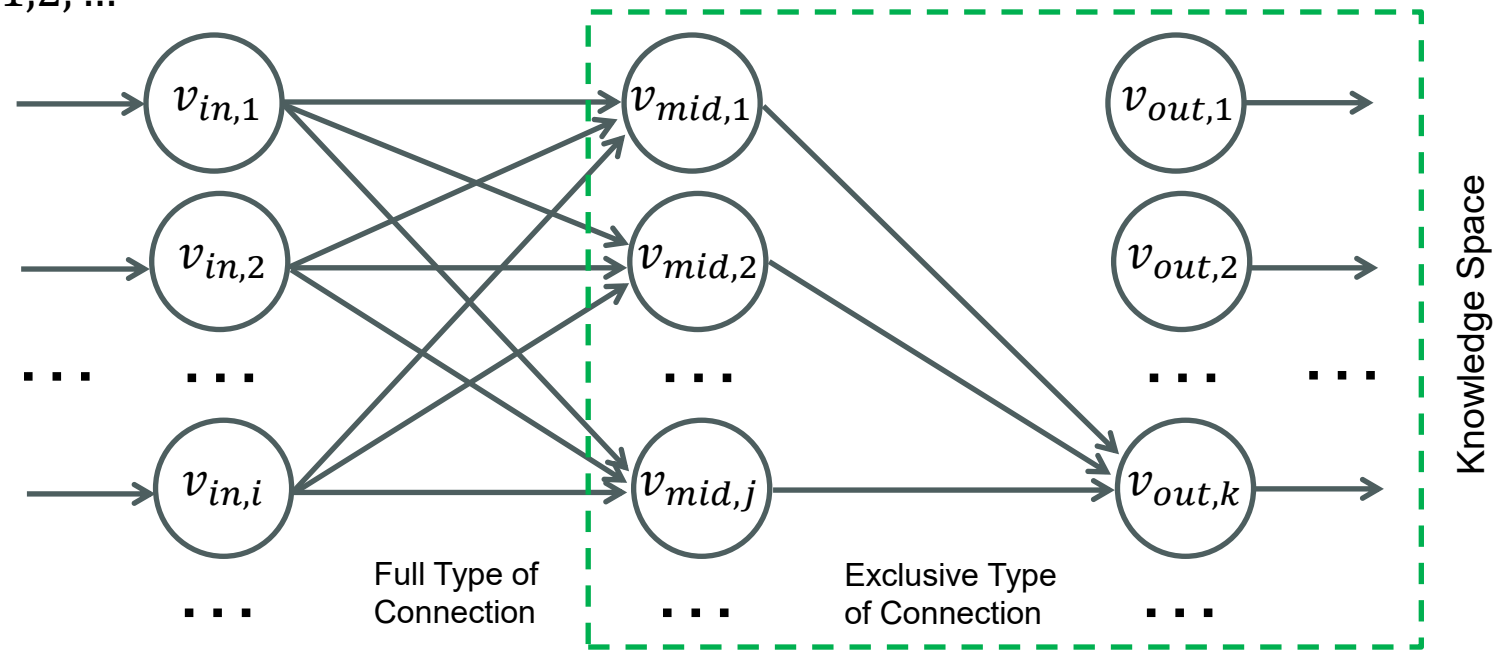
- Output from each node of output layer is the copy of the selected node from the middle layer:

$$v_{mid,j} = \{v_{mid,j,l}, l = 1,2, \ldots\} = \{\mu_j, \Sigma_j, p_j, conceptual\ labels, others\}$$

$$p_j = e^{-\frac{1}{2}\left(v_{in,i}-\mu_j\right)^t \Sigma_j^{-1}\left(v_{in,i}-\mu_j\right)}$$

$$\forall j, v_{out,k} = v_{mid,j}, if\ p_j = max$$

$j = 1,2, \ldots$



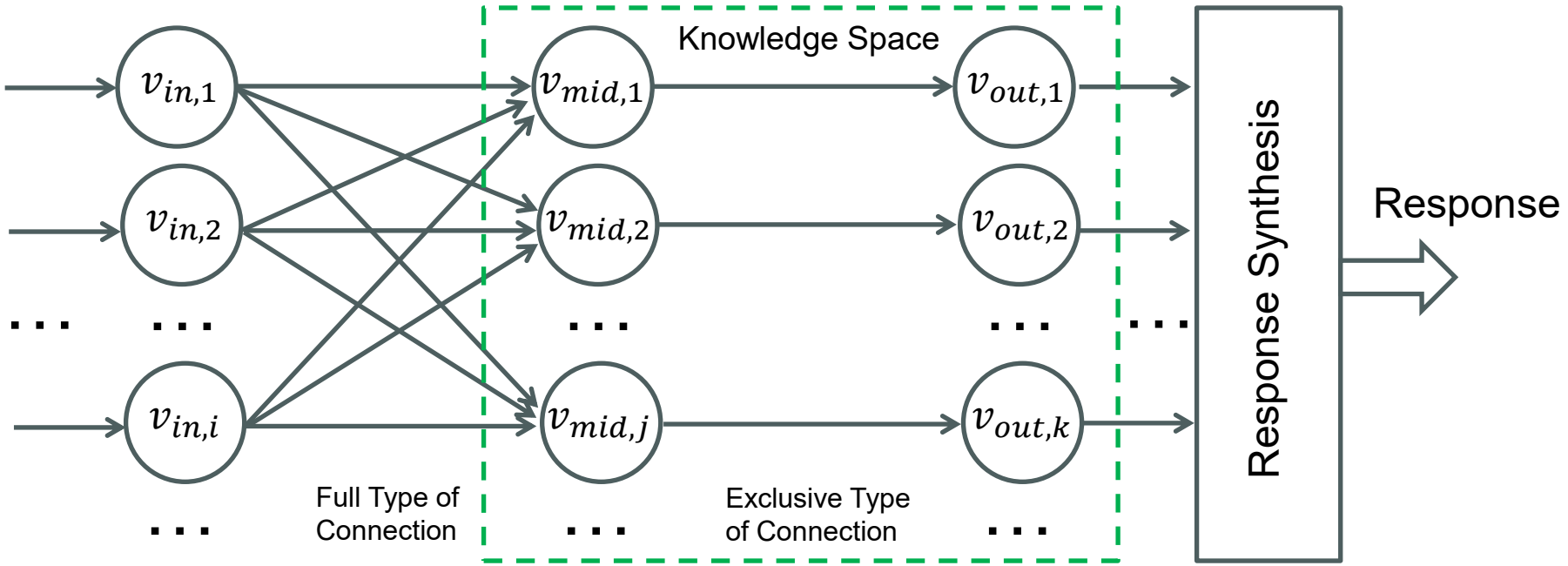
Property 7 of RCE Neural Network

- RCE neural network is a representation of knowledge space which is incrementally enriched by cognition or learning.
- RCE neural network is also an engine for knowledge recognition, which behaves like a sum of SISO systems.

$$\forall k, R_{reply} = S(v_{out,k}), if\ p_k = max$$

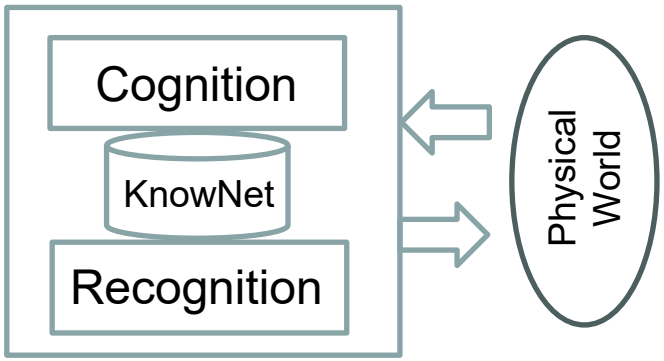
Use of Programming Language

$S(\quad)$: Any function of synthesis



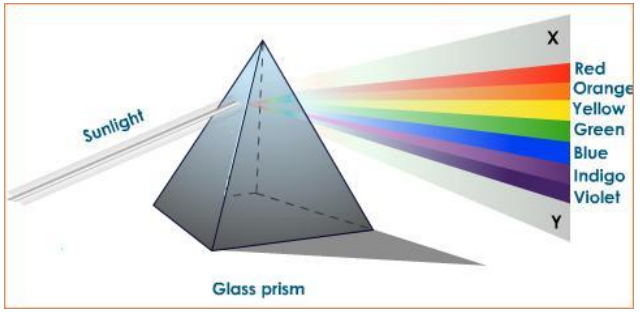
Outline of Lecture 3

- Principle of Cognition (Learning)
- Principle of Artificial Neural Network
- Principle of RCE Neural Network
- Cognition of Colors
- Cognition of Curves
- Cognition of Pattern



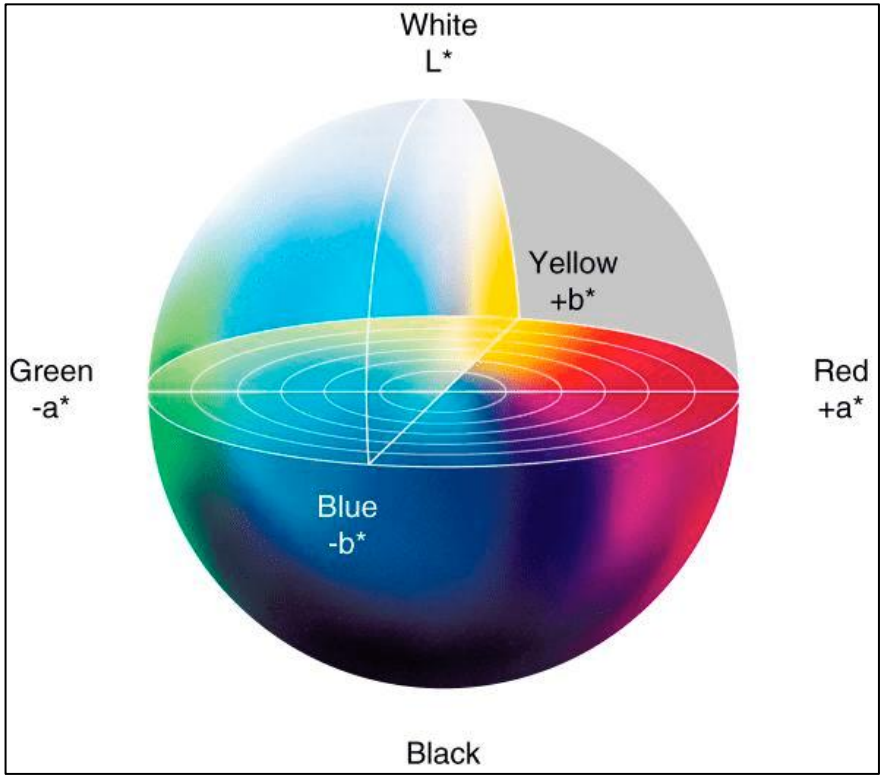
What is a color?

- A color refers to pixels' coordinates in a color space.



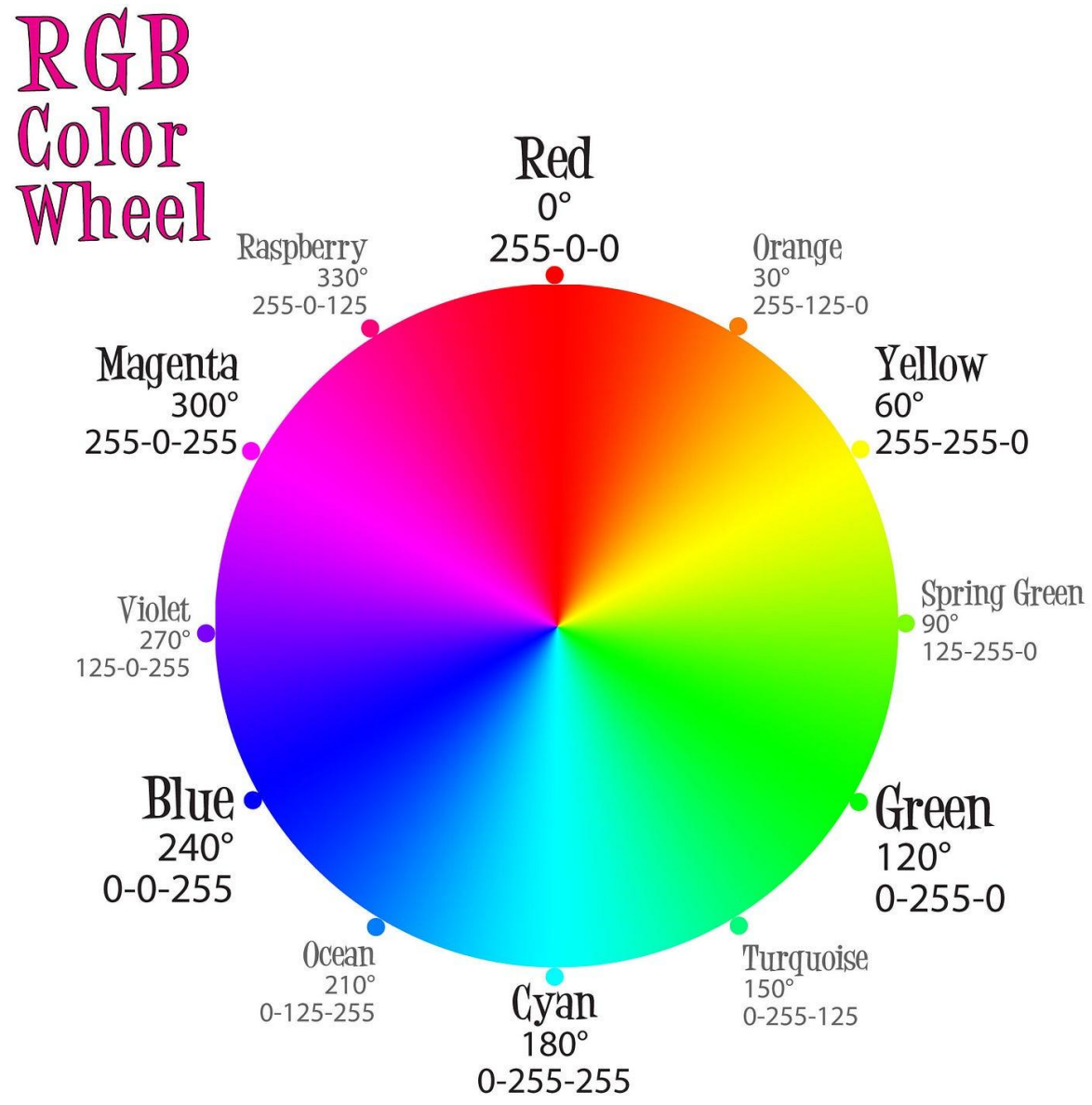
Color Image

Color Space

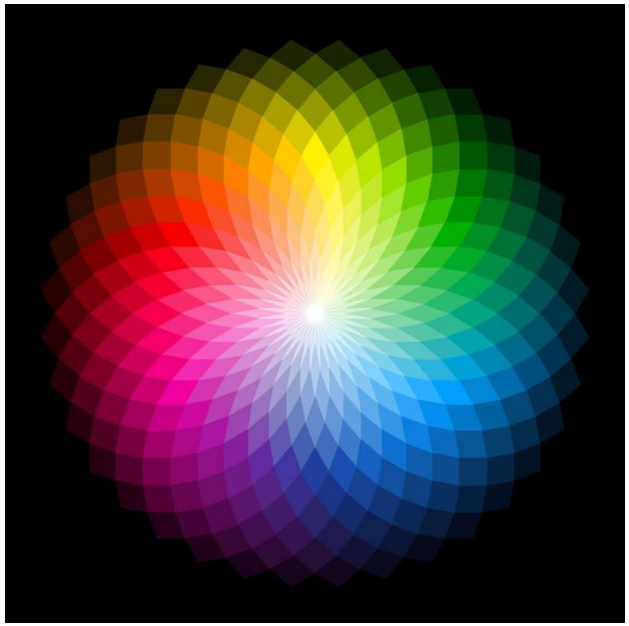


Example of Concept-Physical Knowledge Related to Colors

How to enable machines to cognize or learn such knowledge?



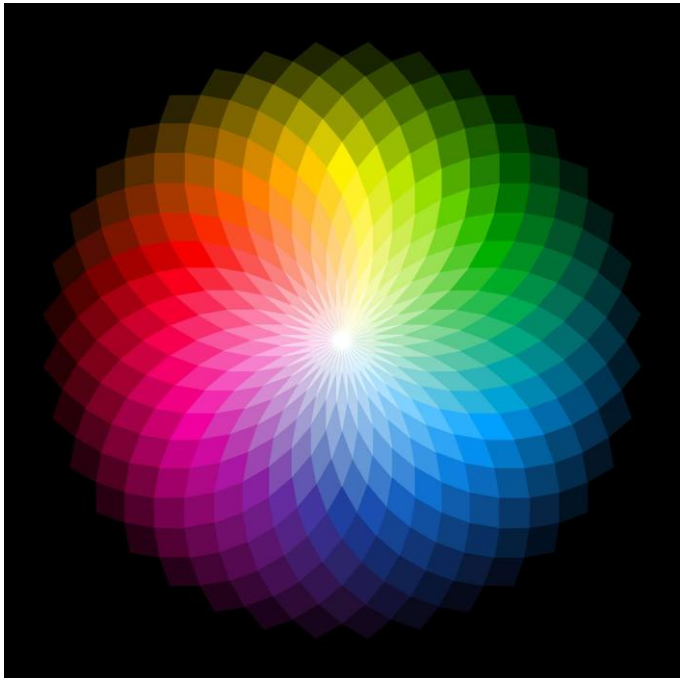
Solution Toward Cognition and Recognition of Colors



Input $v_{in,i} = \{r, g, b, I, L, a, b, u, v, \dots\}$

Images as 2D Matrices

- Color Images for Training



Data Structure of Pixel

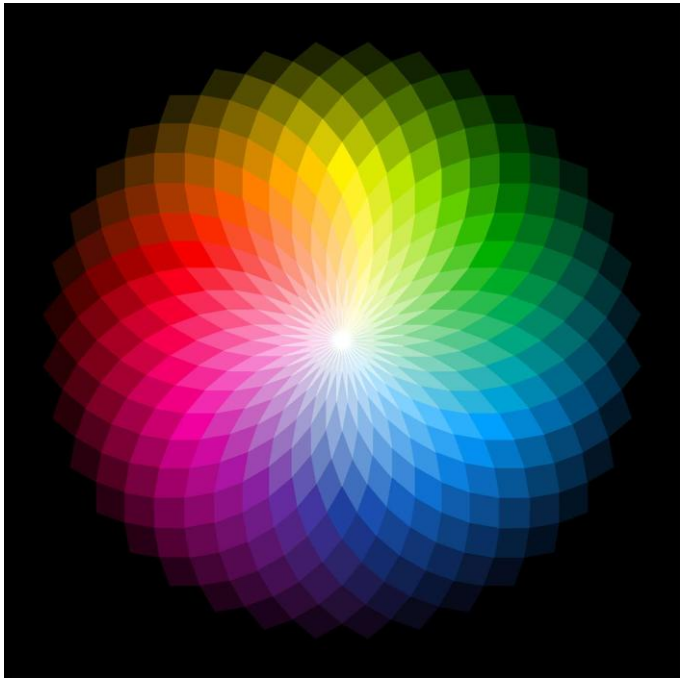
```
typedef   struct
{
        short     r, g, b, I ;
        short     L, a, b ;
        short     u, v ;
        double    x, y, z ;
}
Pixel;
```

Output

$$v_{mid,j} = \{\mu_j, \Sigma_j, p_j, conceptual\ labels, others\}$$

Images as 2D Matrices

- Color Images for Training

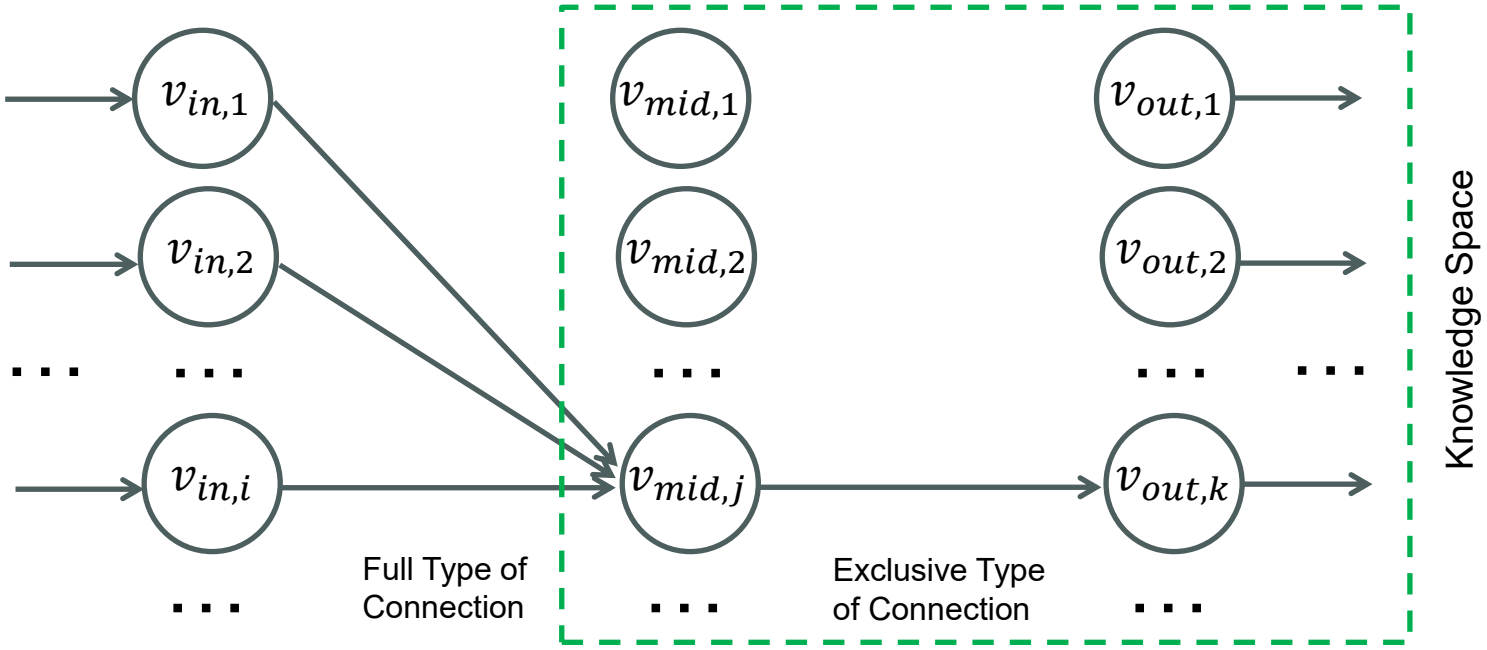


Data Structure of Pixel

```
typedef   struct
{
        short     r, g, b, I ;
        short     L, a, b ;
        short     u, v ;
        double    x, y, z ;
        string    color_name ;
        double    feature_vector[ ];
}
Pixel;
```

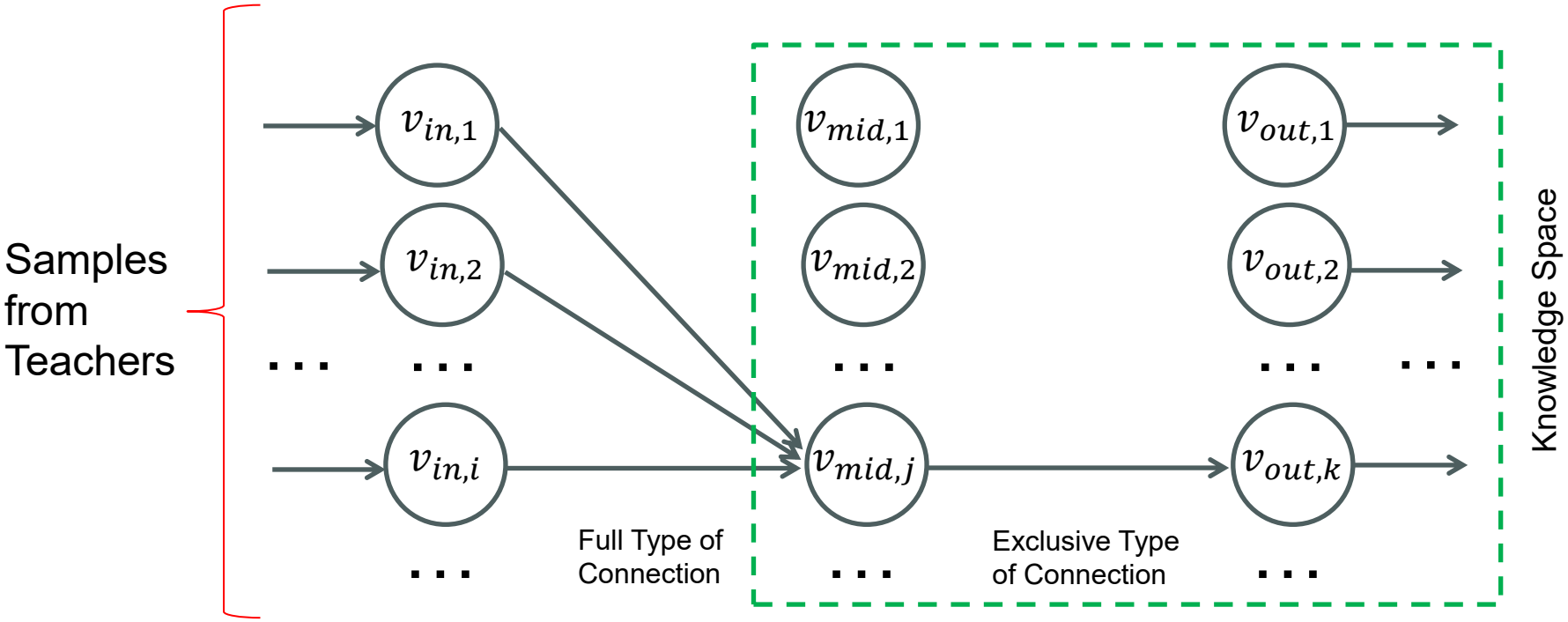
Question

- How to cognize or learn the color that a pixel inside an image belongs to?



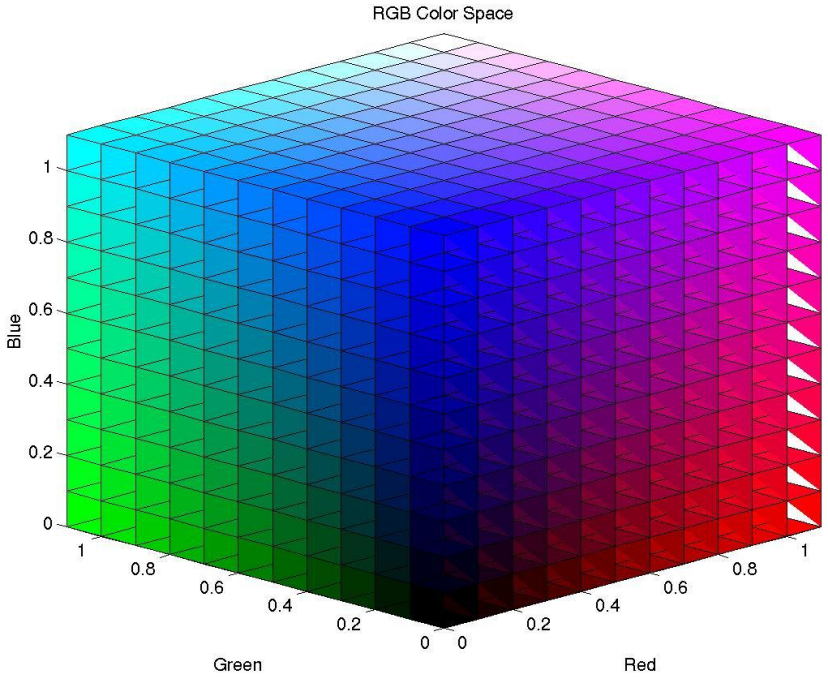
Step 1: To obtain visual samples (case of colors) from input image …

- Solution is to receive the training samples of colors from a teacher.

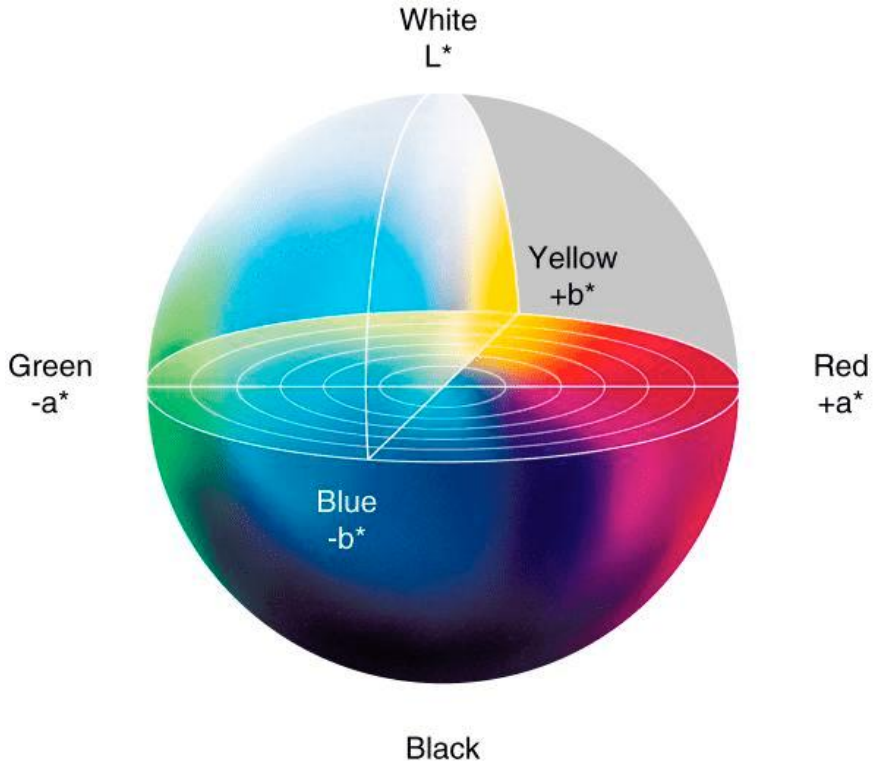


Step 2: To represent a color by a vector

- Use of vectors in a chosen color space such as:



RGB Color Space



L*a*b Color Space

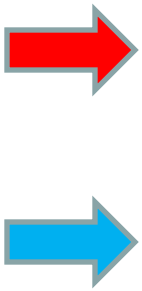
Color Conversion from RGB to L*a*b

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.607 & 0.174 & 0.200 \\ 0.299 & 0.587 & 0.114 \\ 0.000 & 0.066 & 1.116 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

$$\begin{cases} L = 25.0 \bullet (100.0 \bullet Y / Y_{\max})^{1/3} - 16.0 \\ a = 500.0 \bullet [(X / X_{\max})^{1/3} - (Y / Y_{\max})^{1/3}] \\ b = 200.0 \bullet [(Y / Y_{\max})^{1/3} - (Z / Z_{\max})^{1/3}] \end{cases}$$

1. L* axis (lightness) ranges from 0 to 100
2. a* and b* (color attributes) axis range from -128 to +127

RGB Working Space	Reference White	RGB to XYZ [M]
Adobe RGB (1998)	D65	0.5767309 0.1855540 0.1881852 0.2973769 0.6273491 0.0752741 0.0270343 0.0706872 0.9911085
AppleRGB	D65	0.4497288 0.3162486 0.1844926 0.2446525 0.6720283 0.0833192 0.0251848 0.1411824 0.9224628
Best RGB	D50	0.6326696 0.2045558 0.1269946 0.2284569 0.7373523 0.0341908 0.0000000 0.0095142 0.8156958
Beta RGB	D50	0.6712537 0.1745834 0.1183829 0.3032726 0.6637861 0.0329413 0.0000000 0.0407010 0.7845090
Bruce RGB	D65	0.4674162 0.2944512 0.1886026 0.2410115 0.6835475 0.0754410 0.0219101 0.0736128 0.9933071
CIE RGB	E	0.4887180 0.3106803 0.2006017 0.1762044 0.8129847 0.0108109 0.0000000 0.0102048 0.9897952



Ekta Space PS5	D50	0.5938914 0.2729801 0.0973485 0.2606286 0.7349465 0.0044249 0.0000000 0.0419969 0.7832131
NTSC RGB	C	0.6068909 0.1735011 0.2003480 0.2989164 0.5865990 0.1144845 0.0000000 0.0660957 1.1162243
PAL/SECAM RGB	D65	0.4306190 0.3415419 0.1783091 0.2220379 0.7066384 0.0713236 0.0201853 0.1295504 0.9390944
ProPhoto RGB	D50	0.7976749 0.1351917 0.0313534 0.2880402 0.7118741 0.0000857 0.0000000 0.0000000 0.8252100
SMPTE-C RGB	D65	0.3935891 0.3652497 0.1916313 0.2124132 0.7010437 0.0865432 0.0187423 0.1119313 0.9581563
sRGB	D65	0.4124564 0.3575761 0.1804375 0.2126729 0.7151522 0.0721750 0.0193339 0.1191920 0.9503041
Wide Gamut RGB	D50	0.7161046 0.1009296 0.1471858 0.2581874 0.7249378 0.0168748 0.0000000 0.0517813 0.7734287

Color Conversion Program in MATLAB

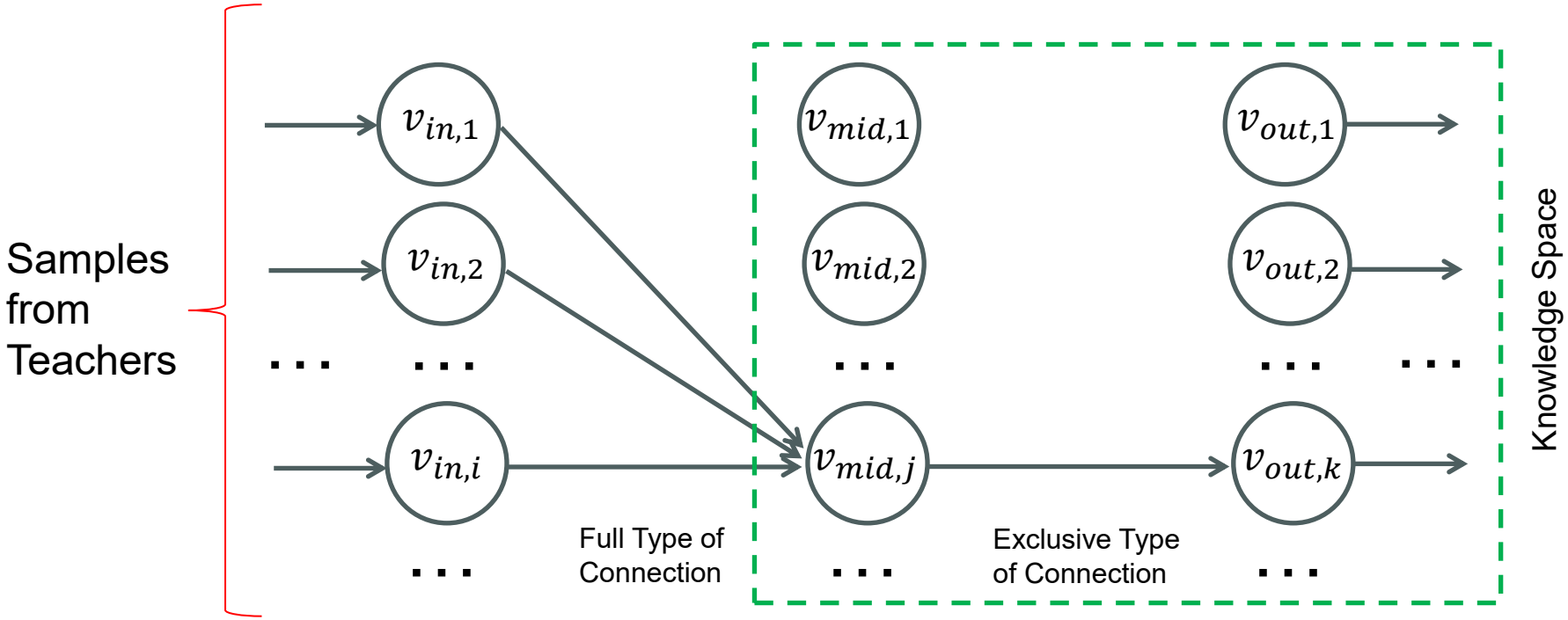
- M=[0.607 0.174 0.2 ; 0.299 0.587 0.114; 0.0 0.066 1.116];
- RGB_MAX = [255 255 255];
- XYZ_MAX = M*RGB_MAX';
- RGB = [50 100 150];
- XYZ = M*RGB';
- x = (XYZ(1)/XYZ_MAX(1));
- y = (XYZ(2)/XYZ_MAX(2));
- z = (XYZ(3)/XYZ_MAX(3));
- L = 25.0*(100.0)^(1/3)*y^(1/3) - 16.0
- a = 500.0*(x^(1/3)-y^(1/3))
- b = 200.0*(y^(1/3) - z^(1/3))

ma4829RGBtoLAB.m

```
1   M=[0.607 0.174  0.2 ; 0.299  0.587  0.114; 0.0  0.066  1.116];
2
3   RGB_MAX = [255  255   255];
4
5   XYZ_MAX = M*RGB_MAX';
6
7   RGB = [50  100  150];
8
9   XYZ = M*RGB';
10
11  x = (XYZ(1)/XYZ_MAX(1));
12  y = (XYZ(2)/XYZ_MAX(2));
13  z = (XYZ(3)/XYZ_MAX(3));
14
15  L = 25.0*(100.0)^(1/3)*y^(1/3) - 16.0
16  a = 500.0*(x^(1/3)-y^(1/3))
17  b = 200.0*(y^(1/3) - z^(1/3))
18
```

Step 3a: To compute the knowledge vector

$$v_{mid,j} = \{\mu_j, \Sigma_j, p_j, conceptual\ labels, others\}$$



Simplified Example …

- Receive a color name and a set of training colors: $\{(L_i, a_i, b_i), i = 1, .., n\}$

- Compute the mean value and variance of a:

$$\mu_a = \frac{1}{n}\sum_{i=1}^{n} a_i \qquad \sigma_a^2 = \frac{1}{n}\sum_{i=1}^{n} (a_i - \mu_a)^2$$

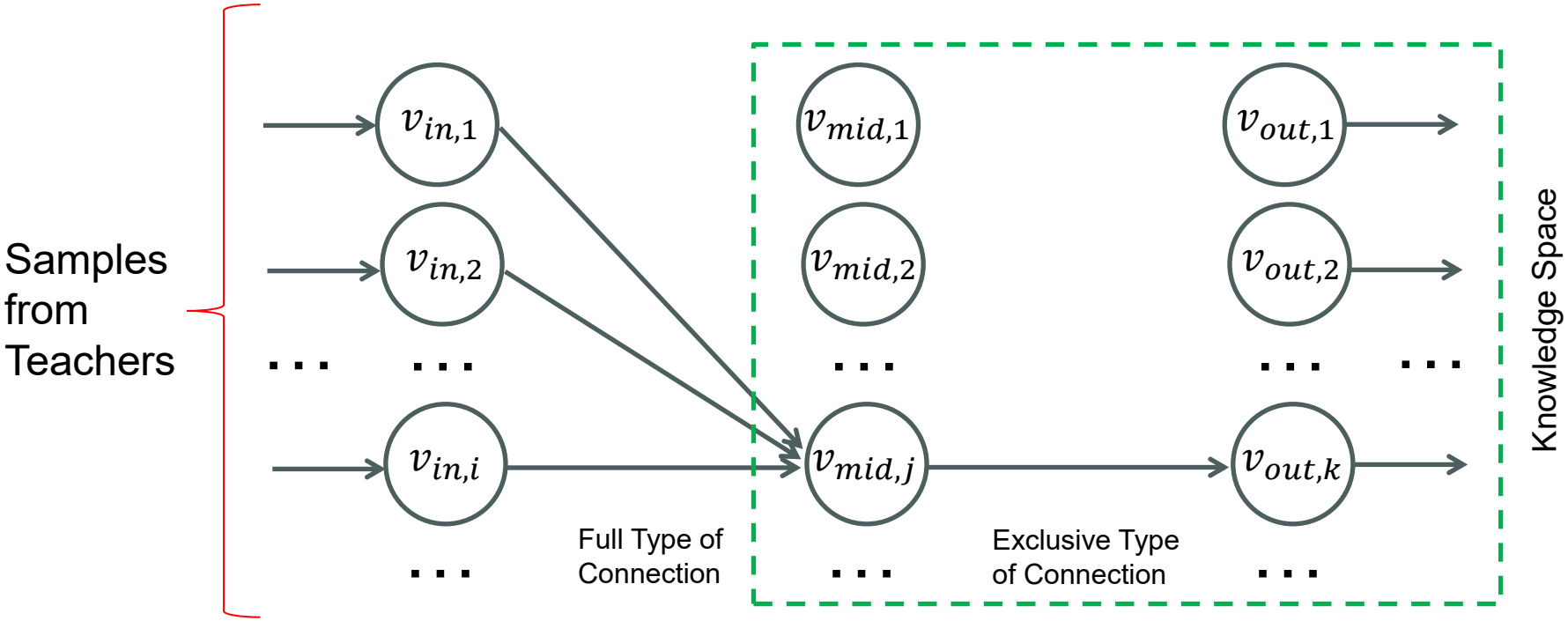
- Compute the mean value and variance of b:

$$\mu_b = \frac{1}{n}\sum_{i=1}^{n} b_i \qquad \sigma_b^2 = \frac{1}{n}\sum_{i=1}^{n} (b_i - \mu_b)^2$$

Simplified Example (continued) …

- Define the possibility function to be:

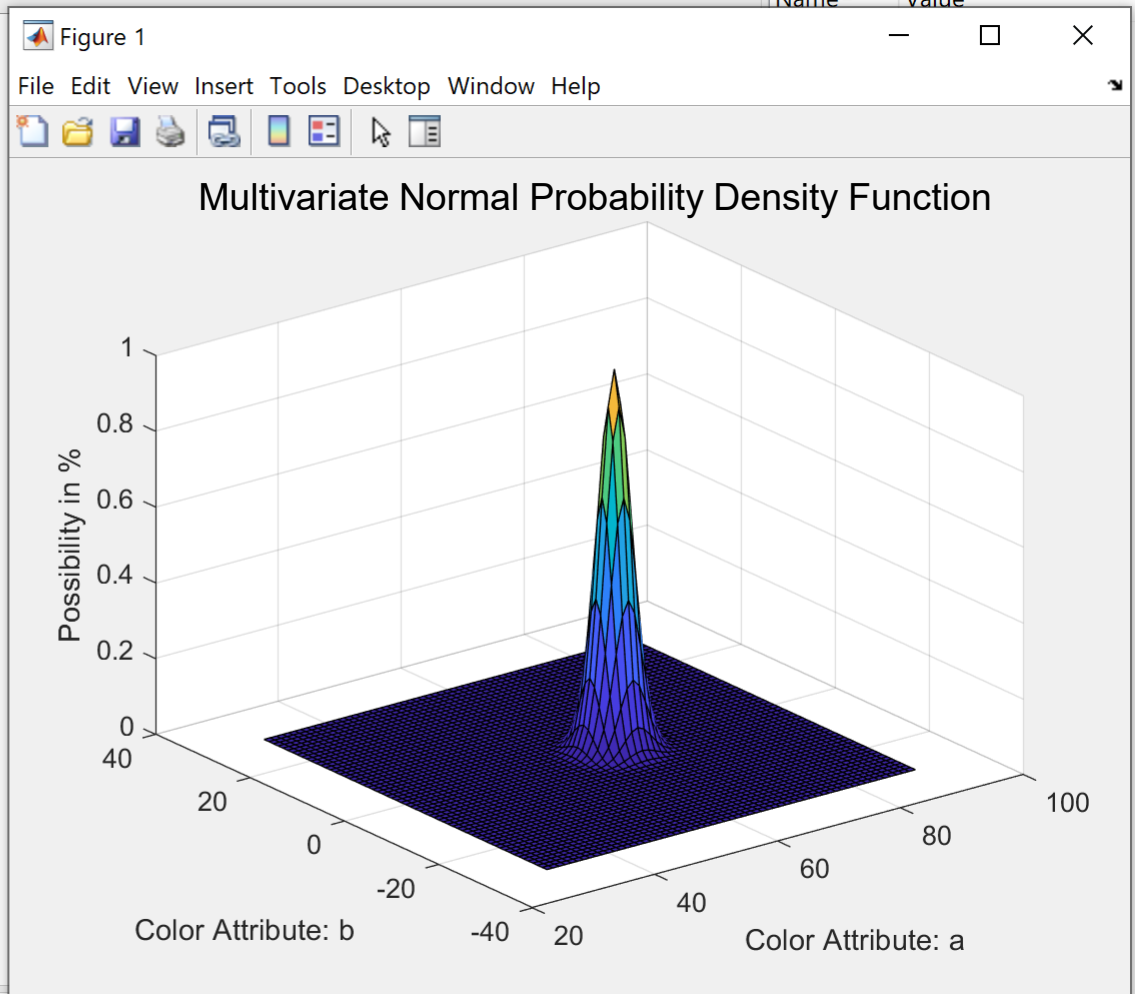
$$p_j(a,b) = e^{-\frac{1}{2}(\frac{a-\mu_a}{\sigma_a})^2} \times e^{-\frac{1}{2}(\frac{b-\mu_b}{\sigma_b})^2}$$



The Possibility Function (i.e. Symbol Grounding Function) Will Look Like …

ma4829module5mvnpdf.m

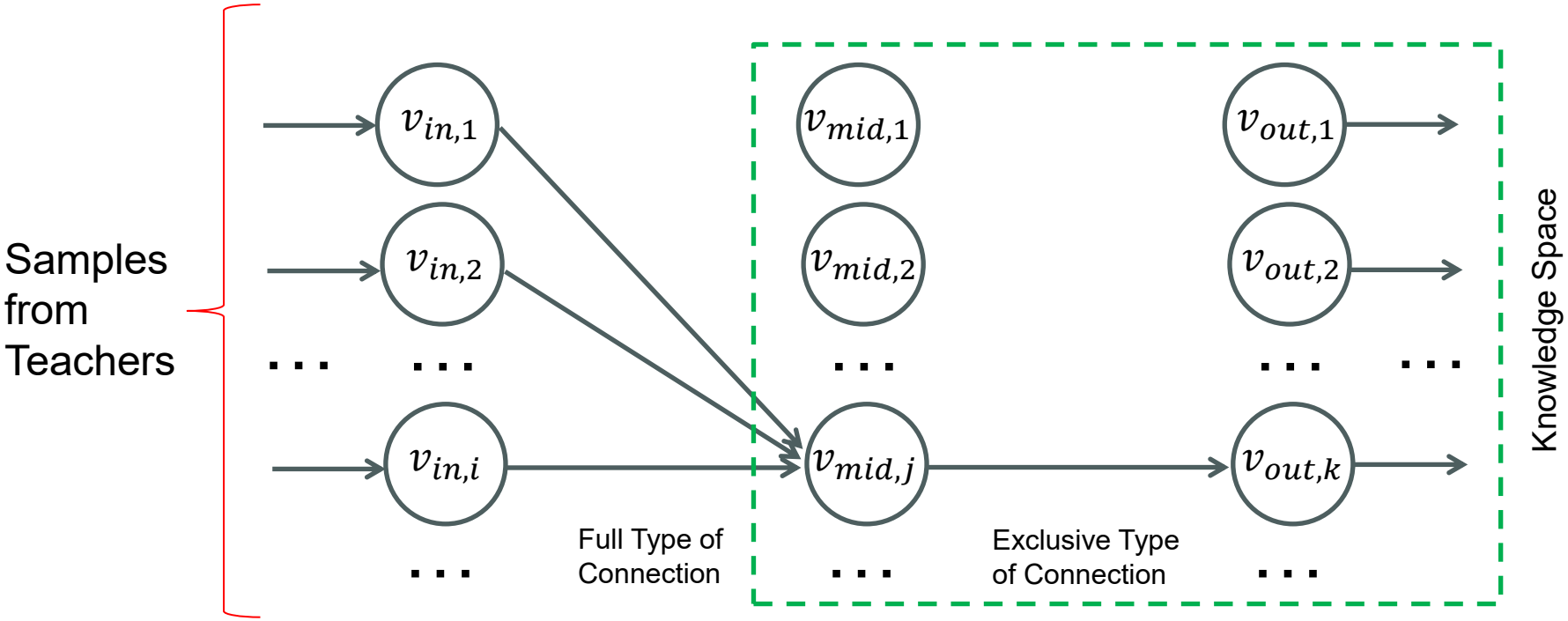
```
clc
mu = [64 0];
sigma = [5.0  0; 0  5.0];
a = 30:1.0:90 ;
b = -30:1.0:30 ;
[A, B] = meshgrid(a, b);
F = mvnpdf([A(:), B(:)], mu, sigma);
Fmax = max(F(:));
F = reshape(F/Fmax, length(a), length(b));
surf(a, b, F);
xlabel('Color Attribute: a');
ylabel('Color Attribute: b');
zlabel('Possibility in %');
```



Simplified Example (continued) …

- Represent the knowledge vector as:

$$v_{mid,j} = \{\mu_a, \mu_b, \sigma_a, \sigma_b, p_j, color\_name, others\}$$



Exercise

- We select a location on an image. Its RGB values are (30, 60, 210). Its eight neighbors' RGB values are:

  (40, 30, 200), (20, 50, 230), (33, 45, 205), (15, 35, 200)

  (80, 90, 230), (65, 30, 215), (35, 20, 190), (25, 15, 225)

  If we name this color as blue, what is its concept-physical representation in L*a*b color space?

$$v_{mid,j} = \{\mu_a, \mu_b, \sigma_a, \sigma_b, p_j, color\_name, others\}$$

Solution: MATLAB Program

```
clc
rgbtoxyz = [0.607  0.174  0.2; 0.299  0.587  0.114; 0.0  0.06  1.116];
xyz_max = rgbtoxyz*[256  256  256]';   ← Change it to [255  255  255]'
rgb=[30, 60, 210; 40, 30, 200; 20, 50, 230; 33, 45, 205;
    15, 35, 200; 80, 90, 230; 65, 30, 215; 35, 20, 190; 25, 15, 225];
[nbdata  nbelement] = size(rgb);
for i=1:nbdata
    xyz = rgbtoxyz*[rgb(i, 1)  rgb(i, 2)  rgb(i, 3)]';
    Lab(i,1) = 25.0*(100.0*xyz(2)/xyz_max(2))^(1/3) - 16.0 ;
    Lab(i,2) = 500.0*[(xyz(1)/xyz_max(1))^(1/3) - (xyz(2)/xyz_max(2))^(1/3)];
    Lab(i,3) = 200.0*[(xyz(2)/xyz_max(2))^(1/3) - (xyz(3)/xyz_max(3))^(1/3)];
end
Lab_mean = mean(Lab)
Lab_variance = var(Lab)
Lab_st = [sqrt(Lab_variance(1)), sqrt(Lab_variance(2)),sqrt(Lab_variance(3))]
```

Solution: Results

```
rgb =

    30    60   210
    40    30   200
    20    50   230
    33    45   205
    15    35   200
    80    90   230
    65    30   215
    35    20   190
    25    15   225
```

```
Lab =

   58.6402    6.0056  -56.2733
   52.3735   31.2209  -63.6544
   56.1555   10.7135  -66.0191
   55.4061   17.1521  -60.1535
   50.3377   15.9002  -67.2439
   69.6571   10.8885  -43.3331
   56.1574   41.1502  -61.5265
   48.4515   37.1687  -67.1953
   47.4895   41.4141  -79.0597
```

```
Lab_mean =

   54.9632   23.5126  -62.7177

Lab_variance =

   44.8548  200.6926   93.0828

Lab_st =

    6.6974   14.1666    9.6479
```

$\mu_b = -62.7177$

$\sigma_b = 9.6479$

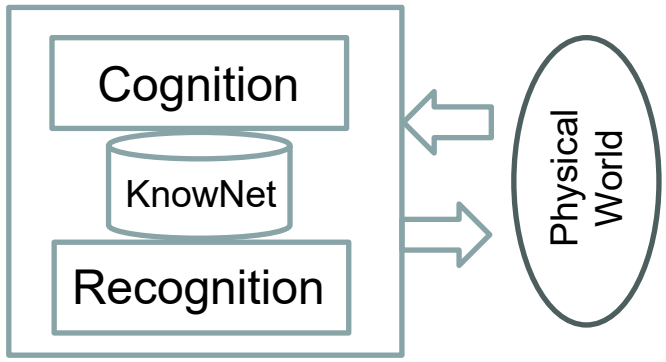
$\mu_a = 23.5126$

$\sigma_a = 14.1666$

$$p_j(a,b) = e^{-\frac{1}{2}\left(\frac{a-\mu_a}{\sigma_a}\right)^2} \times e^{-\frac{1}{2}\left(\frac{b-\mu_b}{\sigma_b}\right)^2}$$

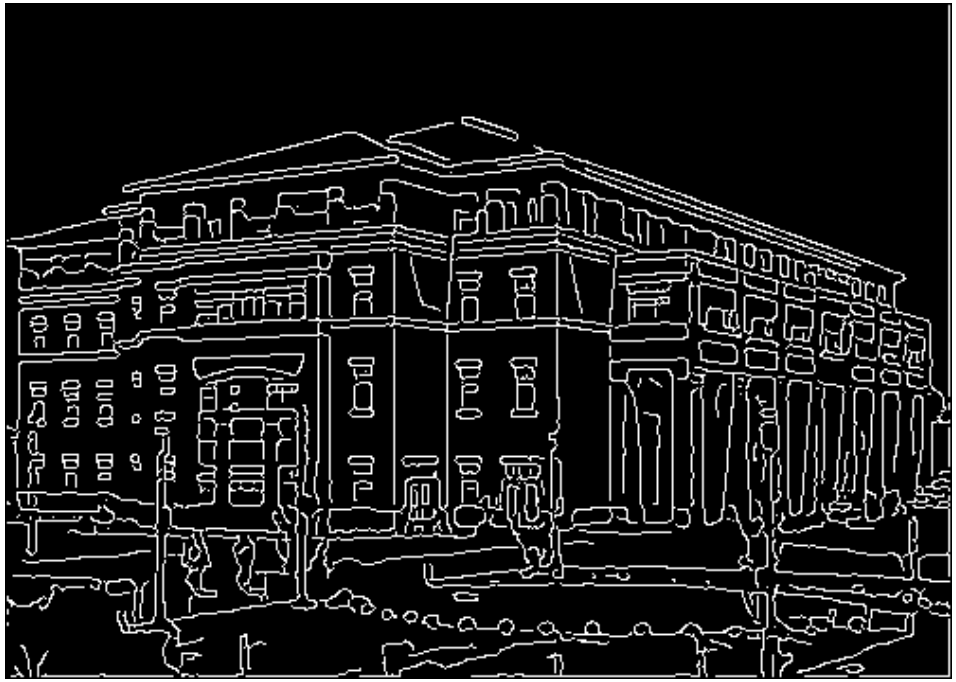
Outline of Lecture 3

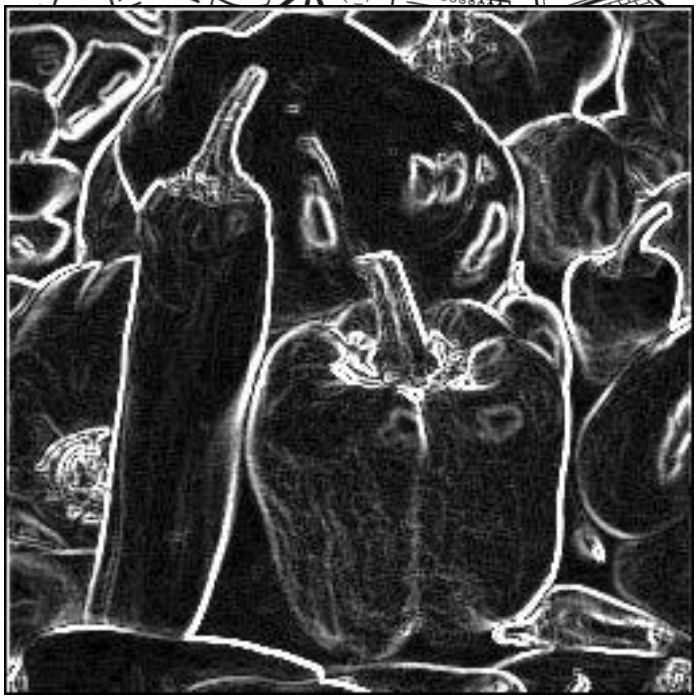
- Principle of Cognition (Learning)
- Principle of Artificial Neural Network
- Principle of RCE Neural Network
- Cognition of Colors
- Cognition of Curves
- Cognition of Pattern



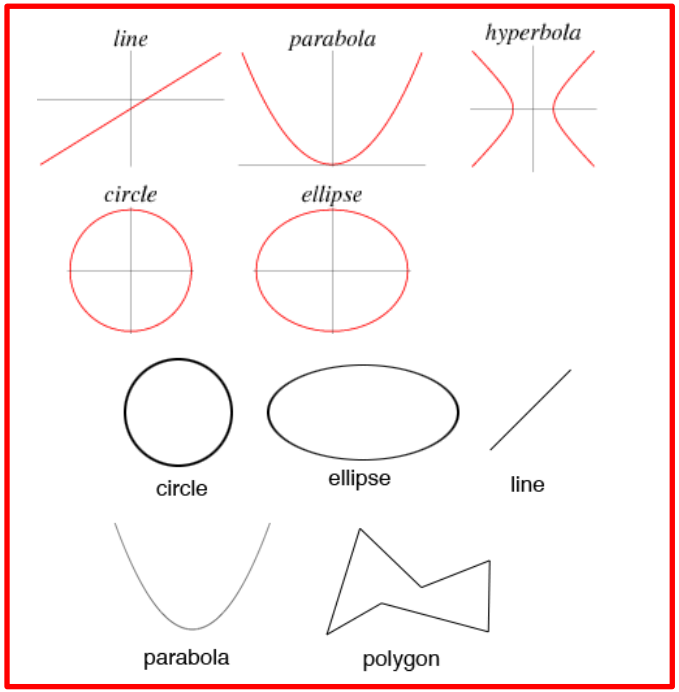
What is a curve?

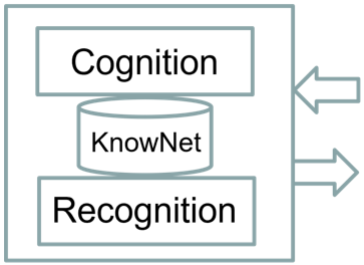
- A curve refers to linked edges inside a digital image.





Solution Toward Cognition and Recognition of Curves

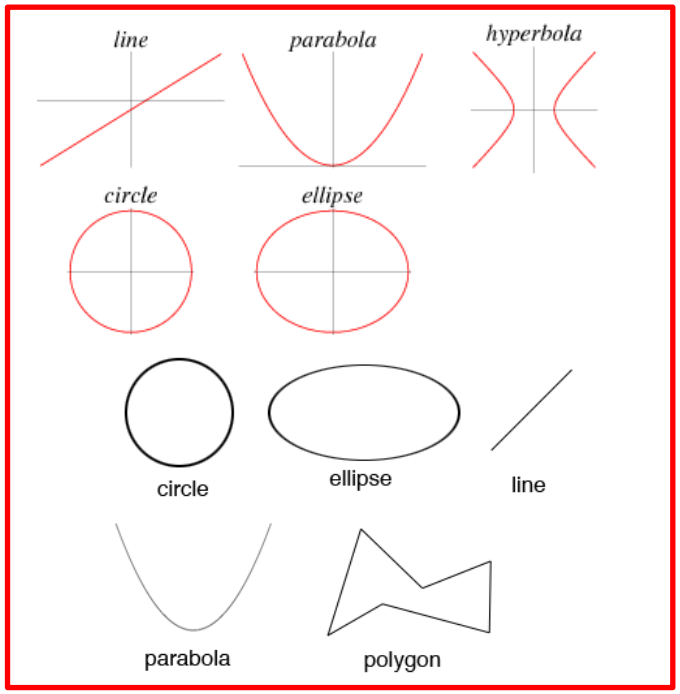




Input $v_{in,i} = \{r, g, b, I, L, a, b, u, v, \dots\}$

Images as 2D Matrices

- Curve Images for Training



Data Structure of Pixel

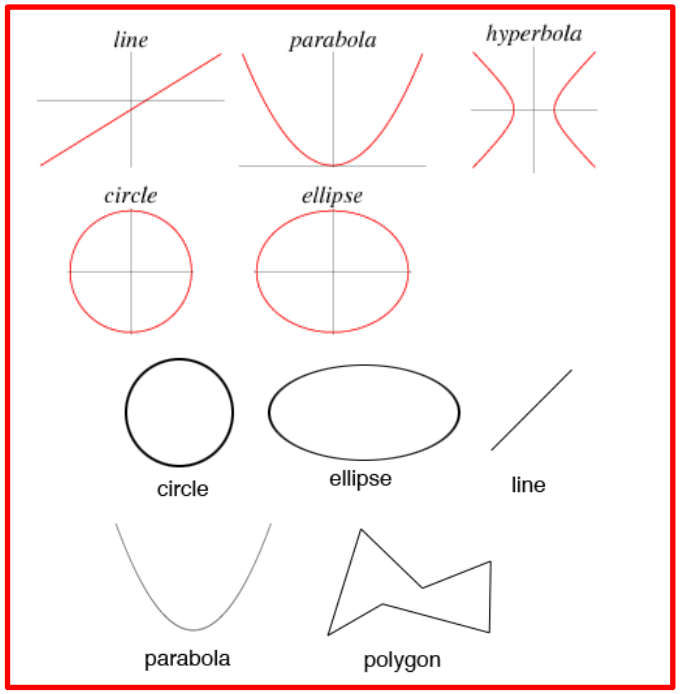
```
typedef   struct
{
        short     r, g, b, I ;
        short     L, a, b ;
        short     u, v ;
        double    x, y, z ;
}
Pixel;
```

Output

$$v_{mid,j} = \{\mu_j, \Sigma_j, p_j, conceptual\ labels, others\}$$

Images as 2D Matrices

- Curve Images for Training

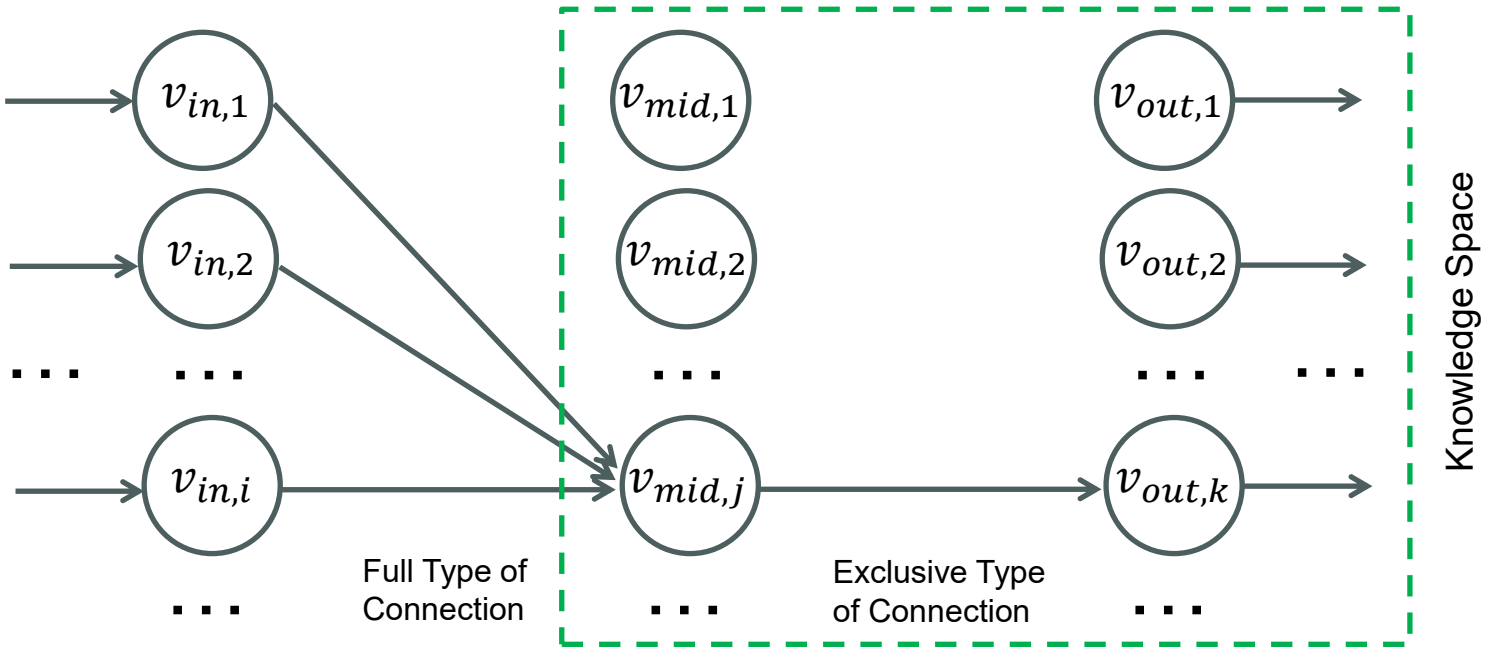


Data Structure of Pixel

```
typedef   struct
{
        short     r, g, b, I ;
        short     L, a, b ;
        short     u, v ;
        double    x, y, z ;
        string    curve_name ;
        double    feature_vector[ ];
}
Pixel;
```

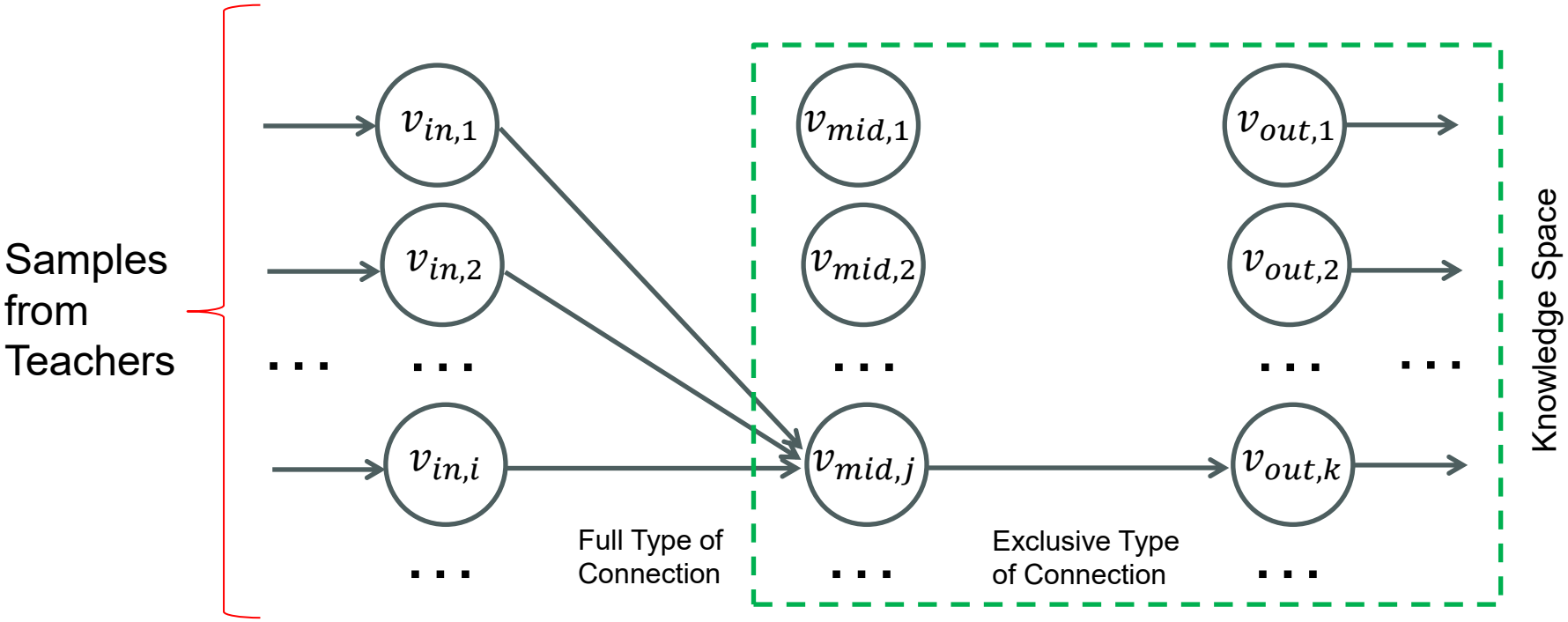
Question

- How to cognize or learn the curve that a list of pixels inside an image belongs to?



Step 1: To obtain visual samples (case of curves) from input image …

- Solution is to receive the training samples of curves from a teacher.



Step 2: To represent a curve by a vector

- Use of the coefficients of equations which describe the curves:

- Lines:

$$v = a \times u + b$$

- Circles:

$$\frac{(u - u_0)^2}{{r_0}^2} + \frac{(v - v_0)^2}{{r_0}^2} - 1 = 0$$

- Ellipses:

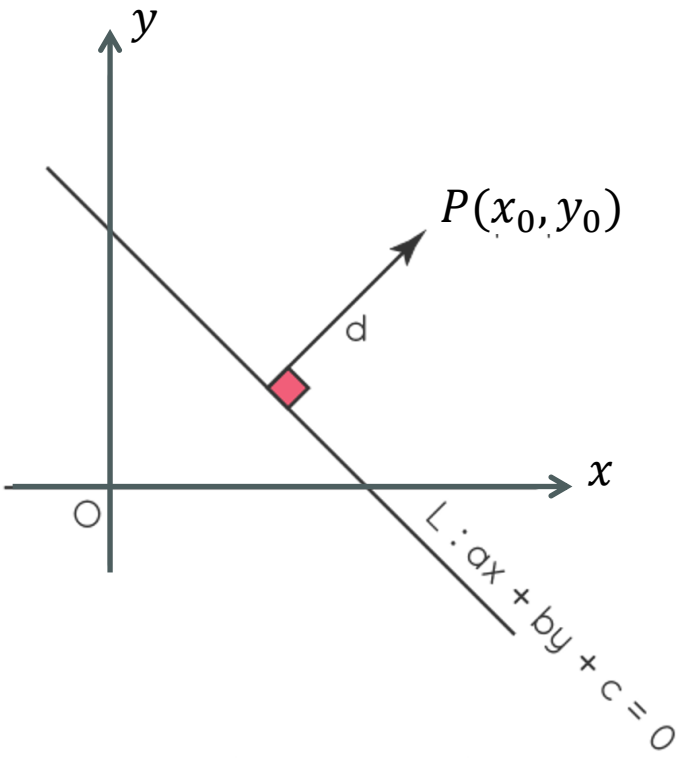
$$\frac{(u - u_0)^2}{r_u^2} + \frac{(v - v_0)^2}{r_v^2} - 1 = 0$$

Use of Polynomial Equations of Curves …

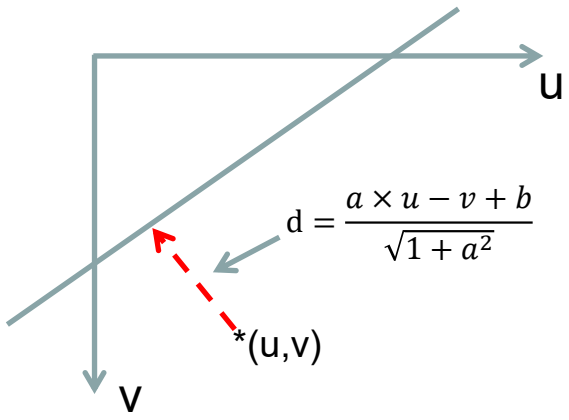
- Polynomials of Degree 2: $v = au^2 + bu + c$
- Polynomials of Degree 3: $v = au^3 + bu^2 + cu + d$
- Polynomials of Degree 4: $v = au^4 + bu^3 + cu^2 + du + e$
- Polynomials of Degree 5: $v = au^5 + bu^4 + cu^3 + du^2 + eu + f$
- etc.

What is the distance to a line?

$$\text{distance}(ax + by + c = 0, (x_0, y_0)) = \frac{|ax_0 + by_0 + c|}{\sqrt{a^2 + b^2}}$$

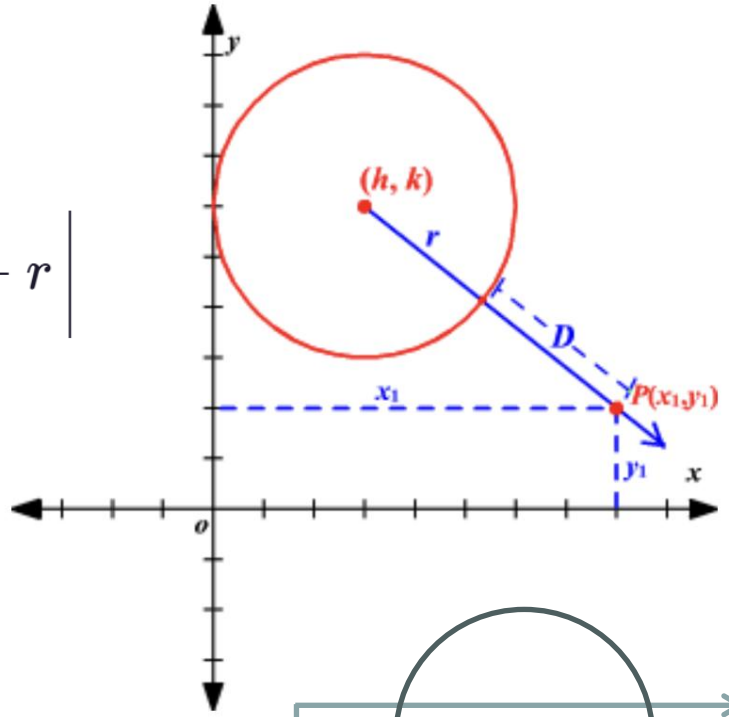


$$v = a \times u + b$$

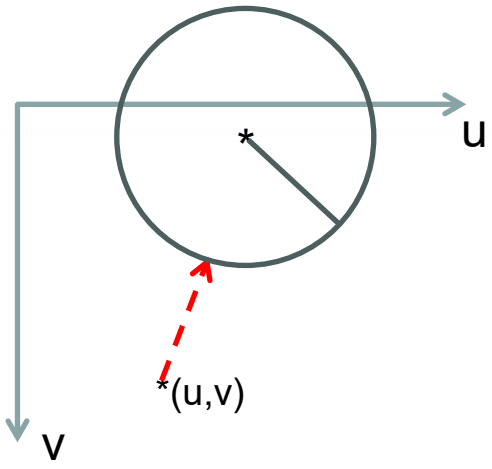


What is the distance to a circle?

$$distance = \left| \sqrt{(x_1 - h)^2 + (y_1 - k)^2} - r \right|$$



$$\frac{(u - u_0)^2}{{r_0}^2} + \frac{(v - v_0)^2}{{r_0}^2} - 1 = 0$$



$$\text{distance} = \sqrt{(u - u_0)^2 + (v - v_0)^2} - r_0$$

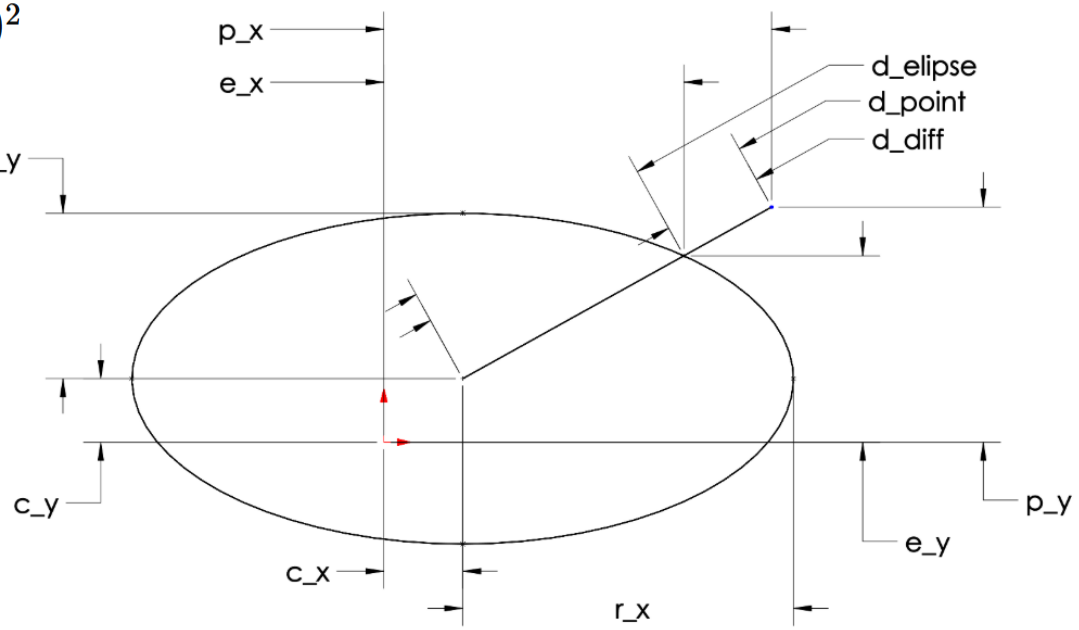
What is the distance to an ellipse?

- Distance of a Point to an Ellipse:

$$d_{point} = \sqrt{(p_x - c_x)^2 + (p_y - c_y)^2}$$

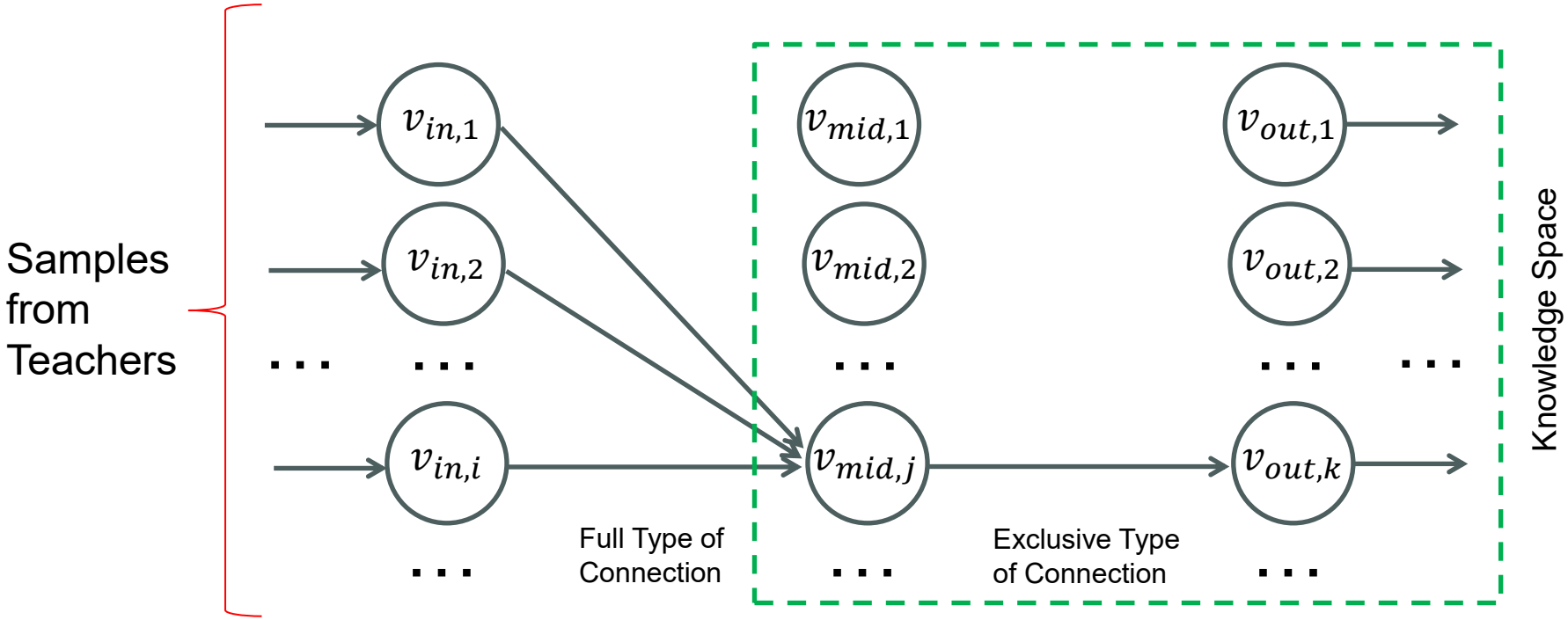
$$d_{ellipse} = \sqrt{\frac{r_x^2 \times r_y^2 \times ((p_x - c_x)^2 + (p_y - c_y)^2)}{(p_x - c_x)^2 \times r_y^2 + (p_y - c_y)^2 \times r_x^2}}$$

$$d_{diff} = d_{point} - d_{ellipse}$$



Step 3a: To compute the knowledge vector

$$v_{mid,j} = \{\mu_j, \Sigma_j, p_j, conceptual\ labels, others\}$$



Simplified Example …

- Receive a curve name and a set of edge pixels: $\{(u_i, v_i), i = 1, .., n\}$
- Represent the curve by an equation of line: $v = a \times u + b$

$$v_i = a \times u_i + b, i = 1, \dots, n$$

- Compute the coefficients of the equation of line:

$$U = \begin{bmatrix} u_1 & 1 \\ u_2 & 1 \\ \dots & \dots \\ u_n & 1 \end{bmatrix} \qquad V = \begin{bmatrix} v_1 \\ v_2 \\ \dots \\ v_n \end{bmatrix}$$

$$U \begin{bmatrix} a \\ b \end{bmatrix} = V$$

$$\begin{bmatrix} a \\ b \end{bmatrix} = (U^t U)^{-1}(U^t V)$$

Simplified Example (continued) …

- Compute the distances from $\{(u_i, v_i), i = 1, .., n\}$ to the line:

$$d_i = \frac{a \times u_i - v_i + b}{\sqrt{1 + a^2}}$$

- Compute the variance of $\{d_i, i = 1, \ldots, n\}$: $\sigma^2 = \frac{1}{n}\sum_{i=1}^{n} d_i^2$

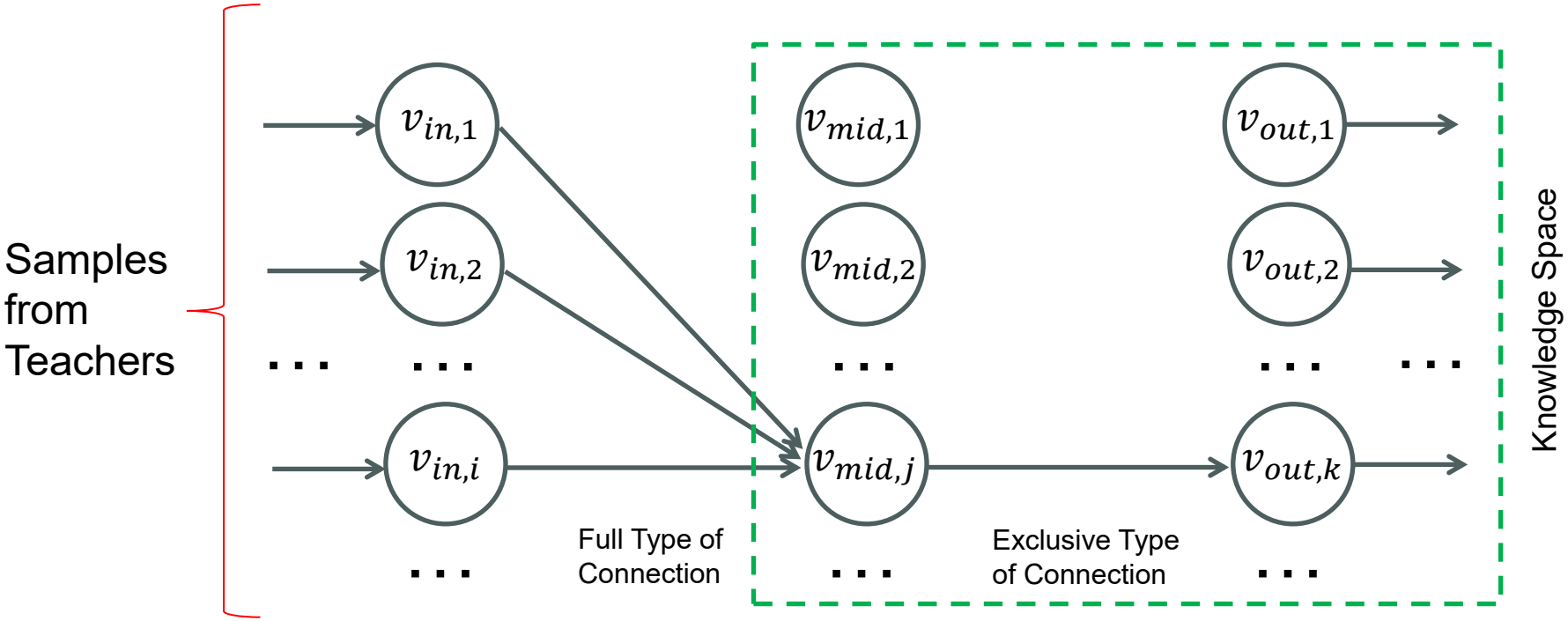
- Define the possibility function to be:

$$p_j(u, v) = e^{-\frac{1}{2}\left(\frac{a \times u - v + b}{\sigma\sqrt{1+a^2}}\right)^2}$$

Simplified Example (continued) …

- Represent the knowledge vector as:

$$v_{mid,j} = \{a, b, \sigma, p_j, curve\_name, others\}$$



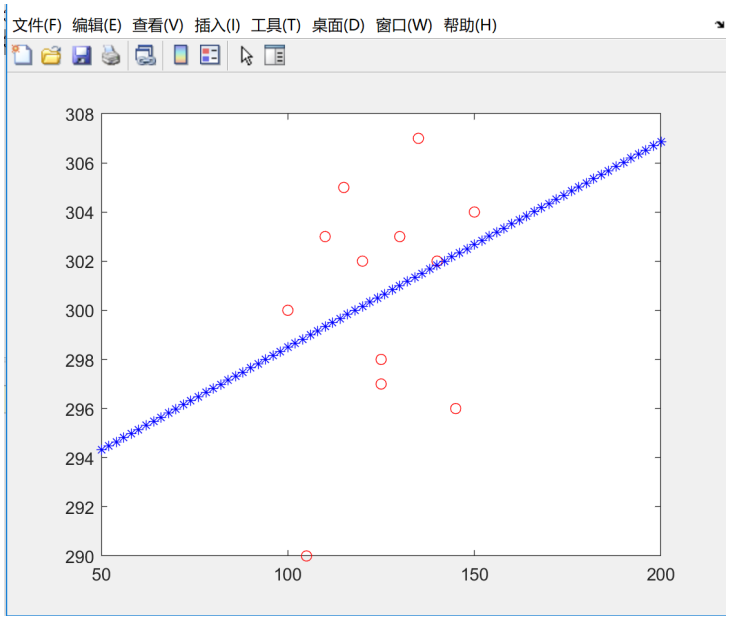
Exercise

- A contour in an image contains the following points:

  u = [100 ; 105; 110; 115; 120; 125; 125; 130; 135; 140; 145; 150]

  v = [300; 290; 303; 305; 302; 298; 297; 303; 307; 302; 296; 304]

  If these data form a straight line, what is its concept-physical representation? $v_{mid,j} = \{a, b, \sigma, p_j, curve\_name, others\}$

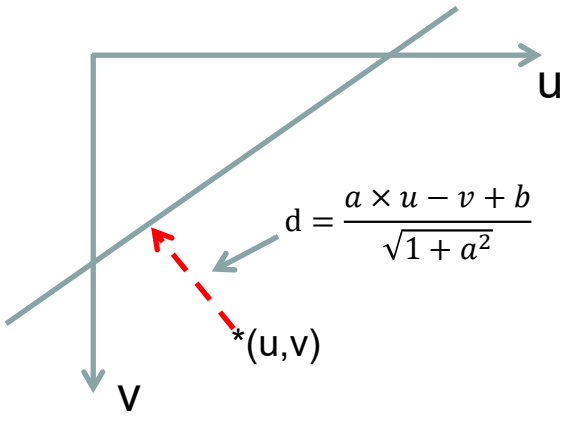


Solution: MATLAB Program

```
clc
xdata = [100 ; 105; 110; 115; 120; 125; 125; 130; 135; 140; 145; 150];
ydata = [300; 290; 303; 305; 302; 298; 297; 303; 307; 302; 296; 304];
AB = polyfit(xdata, ydata, 1);
xvalue = 50:2:200 ;
yvalue = polyval(AB, xvalue);
plot(xdata,ydata, 'ro', xvalue, yvalue,'b*');

for i = 1:length(xdata)
    error(i) = abs(AB(1)*xdata(i)-ydata(i)+AB(2))/sqrt(1+AB(1)^2);
end
mu = mean(error)
a = AB(1)
b = AB(2)
sigma = sqrt(cov(error))
```

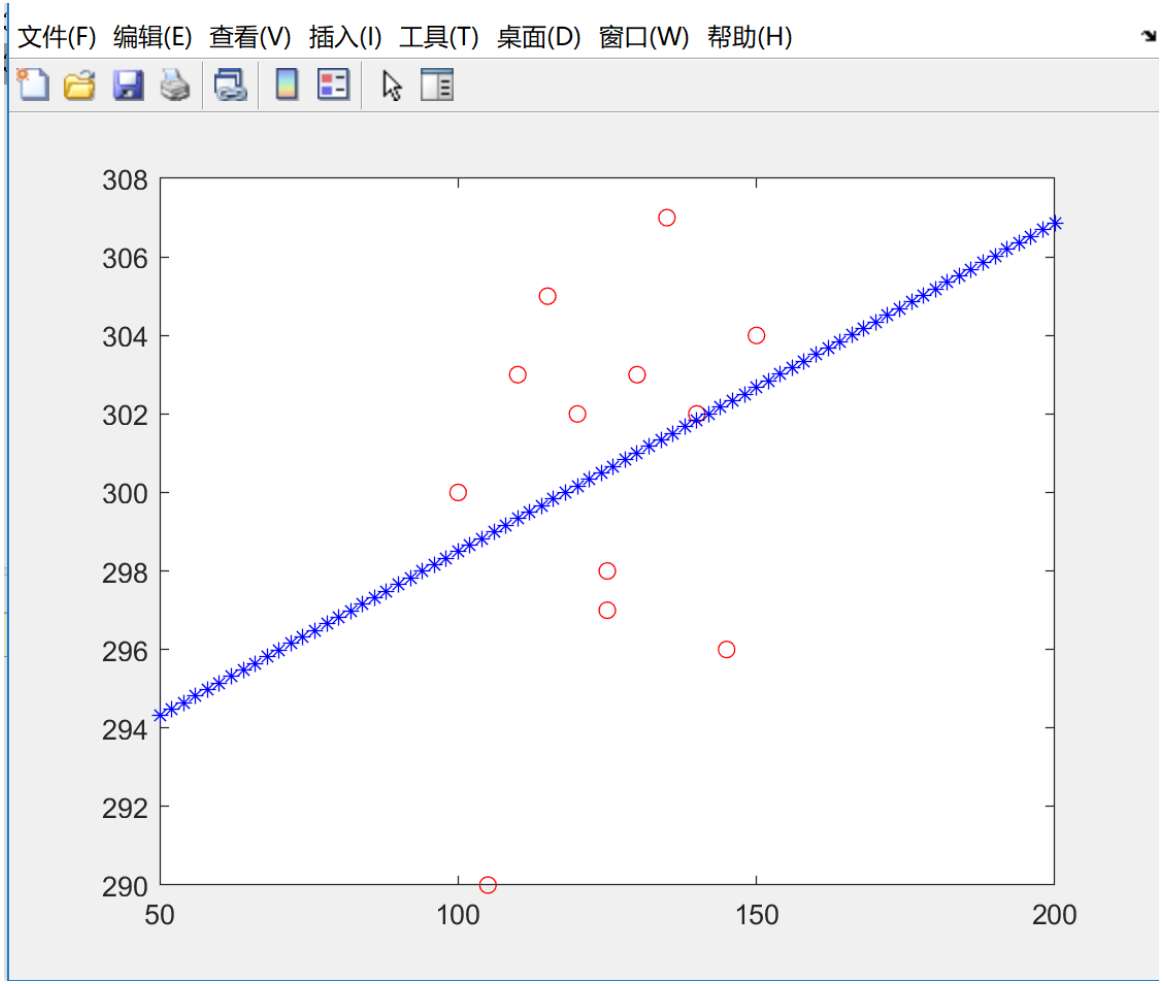
Remove “abs”

Why?



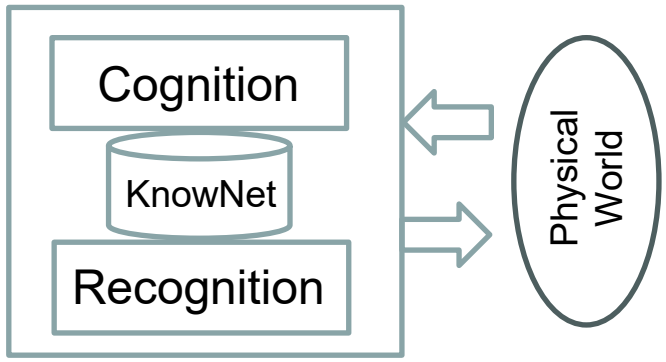
Solution: Results

```
a =

    0.0836


b =

  290.1288


sigma =

    2.5134
```



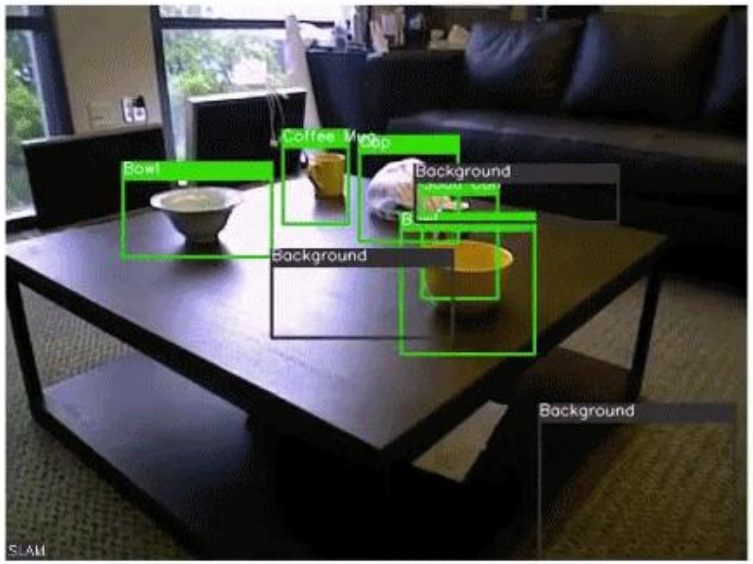
Outline of Lecture 3

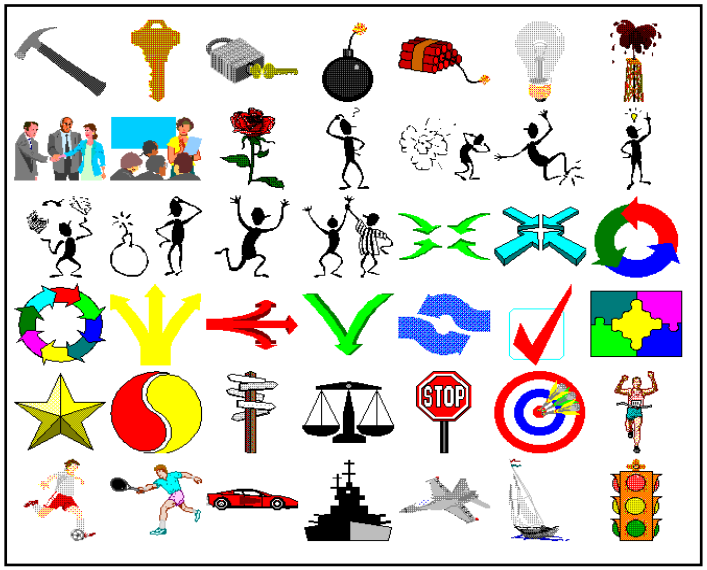
- Principle of Cognition (Learning)
- Principle of Artificial Neural Network
- Principle of RCE Neural Network
- Cognition of Colors
- Cognition of Curves
- Cognition of Pattern

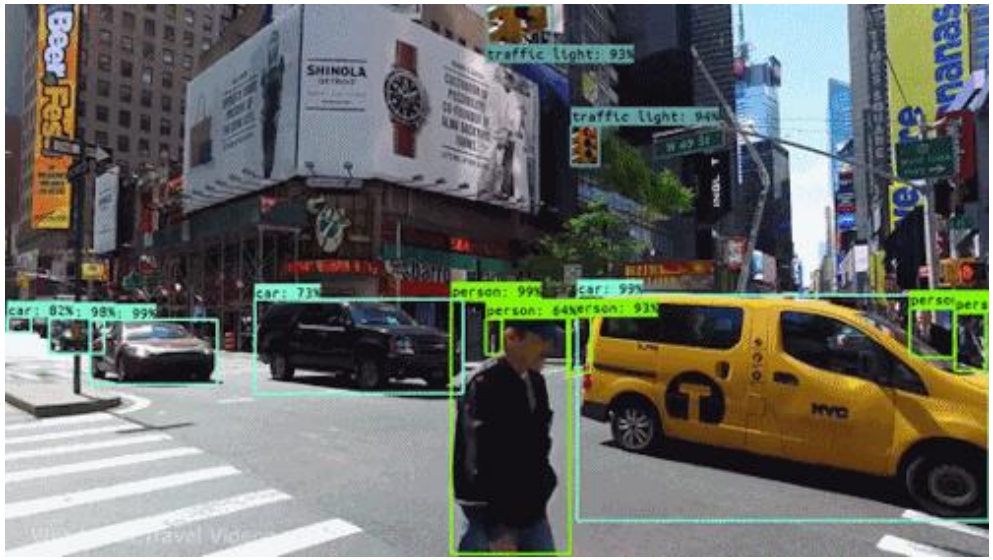


What is a pattern?

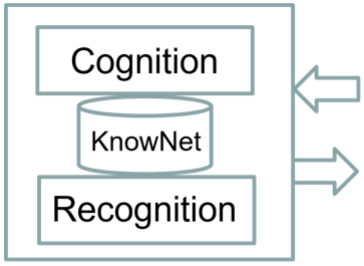
- A pattern inside an image refers to the appearance, shape, or outlook of any visible entity in the physical world or the conceptual worlds (e.g. human-created characters, graphics, drawings, etc).







Solution Toward Cognition and Recognition of Patterns



Input

$$v_{in,i} = \{r, g, b, I, L, a, b, u, v, \ldots\}$$

Images as 2D Matrices

- Pattern Images for Training

Data Structure of Pixel

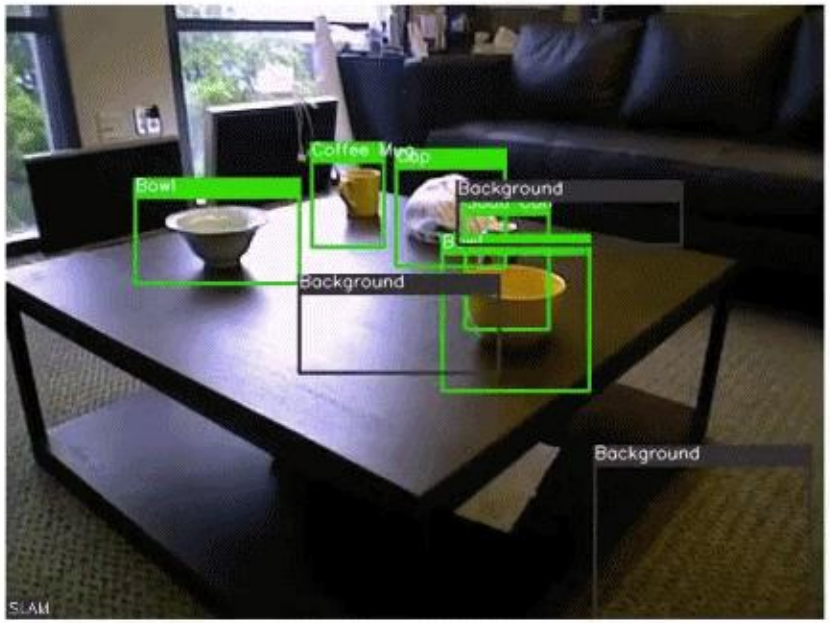
```
typedef   struct
{
        short     r, g, b, I ;
        short     L, a, b ;
        short     u, v ;
        double    x, y, z ;
}
Pixel;
```

Output

$$v_{mid,j} = \{\mu_j, \Sigma_j, p_j, conceptual\ labels, others\}$$

Images as 2D Matrices

- Pattern Images for Training

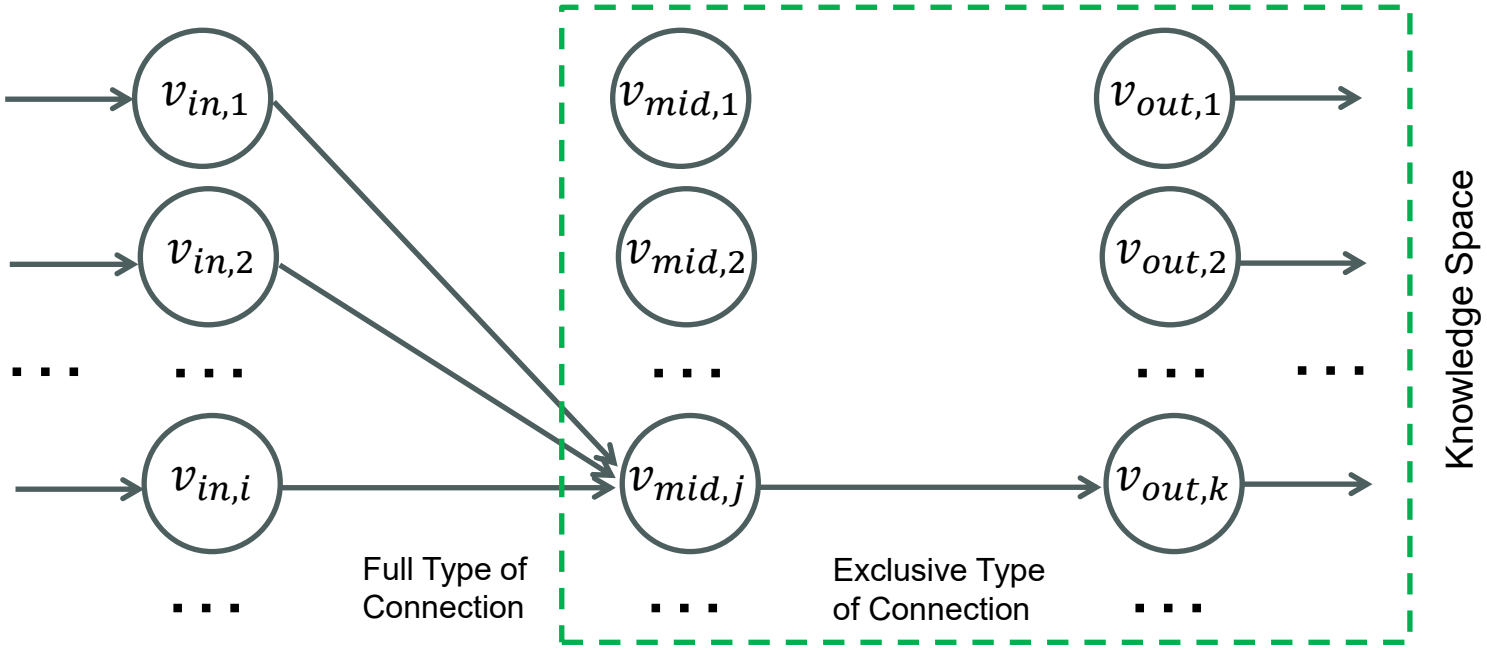


Data Structure of Pixel

```
typedef   struct
{
        short     r, g, b, I ;
        short     L, a, b ;
        short     u, v ;
        double    x, y, z ;
        string    pattern_name ;
        double  feature_vector[ ];
}
Pixel;
```

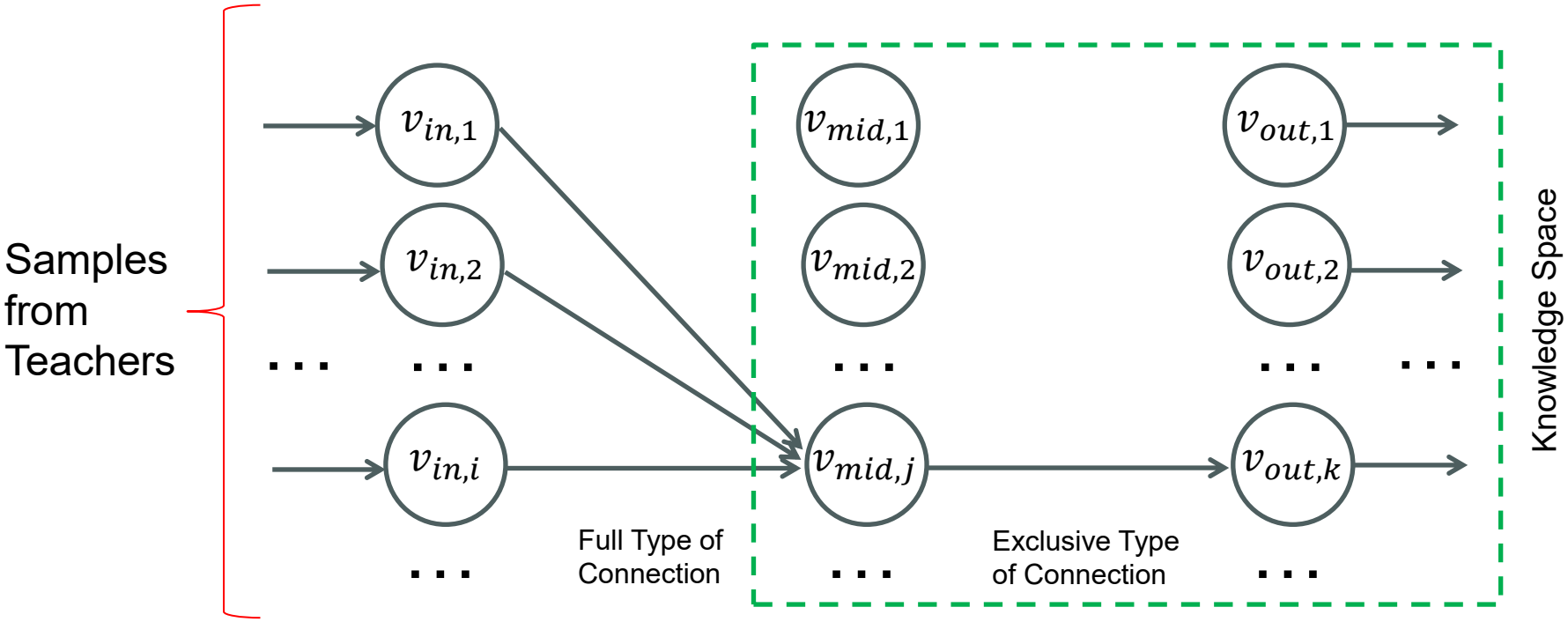
Question

- How to cognize or learn the pattern that a sub-image centered at a pixel belongs to?



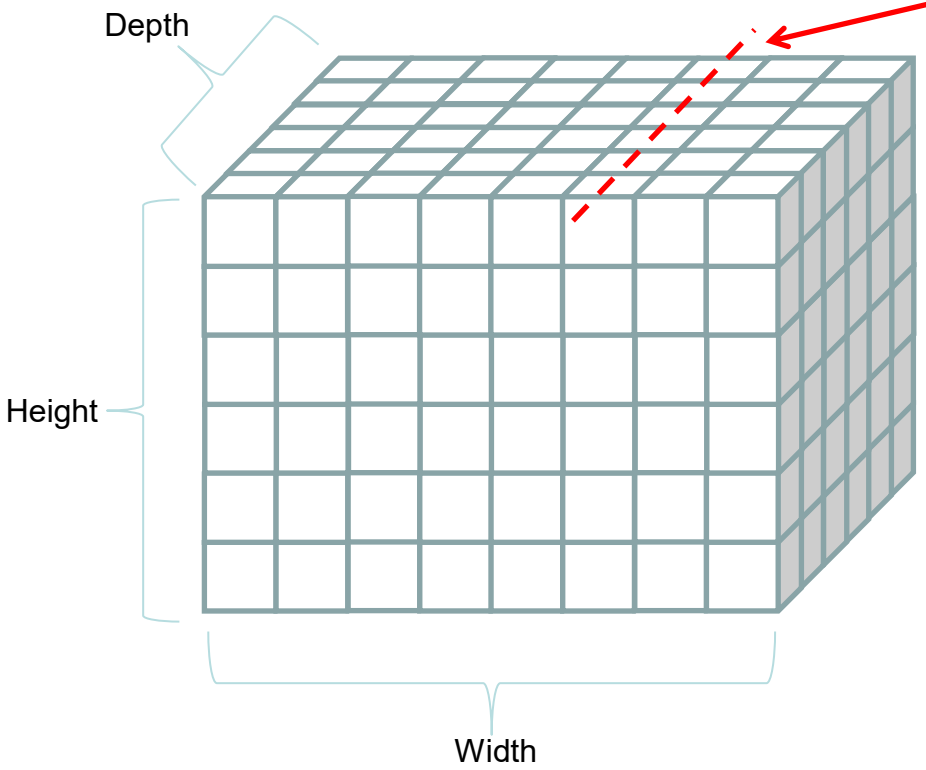
Step 1: To obtain visual samples (case of patterns) from input image …

- Solution is to receive the training samples of patterns from a teacher.



Step 2: To represent a pattern by a vector

- Increase the depth of visual pattern by doing Deep Convolution or Fourier Transform.
- Convert the 2D matrix of pixels' feature vectors into a single vector which is called Apparent/Deep Feature Vectors.



This series of values belongs to a pixel's feature vector

Size of visual pattern: N x M

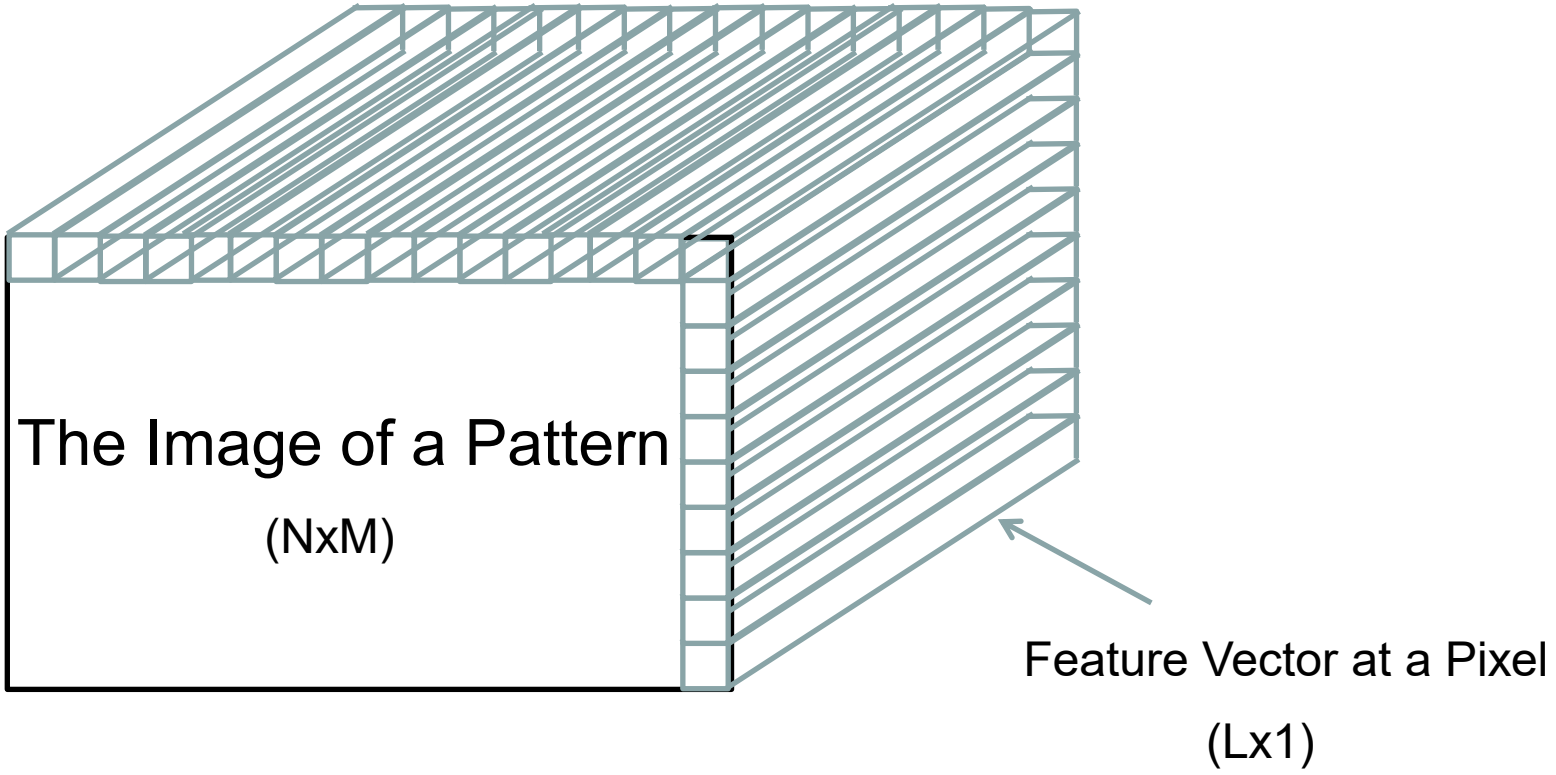
Size of a Pixel's Feature Vector: L x 1

Size of a Pattern's Feature Vector:

$$(N \times M \times L) \times 1$$

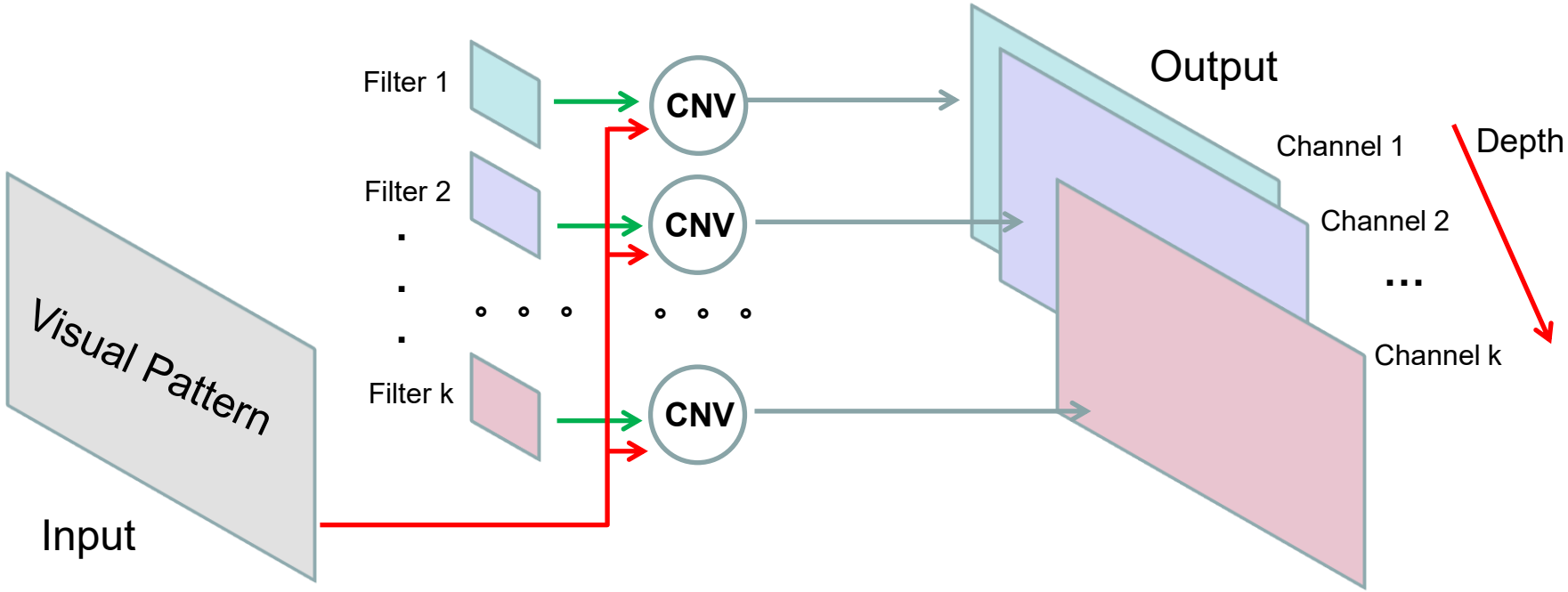
Definition of Apparent/Deep Feature Vector at Pixel

- An apparent/deep feature vector, or feature vector in short, of a pixel refers to a vector which contains all the meaningful data computed at the location of a pixel.



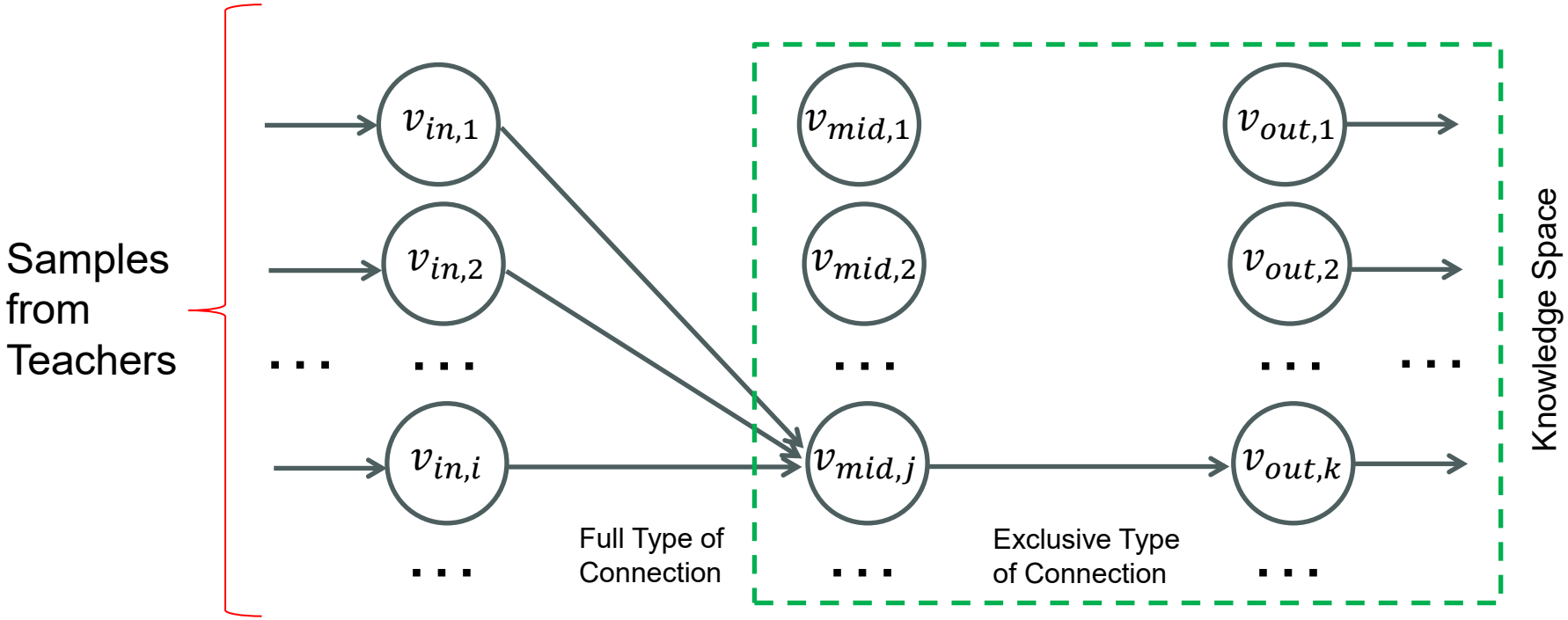
How to compute a feature vector at a pixel?

- A feature vector at pixel can be obtained in many different ways such as:
  - From intensities
  - From colors
  - From Fourier Transform
  - From lists of convolutions (i.e. Deep Convolution)



Step 3a: To compute the knowledge vector

$$v_{mid,j} = \{\mu_j, \Sigma_j, p_j, conceptual\ labels, others\}$$



Simplified Example …

Consider that these vectors belong to a hyper sphere

- Receive a pattern's name and a set of its training data: $\{f_i, i = 1, .., n\}$

- Compute the center and radius-variance of the hyper-sphere:

$$\mu_f = \frac{1}{n}\sum_{i=1}^{n} f_i \qquad \sigma_f^2 = \frac{1}{n}\sum_{i=1}^{n} (f_i - \mu_f)^t (f_i - \mu_f)$$

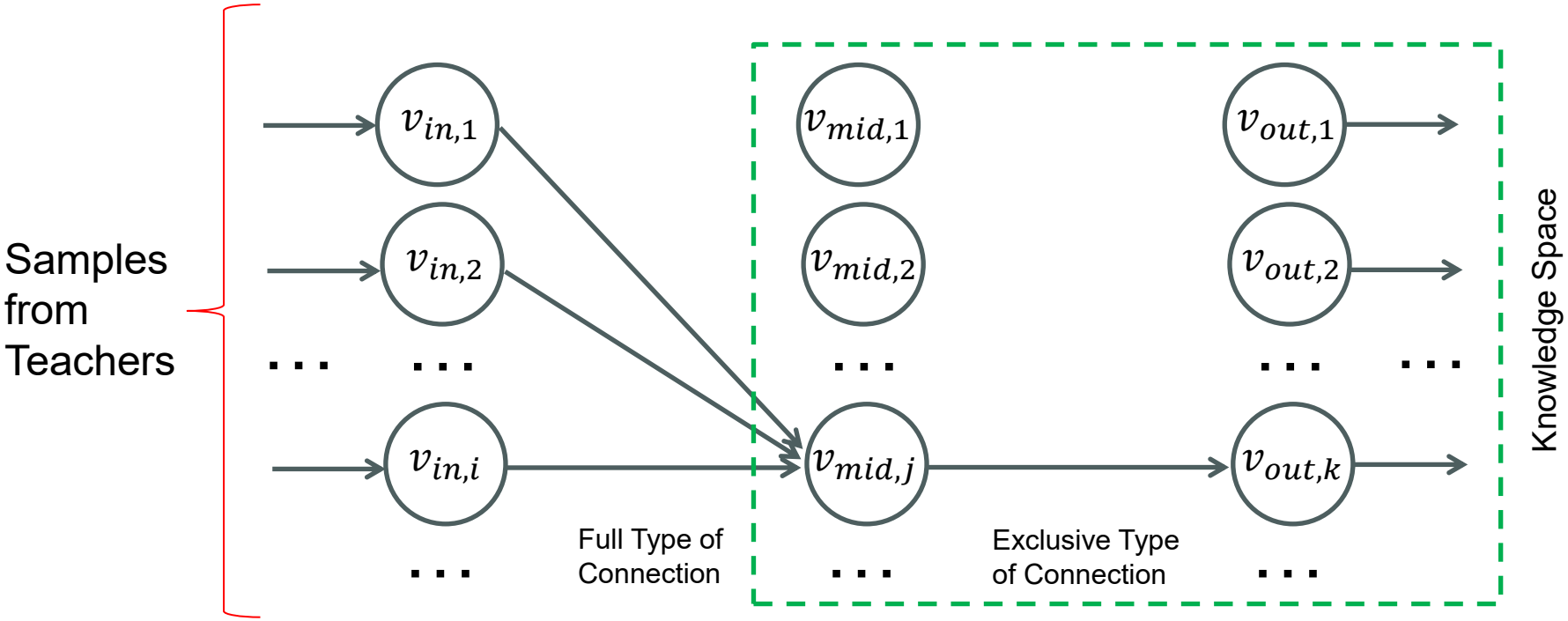
- Define the possibility function to be:

$$P_j(f) = e^{-\frac{1}{2\sigma_f^2}(f-\mu_f)^t (f-\mu_f)}$$

Simplified Example (continued) …

- Represent the knowledge vector as:

$$v_{mid,j} = \{\mu_f, \sigma_f, p_j, pattern\_name, others\}$$



Exercise

- We have three facial images of an object (i.e. images in grayscale), which are given below. If the pattern's name is "rectangle", what is the concept-physical representation of the pattern? (assume that we do not use histogram of gradients)

	0	1	2	3	4	5
0	150	150	150	150	150	150
1	150	50	50	50	50	150
2	150	50	50	50	50	150
3	150	50	50	50	50	150
4	150	50	50	50	50	150
5	150	50	50	50	50	150
6	150	50	50	50	50	150
7	150	50	50	50	50	150
8	150	150	150	150	150	150

Pattern's Training Data 1

	0	1	2	3	4	5
0	150	150	150	150	150	150
1	150	60	60	60	60	150
2	150	60	60	60	60	150
3	150	60	60	60	60	150
4	150	60	60	60	60	150
5	150	60	60	60	60	150
6	150	60	60	60	60	150
7	150	60	60	60	60	150
8	150	150	150	150	150	150

Pattern's Training Data 2

	0	1	2	3	4	5
0	150	150	150	150	150	150
1	150	70	70	70	70	150
2	150	70	70	70	70	150
3	150	70	70	70	70	150
4	150	70	70	70	70	150
5	150	70	70	70	70	150
6	150	70	70	70	70	150
7	150	70	70	70	70	150
8	150	150	150	150	150	150

Pattern's Training Data 3

Solution in MATLAB:

Input of Training Data to MATLAB

```
image1 = [150  150  150  150  150  150
          150   40   40   40   40  150
          150   40   40   40   40  150
          150   40   40   40   40  150
          150   40   40   40   40  150
          150   40   40   40   40  150
          150   40   40   40   40  150
          150   40   40   40   40  150
          150   40   40   40   40  150
          150  150  150  150  150  150];

image2 = [150  150  150  150  150  150
          150   60   60   60   60  150
          150   60   60   60   60  150
          150   60   60   60   60  150
          150   60   60   60   60  150
          150   60   60   60   60  150
          150   60   60   60   60  150
          150   60   60   60   60  150
          150   60   60   60   60  150
          150  150  150  150  150  150];

image3 = [150  150  150  150  150  150
          150   80   80   80   80  150
          150   80   80   80   80  150
          150   80   80   80   80  150
          150   80   80   80   80  150
          150   80   80   80   80  150
          150   80   80   80   80  150
          150   80   80   80   80  150
          150   80   80   80   80  150
          150  150  150  150  150  150];
```

Solution: MATLAB Program

```
42 —    n = 3 ;
43 —    for i=1:9
44 —        for j=1:6
45 —            f1((i-1)*6+j) = image1(i, j);
46 —            f2((i-1)*6+j) = image2(i, j);
47 —            f3((i-1)*6+j) = image3(i, j);
48 —        end
49 —    end
50 —    f1 = f1'; % Change to column vector
51 —    f2 = f2'; % Change to column vector
52 —    f3 = f3'; % Change to column vector
53
54 —    mu_f = (1/n)*(f1+f2+f3); % mean of three columns vectors
55 —    disp(mu_f');
56 —    e1 = f1 - mu_f; % error vector from image 1
57 —    e2 = f2 - mu_f; % error vector from image 2
58 —    e3 = f3 - mu_f; % error vector from image 3
59 —    variance_f = (1/n)*(e1'*e1+e2'*e2+e3'*e3);
60 —    sigma_f = sqrt(variance_f)
```

Convert Training Data into Vectors.

Compute Three Error Vectors

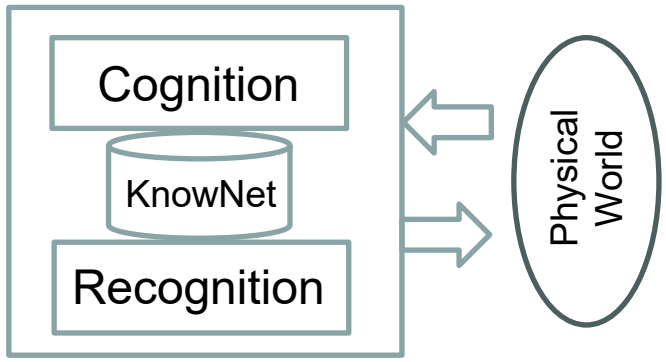
Solution: Results

```
mu_f =

  Elements 1 to 17

    150   150   150   150   150   150   150    60    60    60    60   150   150

     60    60    60    60

  Elements 18 to 34

    150   150    60    60    60    60   150   150    60    60    60    60   150

    150    60    60    60

  Elements 35 to 51

     60   150   150    60    60    60    60   150   150    60    60    60    60

    150   150   150   150

  Elements 52 to 54

    150   150   150


sigma_f =

    86.4099
```

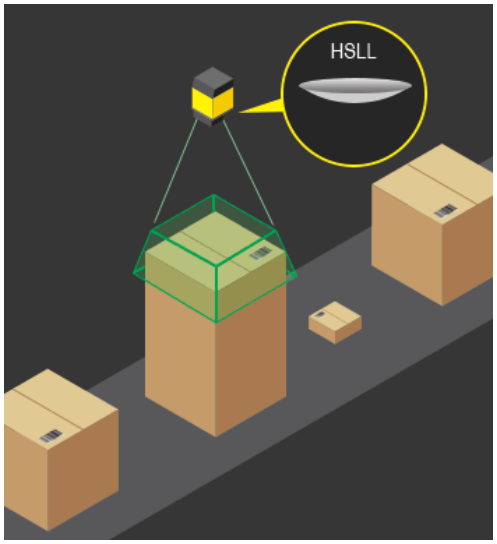
$$P_j(f) = e^{-\frac{1}{2\sigma_f^2}(f-\mu_f)^t\,(f-\mu_f)}$$

Summary of Lecture 3

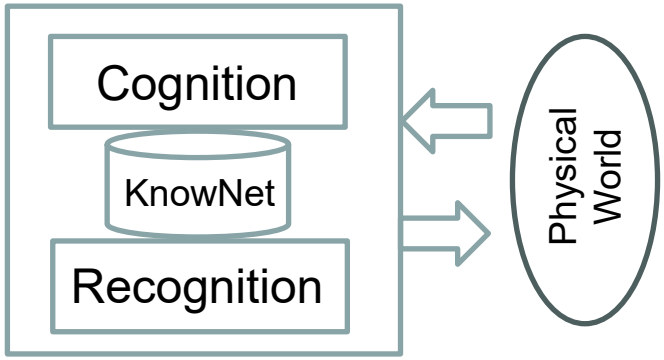
- Principle of Cognition (Learning)
- Principle of Artificial Neural Network
- Principle of RCE Neural Network
- Cognition of Colors
- Cognition of Curves
- Cognition of Pattern



Outline of Module 4



- Lecture 1:
  - Geometry-Driven Computation
- Lecture 2:
  - Fuzziness-Driven Computation
- Lecture 3:
  - Cognition-Driven Computation
- Lecture 4:
  - Recognition-Driven Computation
- Lecture 5:
  - Computation Using Monocular Vision
- Lecture 6:
  - Computation Using Binocular Vision



NANYANG TECHNOLOGICAL UNIVERSITY

School of Mechanical & Aerospace Engineering

Design, Machine, Control, Intelligence

Lecture 4 of Module 4

AI 3.0

MA4829 Machine Intelligence

Recognition-Driven Computation

XIE Ming, PhD (France)

http://personal.ntu.edu.sg/mmxie

"We are living inside an ocean of signals"

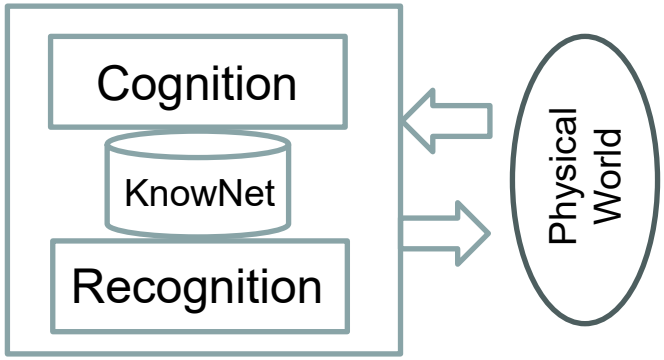
What is a system with intelligence inside?

Outline of Lecture 4

- Principle of Recognition (Understanding)
- Principle of Artificial Neural Network
- Principle of RCE Neural Network

Revision

- Recogniton of Colors
- Recognition of Curves
- Recognition of Patterns
- Practices in MATLAB

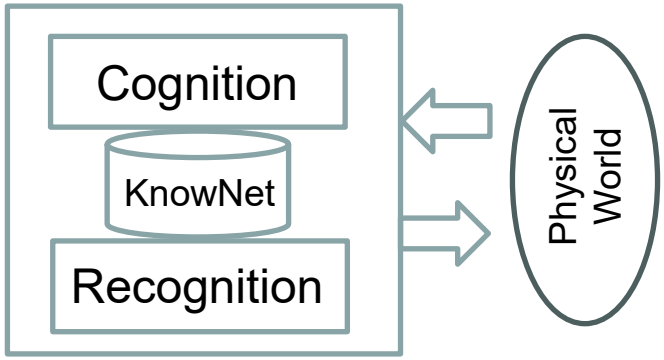


Outline of Lecture 4

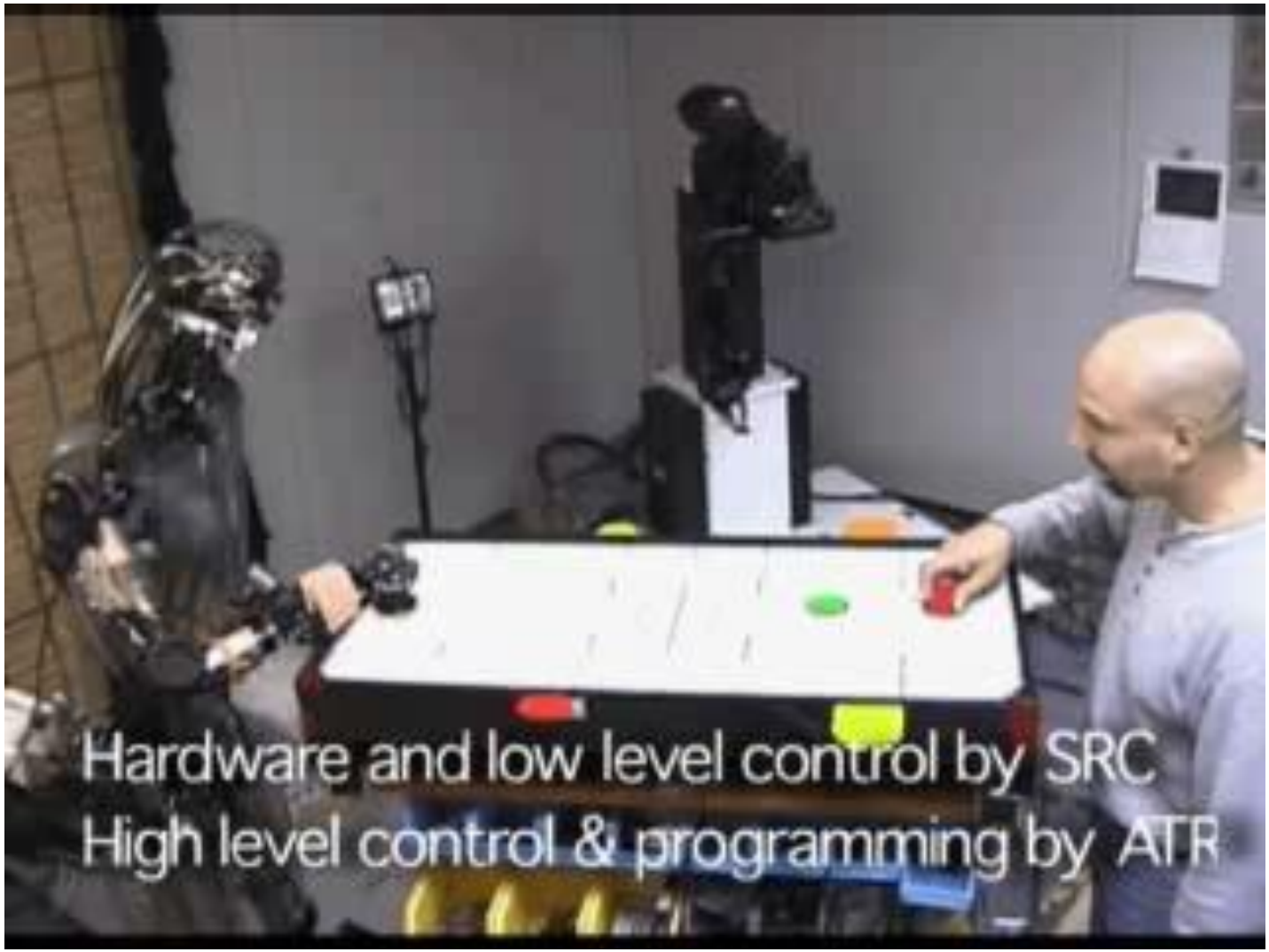
- Principle of Recognition (Understanding)
- Principle of Artificial Neural Network
- Principle of RCE Neural Network

Revision

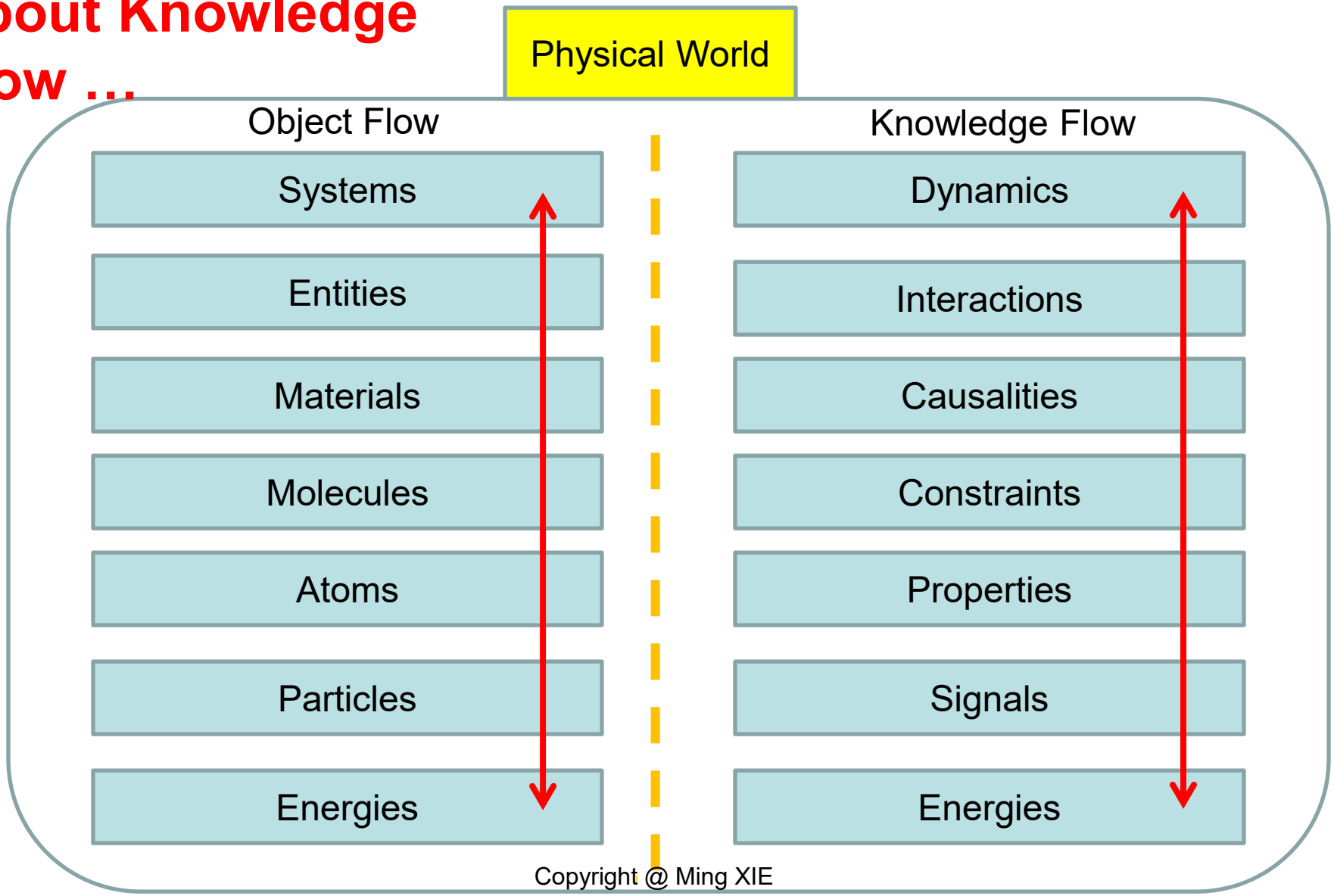
- Recogniton of Colors
- Recognition of Curves
- Recognition of Patterns
- Practices in MATLAB



Recognition enables autonomy …



About Knowledge Flow …



How to recognize or understand conceptual knowledge?

Answer:

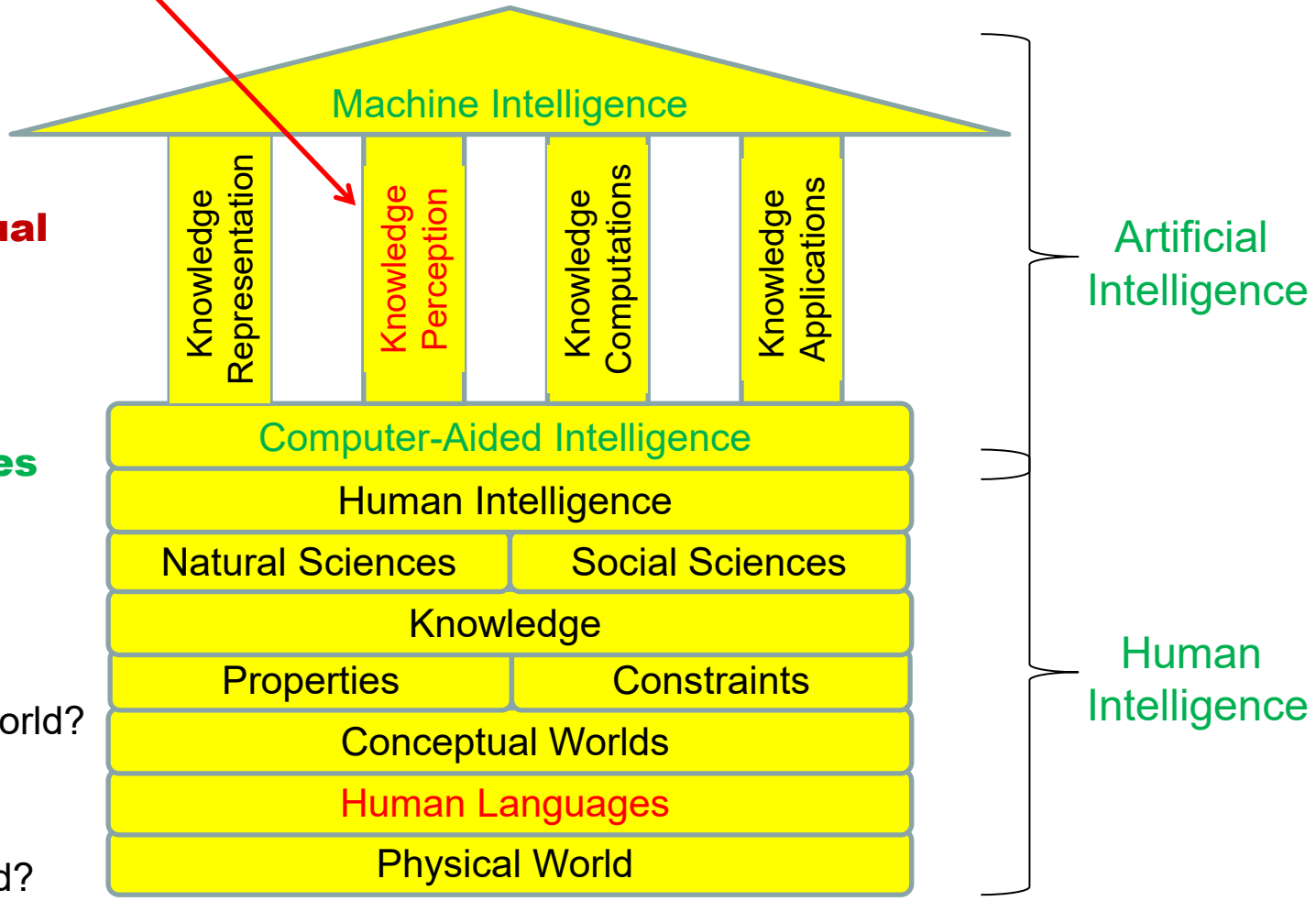
The mental capability of transforming visual signals into knowledge.

Such mental capability includes visual cognition and recognition.

What is conceptual world?

?

What is physical world?



Scenario of Visual Recognition

Cognition

Teacher: This is a blue color.
Machine: Got it. It is a blue color.

Re-cognition

Teacher: What is this color?
Machine: It is a blue color.

What is recognition in general?

- Recognition is to re-cognize anything which is known before. Recognition has to be unsupervised.

  - What is this place?
  - What is the building in the foreground?
  - What is the building in the middle of the background?

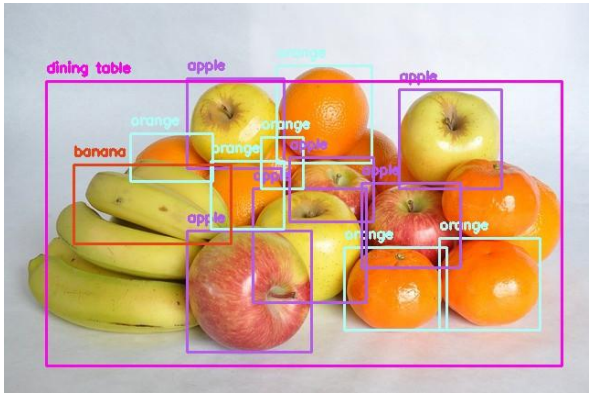
Definition of Recognition

Definition

- It is a mental process of cumulatively remembering familiar knowledge about familiar entities.

Similar Terms

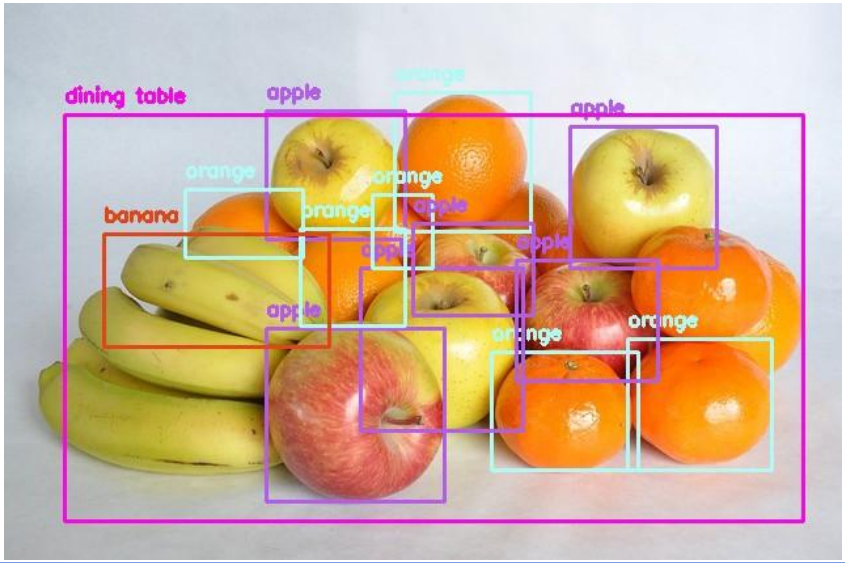
- Symbol Grounding
- Pattern Matching
- Unsupervised Classification



What to recognize from visual signals?

- Photometric Knowledge of Entities from the Physical World
  - Luminance-related Appearances
  - Chrominance-related Appearances

- Geometrical Knowledge of Entities from the Physical World
  - Points
  - Lines
  - Curves
  - Shapes
  - Surfaces
  - Objects
  - Exploded Drawings, etc.

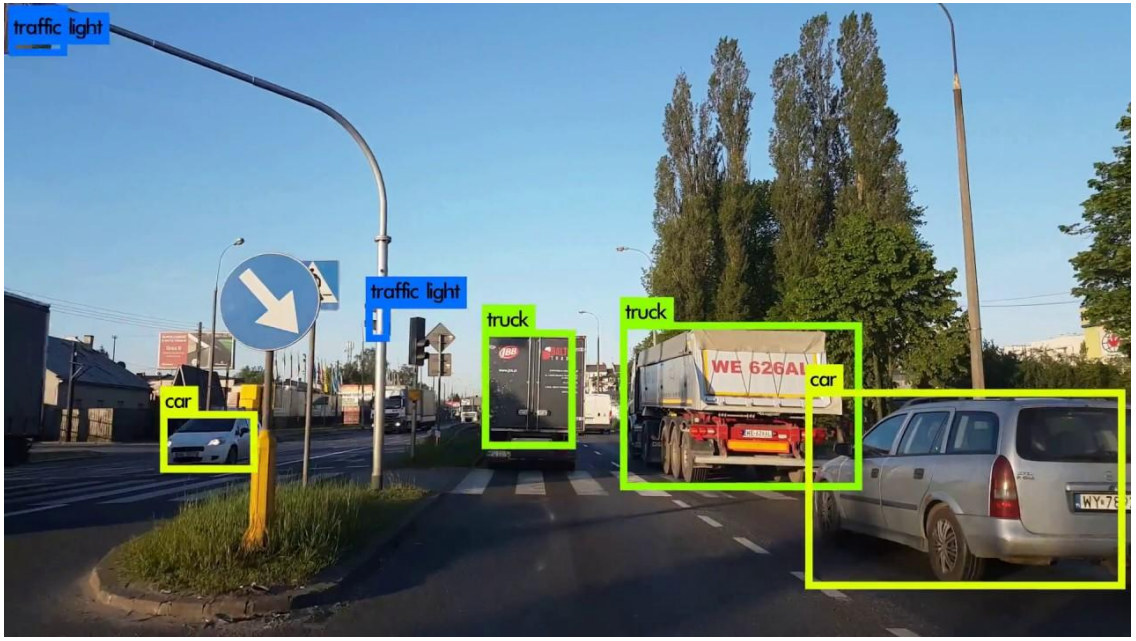


Terminology Alerts

- Objects
  - Objects are the entities inside the physical world.
- Images
  - Images are the projections of scenes onto cameras.
- Colors
  - Colors are the coordinates inside a color space.
- Curves
  - Curves are the linked edges inside a digital image.
- Patterns
  - Patterns are the shapes or outlooks of individual objects.

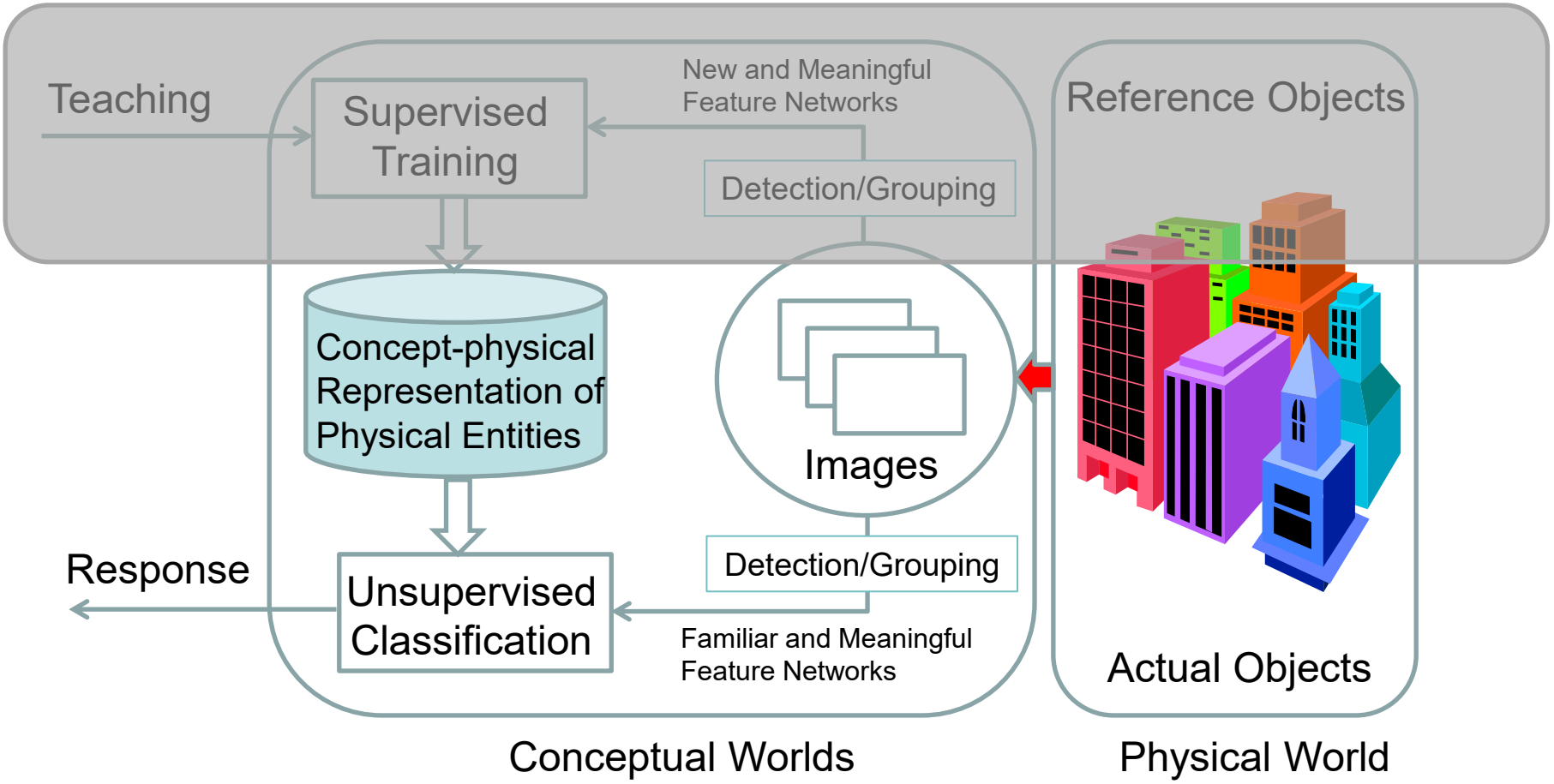
How to recognize familiar and meaningful features from visual signals?

- Step 1: Detection of familiar and Meaningful Features
- Step 2: Grouping of Familiar and Meaningful Features into Networks
- Step 3: Unsupervised Classification of Feature Networks



How to Design Visual Recognition Systems?

- Follow this generic blueprint …



What is the difference between cognition and recognition?

Cognition

- Feature Detection and Grouping
- Knowledge Representation
- Teaching/Learning
- Memorization inside Memory

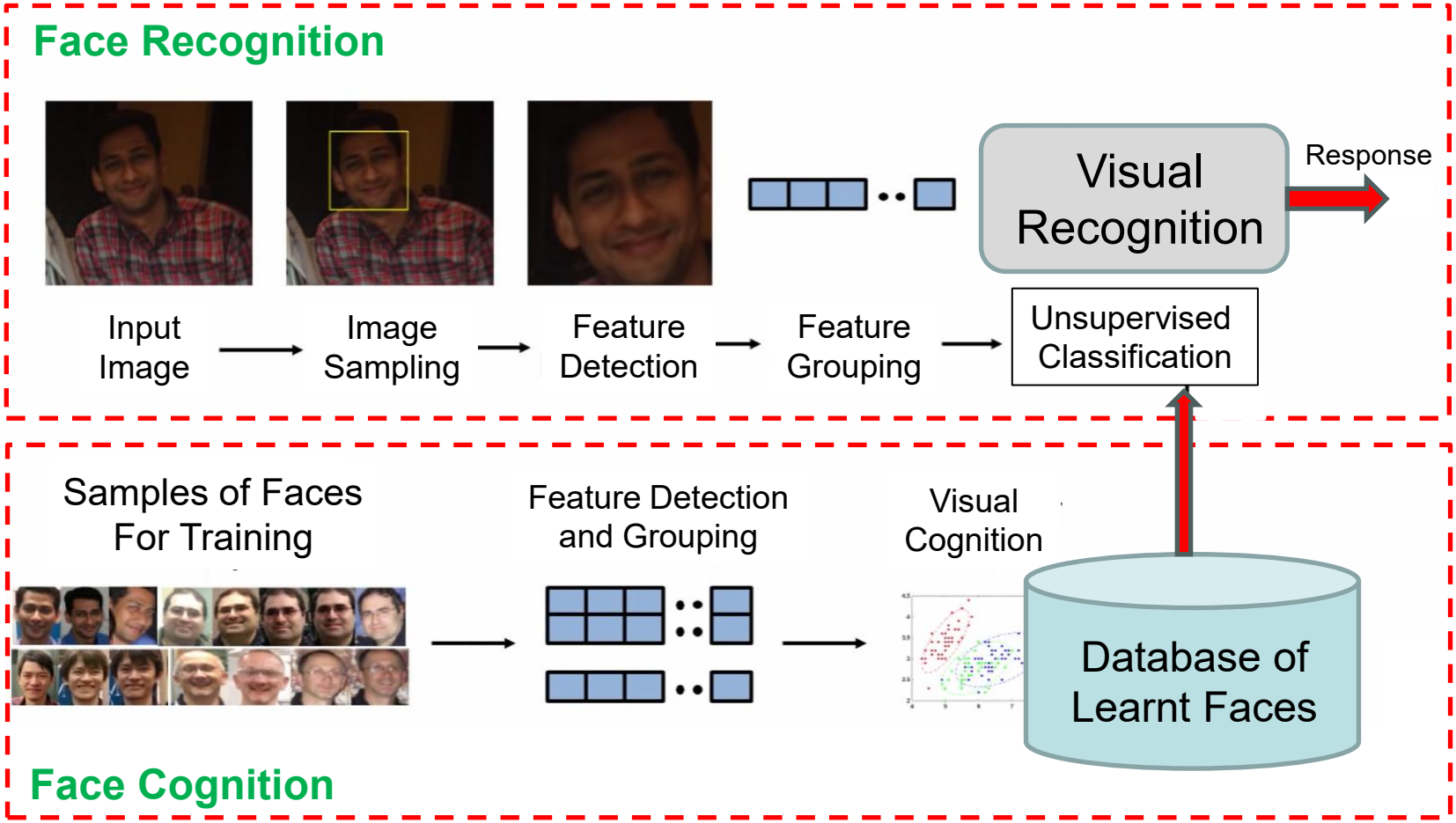
Teach Once and Use Forever

Re-Cognition

- Feature Detection and Grouping
- Knowledge Representation
- Symbol Grounding or Matching
- Output of Response

Long lasting unsolved challenge in AI

Comparison between Cognition of Face Images and Recognition of Face Images



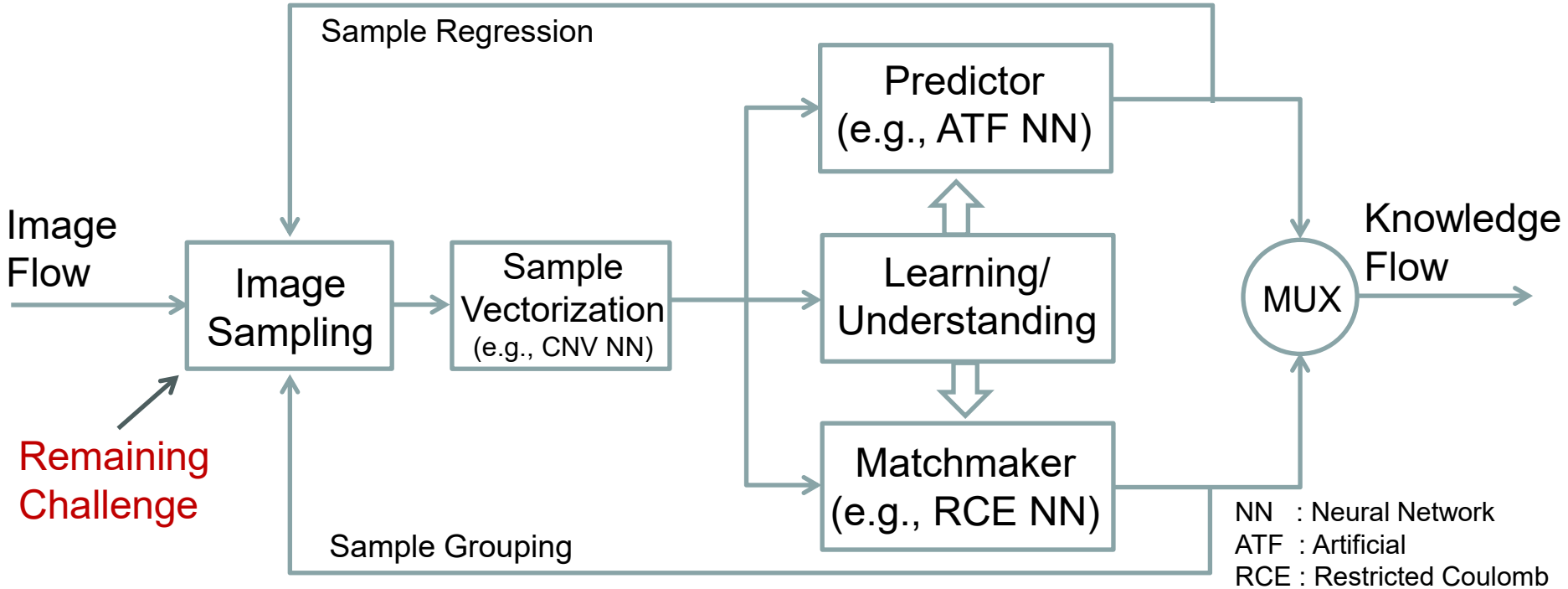
What are solutions underlying recognition of …

- Colors
- Curves
- Patterns
- Products
- etc?




Answer: to connect physical world to conceptual world (Feature Extraction → Symbol Grounding)

- Key Steps Involve Use of 2D-Model-Based Cognition and Recognition:
  - Pattern/Object Detection Followed by Categorization
  - Pattern/Object Detection Following by Identification

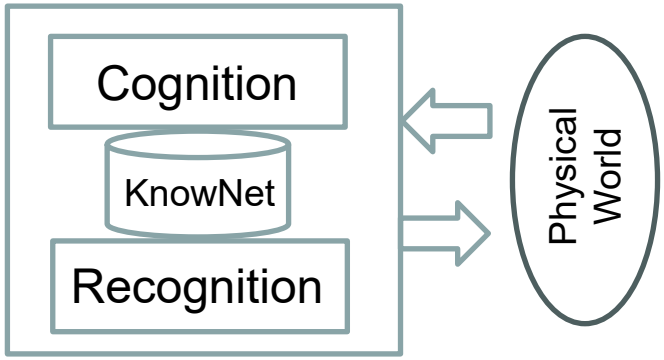


Note: We advocate KnowNet which is a knowledge network.

Outline of Lecture 4

- Principle of Recognition (Understanding)
- Principle of Artificial Neural Network
- Principle of RCE Neural Network
- Recogniton of Colors
- Recognition of Curves
- Recognition of Patterns
- Practices in MATLAB

Revision

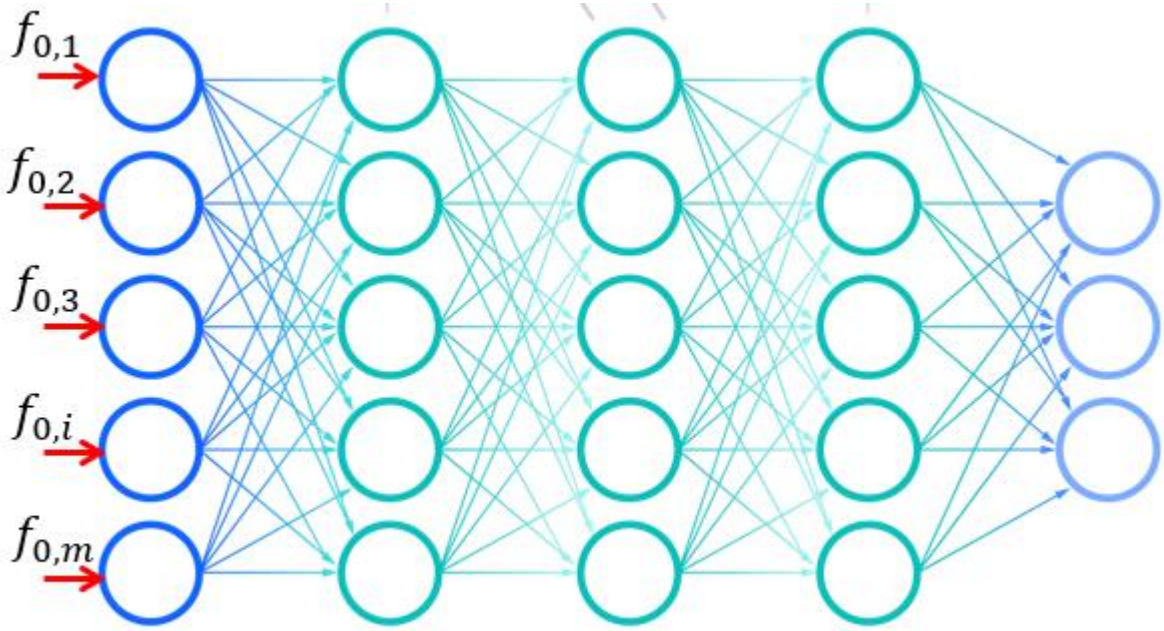


History of Artificial Neural Network (ANN)

- **Warren McCulloch and Walter Pitts (1943):** Often cited as the original inventors of the first computational model for neural networks. They published "A Logical Calculus of the Ideas Immanent in Nervous Activity," which proposed a simplified mathematical model of a biological neuron (the **McCulloch-Pitts neuron**) capable of performing logical functions.
- Frank Rosenblatt[1] (1958) created the perceptron, an algorithm for pattern recognition.
- The first deep learning multilayer perceptron trained by stochastic gradient descent was published in 1967 by Shun'ichi Amari.
- Backpropagation is an efficient application of the chain rule derived by Gottfried Wilhelm Leibniz in 1673.
- The origin of the Convolutional Neural Network (CNN) architecture is the "neocognitron" introduced by Kunihiko Fukushima in 1980.

Definition of Artificial Neural Network

- An artificial neural network is an associative memory of values for the representation of a system of linear equations.



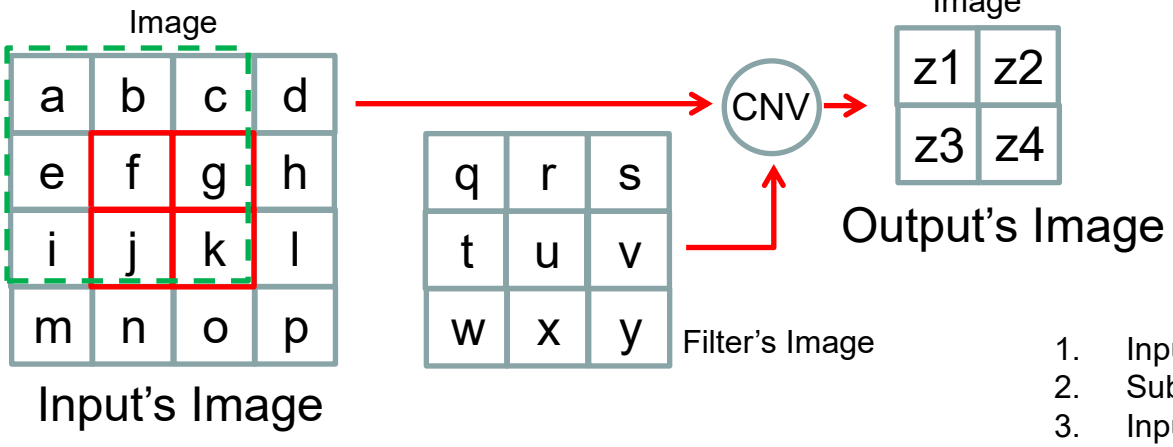
Property 1 of Artificial Neural Network

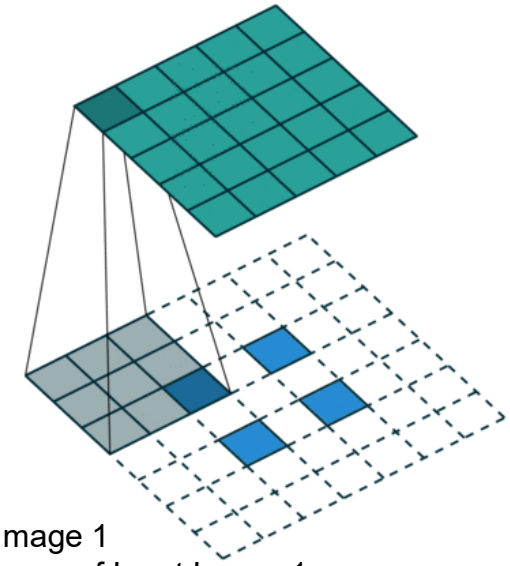
- An artificial neural network is a solution for the implementation of the convolution between an input and a filter.

- What is a filter?
  - A filter is a selector of feature or knowledge.
  - A filter is better to be the outcome of top-down design.
  - A filter could be the outcome of bottom-up optimization.

Example of Image Convolution

- Convolutional Neural Network (i.e. CNN)

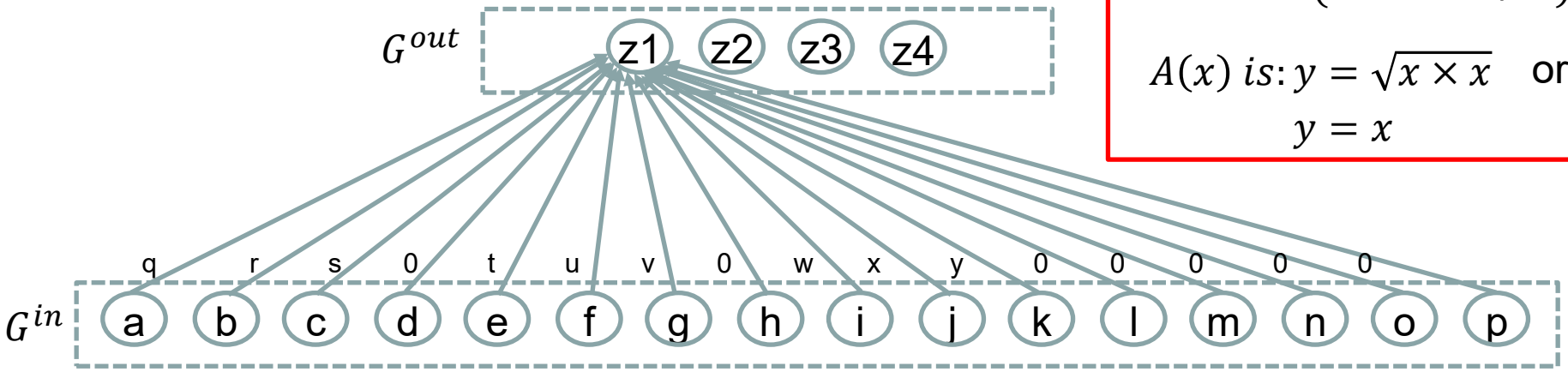




1. Input Image 1
2. Sub-image of Input Image 1
3. Input Image 2

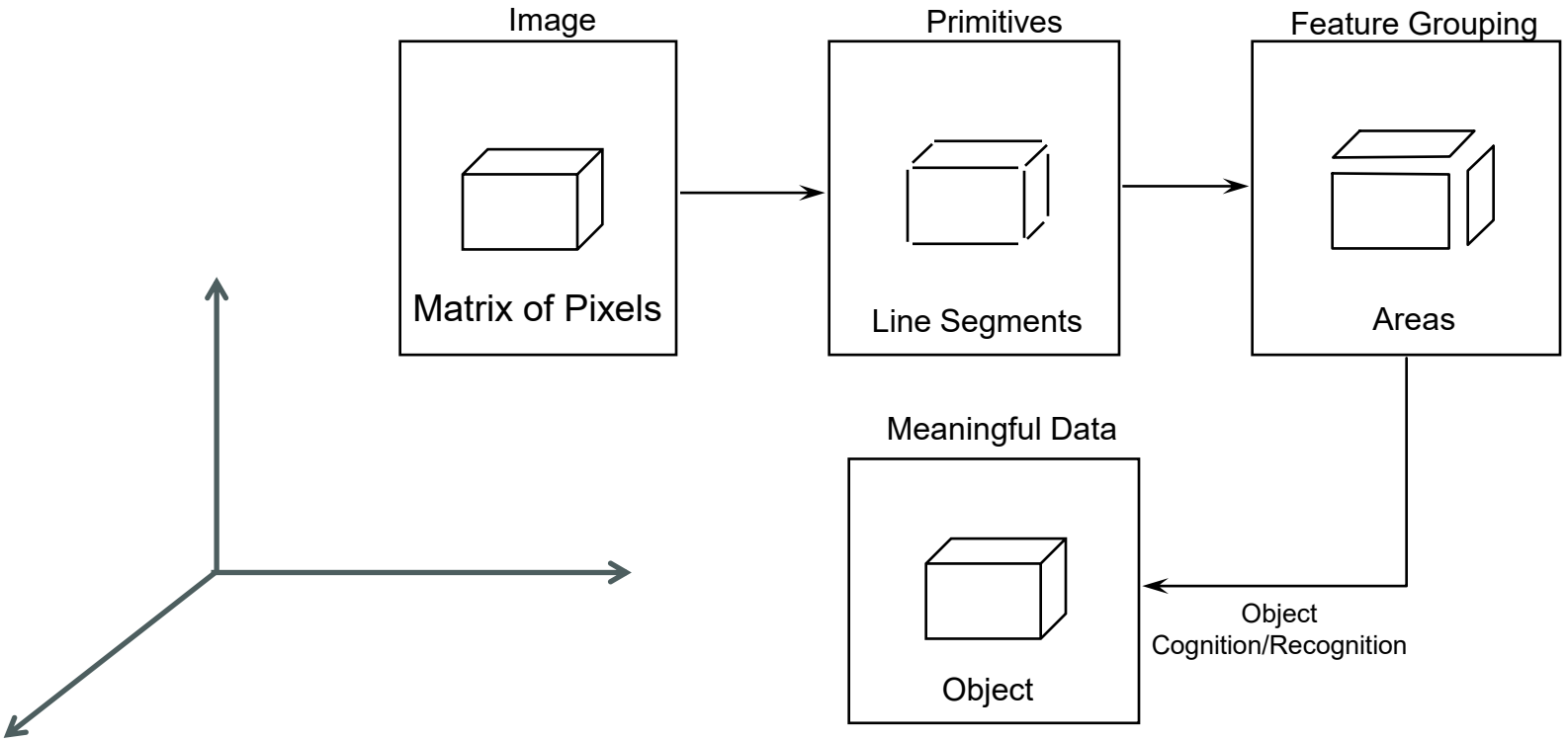
$$G^{out} = A(W \times G^{in} + B)$$

$$A(x)\ is: y = \sqrt{x \times x} \quad \text{or} \quad y = x$$



Property 2 of Artificial Neural Network

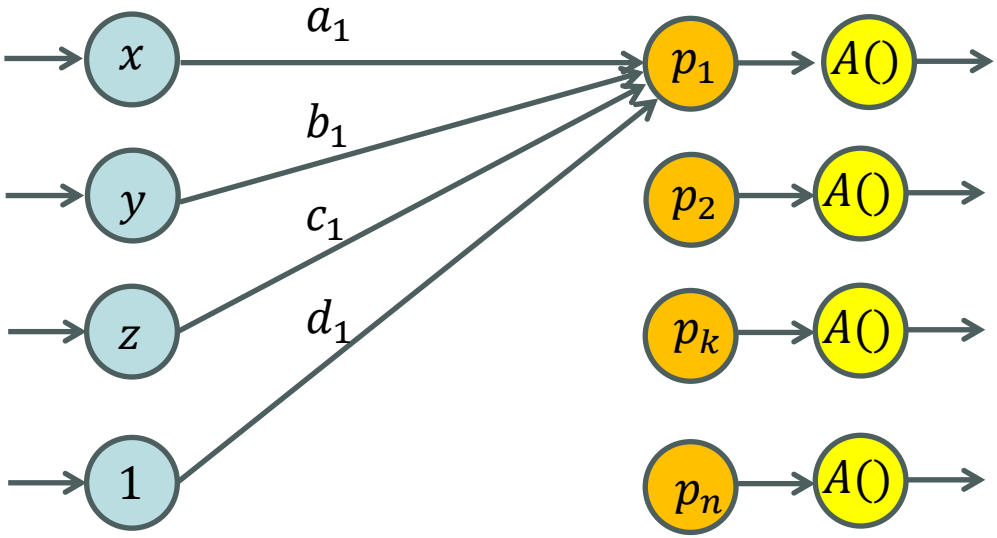
- An artificial neural network is a solution for the implementation of a system of equations which represent the boundaries inside a knowledge space.
- An artificial neural network is better to be the outcome of top-down design.
- An artificial neural network could be the outcome of bottom-up optimization.



Example of Boundary Representation

- Representational Neural Network (i.e. RNN)

$$\begin{cases} p_1 = a_1 x + b_1 y + c_1 z + d_1 \\ p_2 = a_2 x + b_2 y + c_2 z + d_2 \\ \cdots \\ p_n = a_n x + b_n y + c_n z + d_n \end{cases}$$



MIMO = Sum of MISO

A(): Activation Function

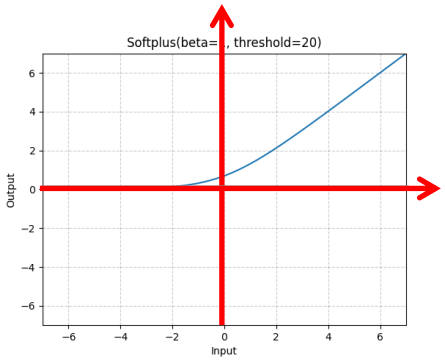
$output = A(input)$

Popular Activation Functions:

- SoftPlus Function:

$$\mathrm{X} = \{x_i, i = 1,2, \dots, n\}$$

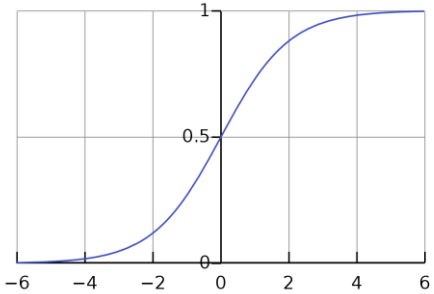
$$A(X) = \{\ln(1 + e^{x_i}), i = 1,2, \dots, n\}$$



- SoftMax Function:

$$\mathrm{X} = \{x_i, i = 1,2, \dots, n\}$$

$$A(X) = \{\frac{e^{x_i}}{\sum_{i=1}^{n} e^{x_i}}, i = 1,2, \dots n\}$$



- Sigmoid Function:

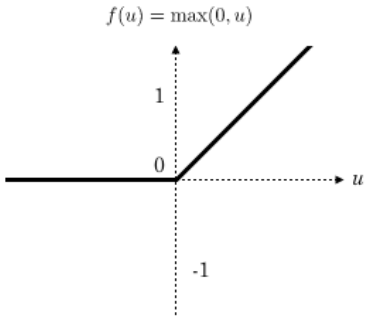
$$\mathrm{X} = \{x_i, i = 1,2, \dots, n\}$$

$$A(X) = \{\frac{1}{1 + e^{-x_i}}, i = 1,2, \dots n\}$$

- ReLU (Rectified Linear Unit):

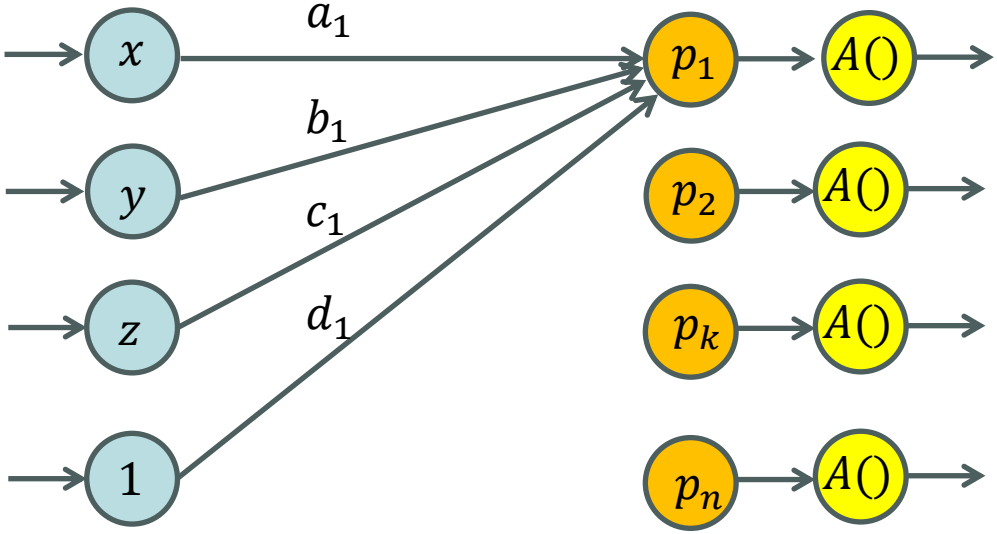
$$\mathrm{X} = \{x_i, i = 1,2, \dots, n\}$$

$$A(X) = \{\max(0, x_i), i = 1,2, \dots, n\}$$



Property 3 of Artificial Neural Network

- An artificial neural network is a predictor or classifier which could tell the identity and/or category of given physical knowledge.
- An artificial neural network is a solution to the symbol-grounding problem in AI.



MIMO = Sum of MISO

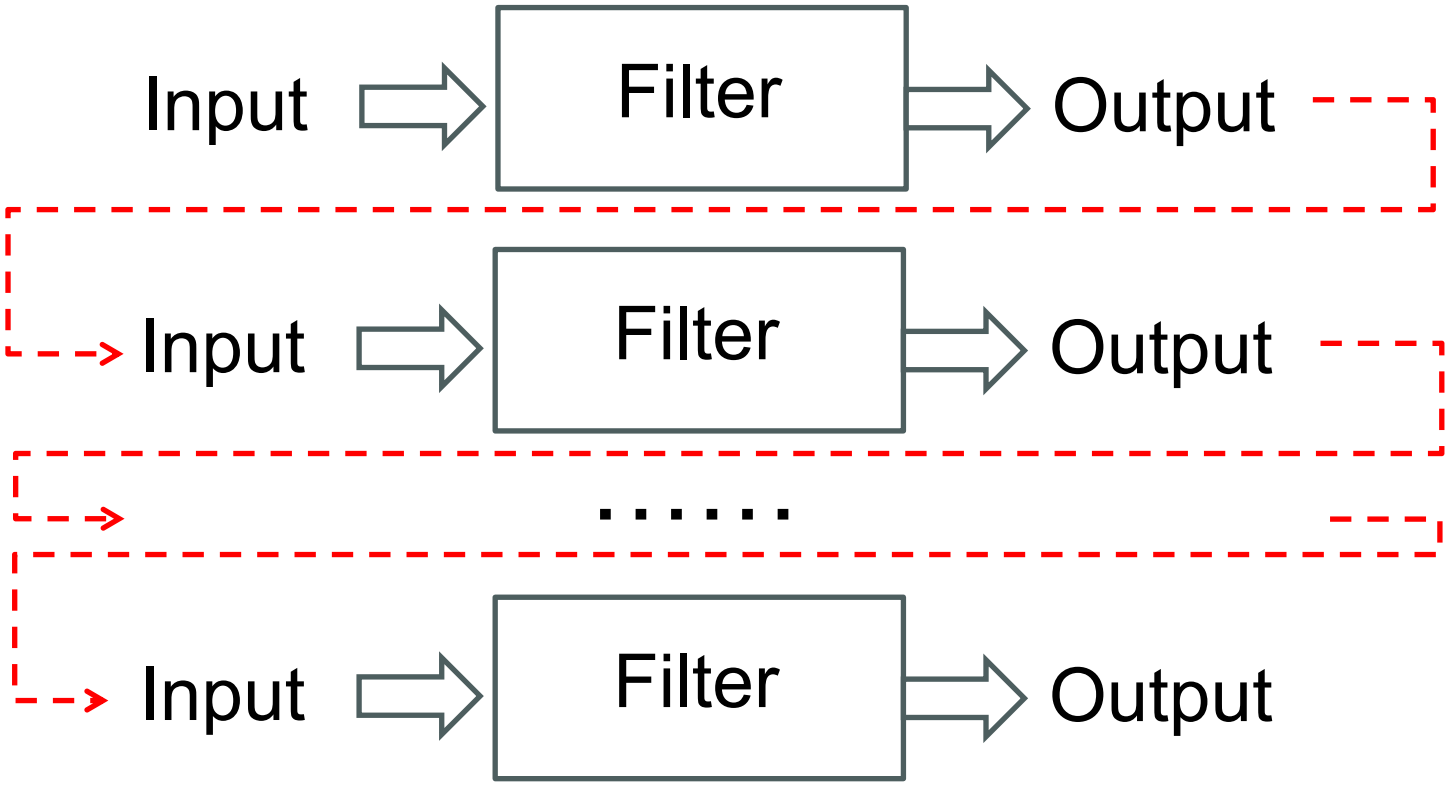
A(): Activation Function

$$A(input) = \begin{cases} -1, if\ input < 0 \\ 0, if\ input = 0 \\ +1, if\ input > 0 \end{cases}$$

{belong, not belong}

What is deep neural network?

- It is a cascaded artificial neural network which integrates artificial neural network's computational power and prediction capability together.



Details of Deep Neural Network

(output = symbol with belief) $A(X)$: $SoftPlus, SoftMax, Sigmoid, etc$

Feature Vector at Layer 0:

Input = $F_0 = \{f_{0,i}, i = 1,2, \dots, m\}$

Feature Vector at Layer 1:

$F_1 = A(\mathrm{W}_0 \times F_0 + \mathrm{B}_0)$

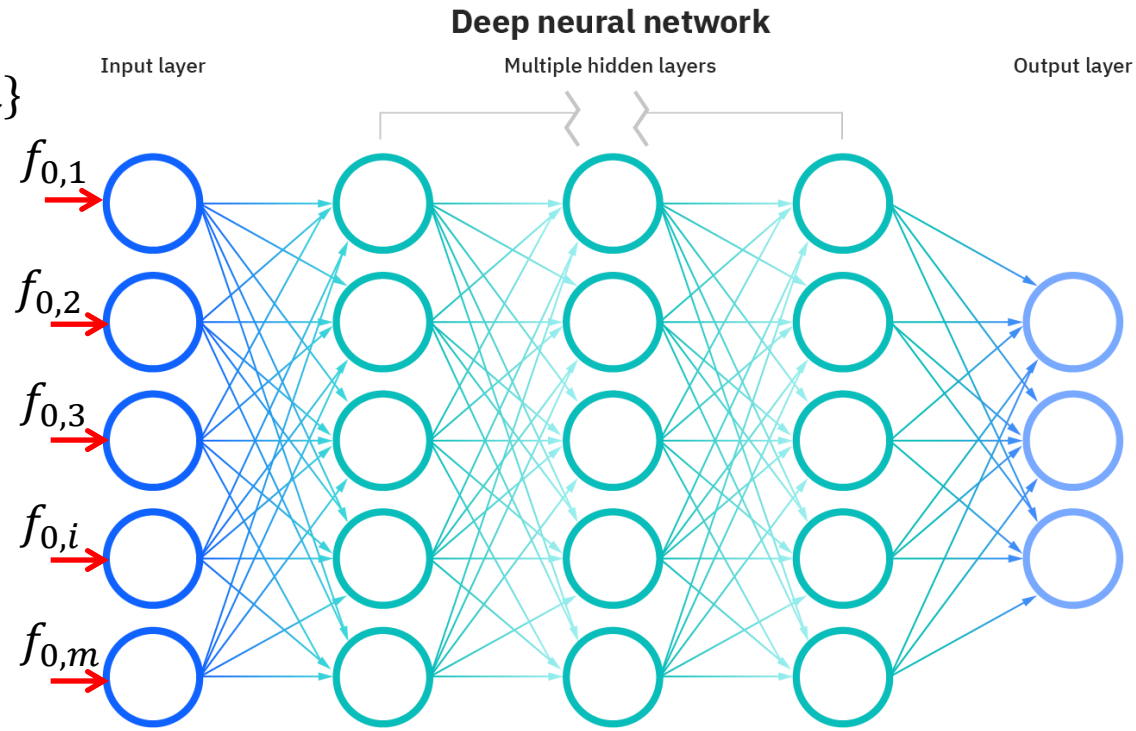
Feature Vector at Layer k:

$F_k = A(\mathrm{W}_{k-1} \times F_{k-1} + \mathrm{B}_{k-1})$

.....

Feature Vector at Layer n:

$F_n = A(\mathrm{W}_{n-1} \times F_{n-1} + \mathrm{B}_{n-1})$



Conceptual Meanings at Output:

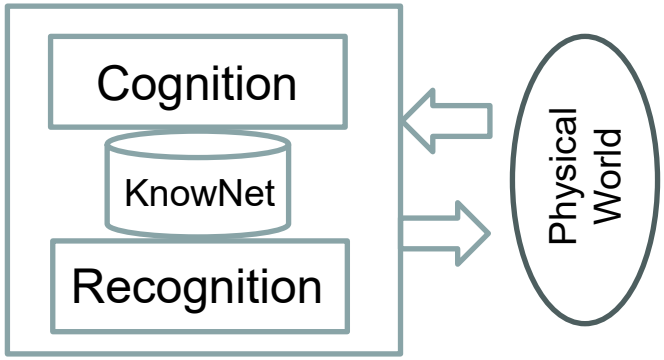
$Belief = \{[belief_j, possibility_j(F_n)], j = 1,2, \dots, L\}$

MIMO

Outline of Lecture 4

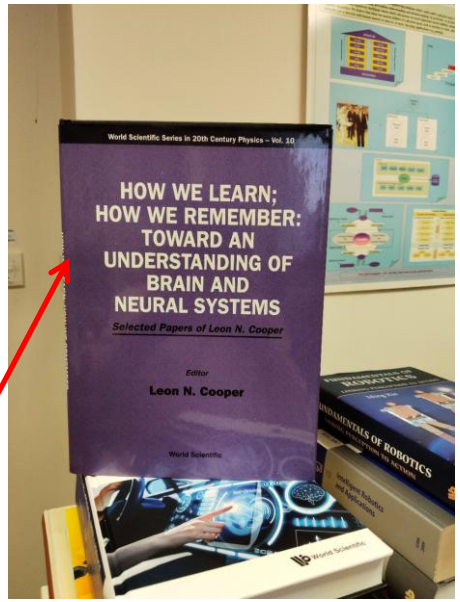
- Principle of Recognition (Understanding)
- Principle of Artificial Neural Network
- Principle of RCE Neural Network
- Recogniton of Colors
- Recognition of Curves
- Recognition of Patterns
- Practices in MATLAB

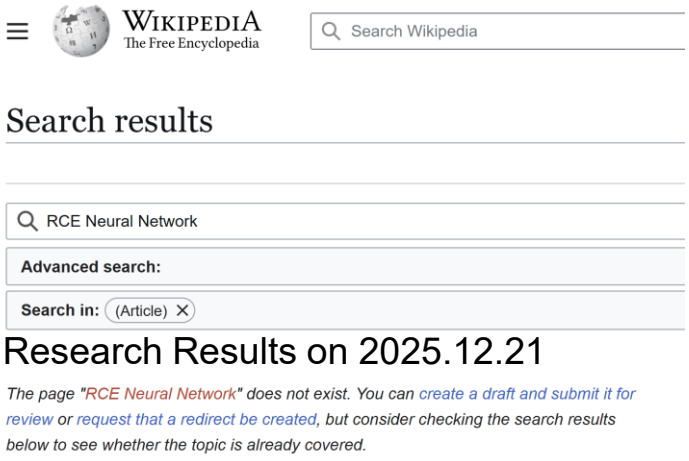
Revision



History of RCE Neural Network (RNN)

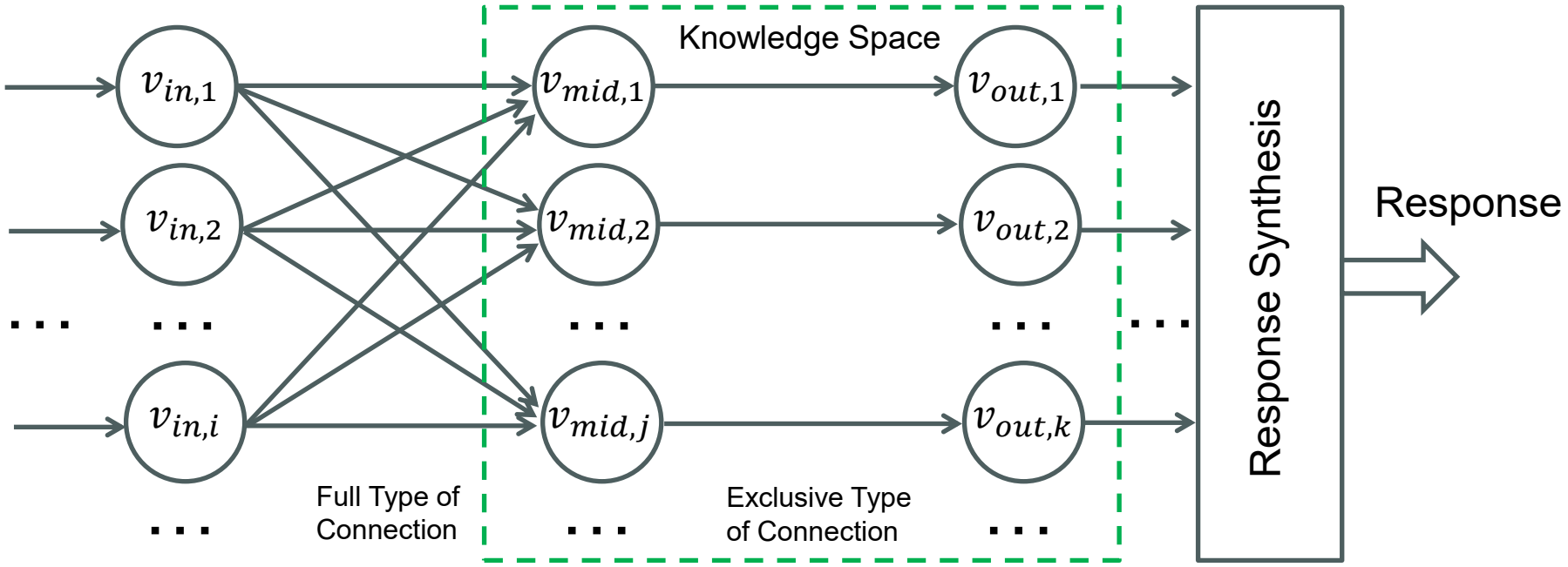
- **Leon N. Cooper** (né **Kupchik**; February 28, 1930 – October 23, 2024) was an American theoretical physicist and neuroscientist. He won the Nobel Prize in Physics for his work on superconductivity. Cooper developed the concept of Cooper pairs and collaborated with John Bardeen and John Robert Schrieffer to develop the BCS theory of conventional superconductivity. In neuroscience, Cooper co-developed the BCM theory of synaptic plasticity.
- Restricted Coulomb RCE Neural Network was developed by a team led by Leon N. Cooper in the late 1970s.
- Some modified versions of RCE Neural Network was developed by T. Olmez et al (1993) and M. Xie et al (2003).





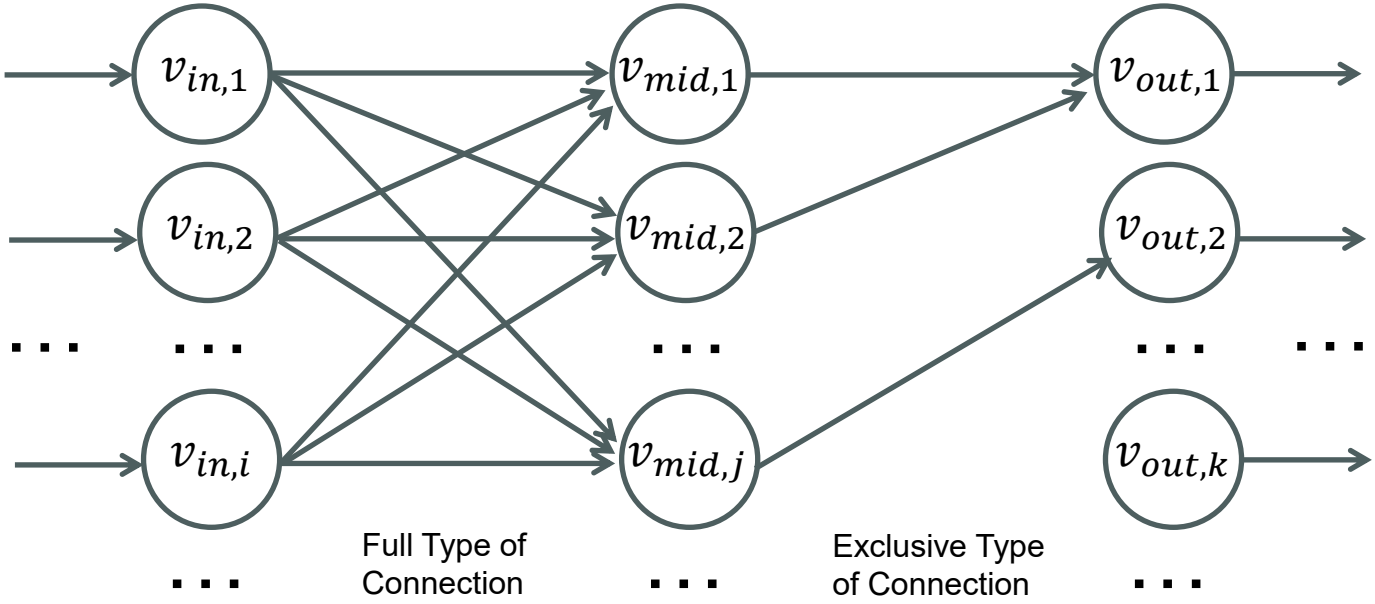
Definition of RCE Neural Network

- A RCE neural network is an associative memory of vectors for the representation of a knowledge space.



Property 1 of RCE Neural Network

- RCE neural network is a vectorized neural network which has three layers.
- Each layer consists of a list of knowledge vectors. The weighting coefficient at each link is 1.
- Between the input layer and middle layer (also called prototype layer), the connection among the node is full-type of connection.
- Between the middle layer and output layer, the connection among the nodes is exclusive type of connection.



Property 2 of RCE Neural Network

- RCE neural network is a MIMO system which could be represented as:
  - Sum of MISO
  - Sum of SIMO
  - Sum of SISO
- Hence, RCE neural network represents a knowledge space which supports incremental learning.

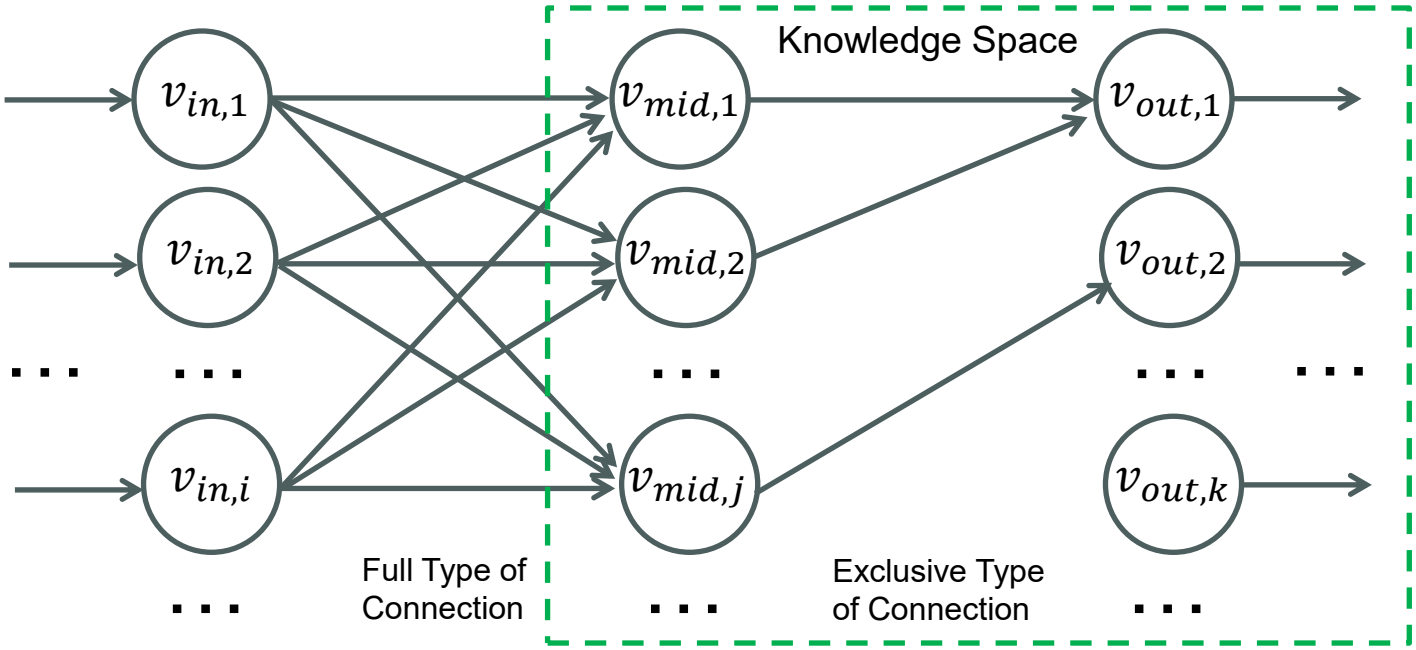
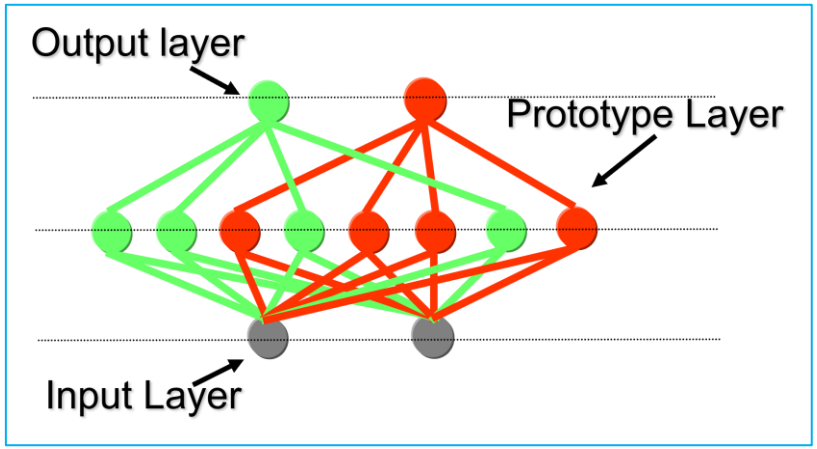


Illustration of Property 2:

- MIMO = Sum of SISO
- RCE NN is an ideal system!

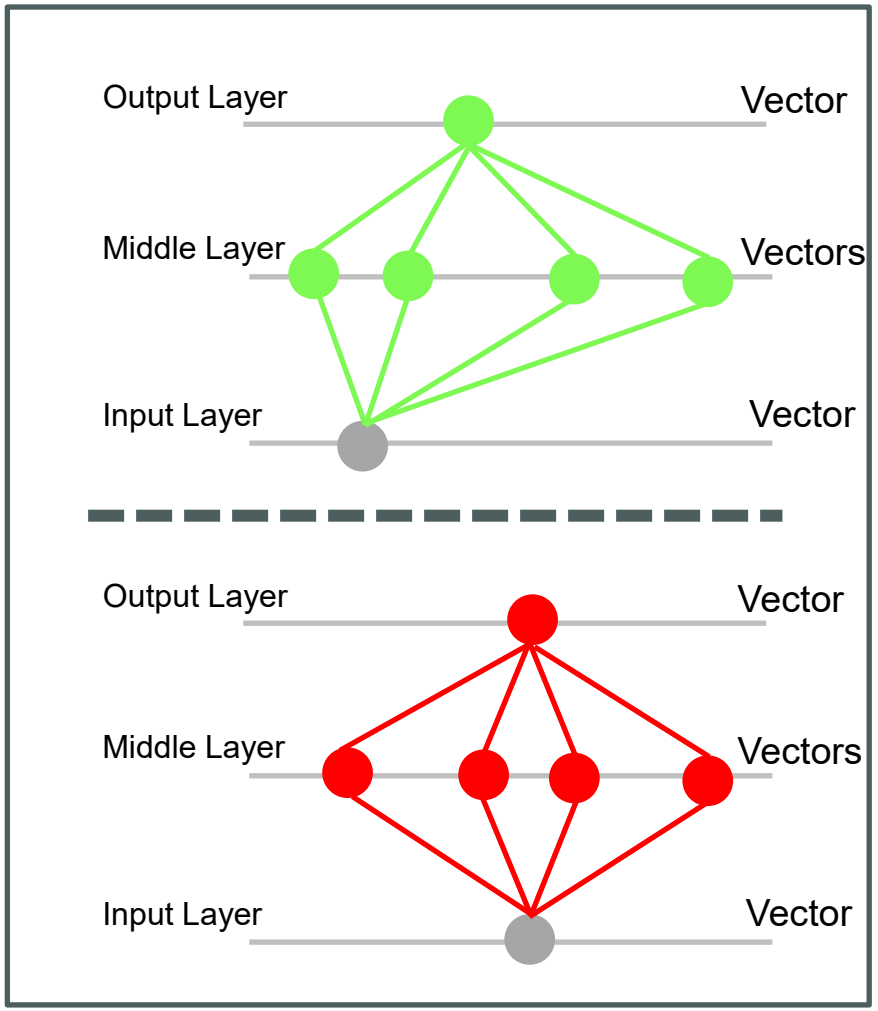


MIMO = Sum of MISO

MIMO = Sum of SISO

=

Incremental Learning



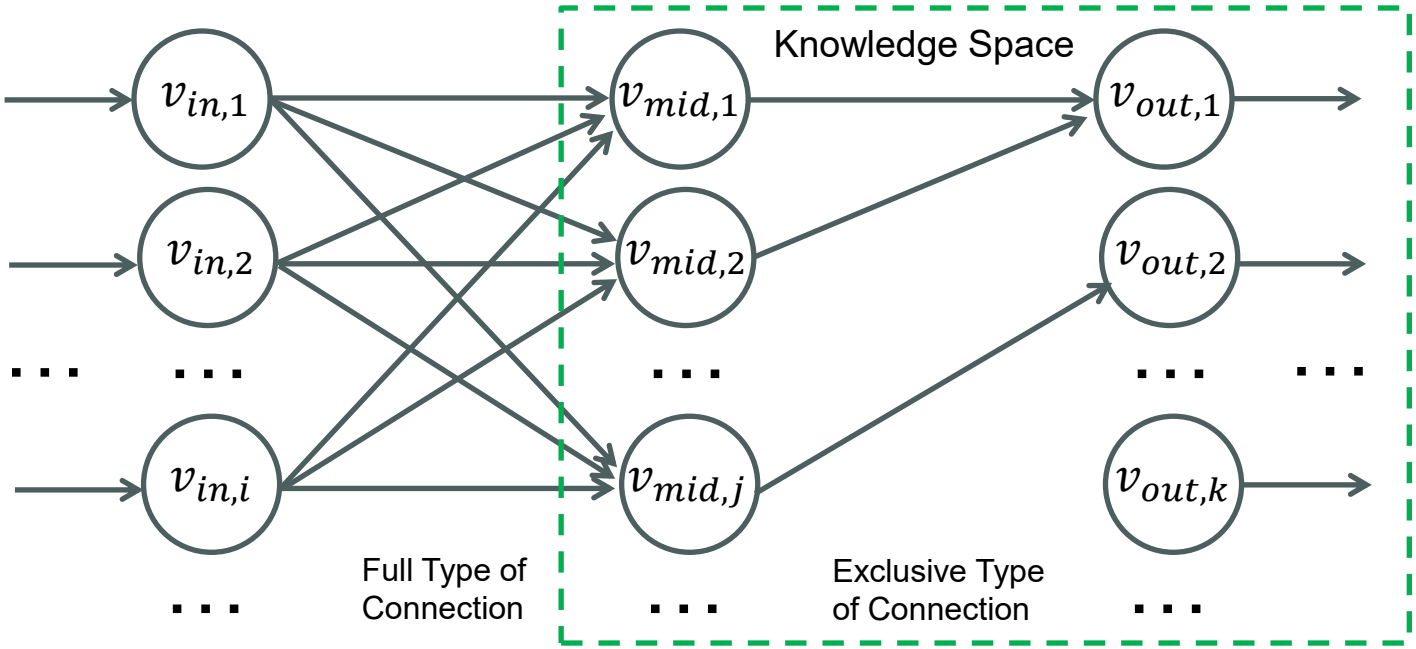
Property 3 of RCE Neural Network

- Each vector at the input layer is a feature vector which could be represented as:

$$v_{in,i} = \{v_{in,i,l}, l = 1,2, \ldots\}$$

Could be incrementally increased

- Output from each node of input layer is its feature vector.



Property 4 of RCE Neural Network

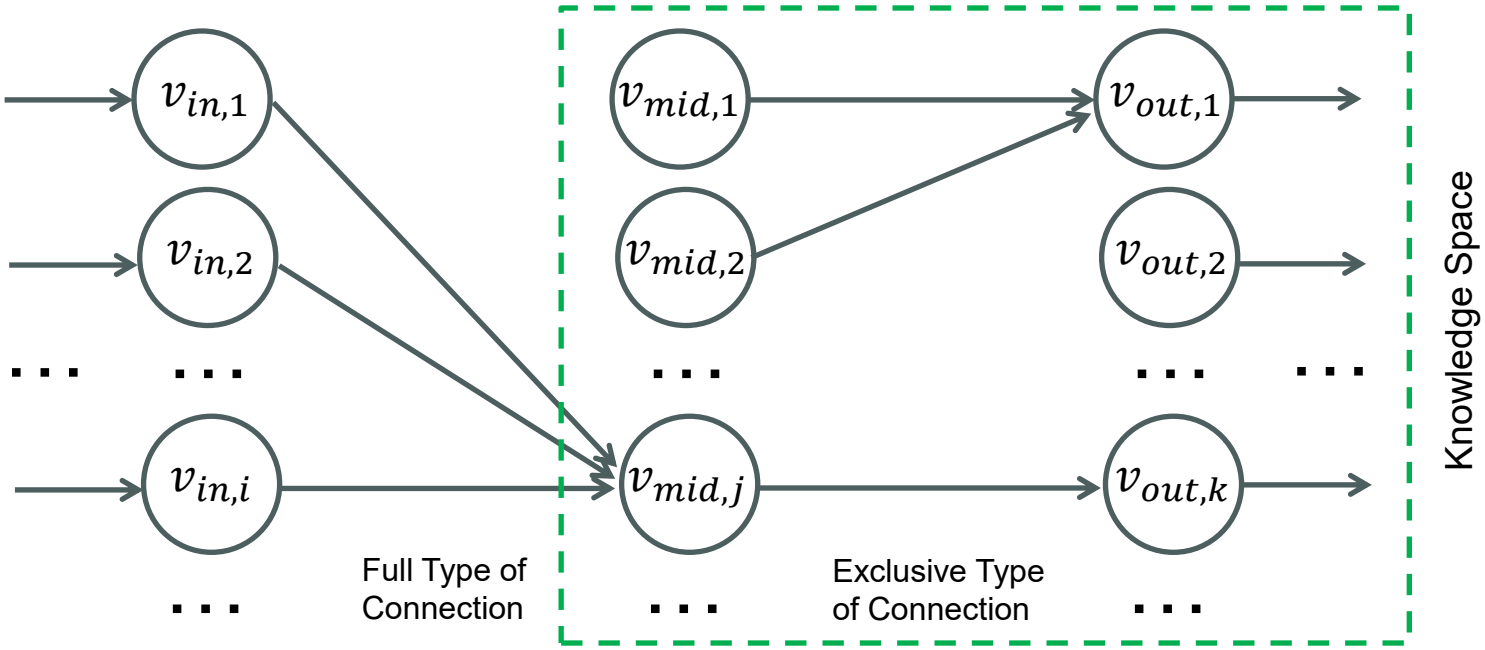
- Each vector at the middle layer (also called prototype layer) is knowledge vector which could be represented as:

$$v_{mid,j} = \{v_{mid,j,l}, l = 1,2, \ldots\} = \{\mu_j, \Sigma_j, p_j, conceptual\ labels, others\}$$

μ_j: mean of training inputs
Σ_j: co-variance of training inputs

$j = 1,2, \ldots$

From Teachers/Trainers



Property 5 of RCE Neural Network

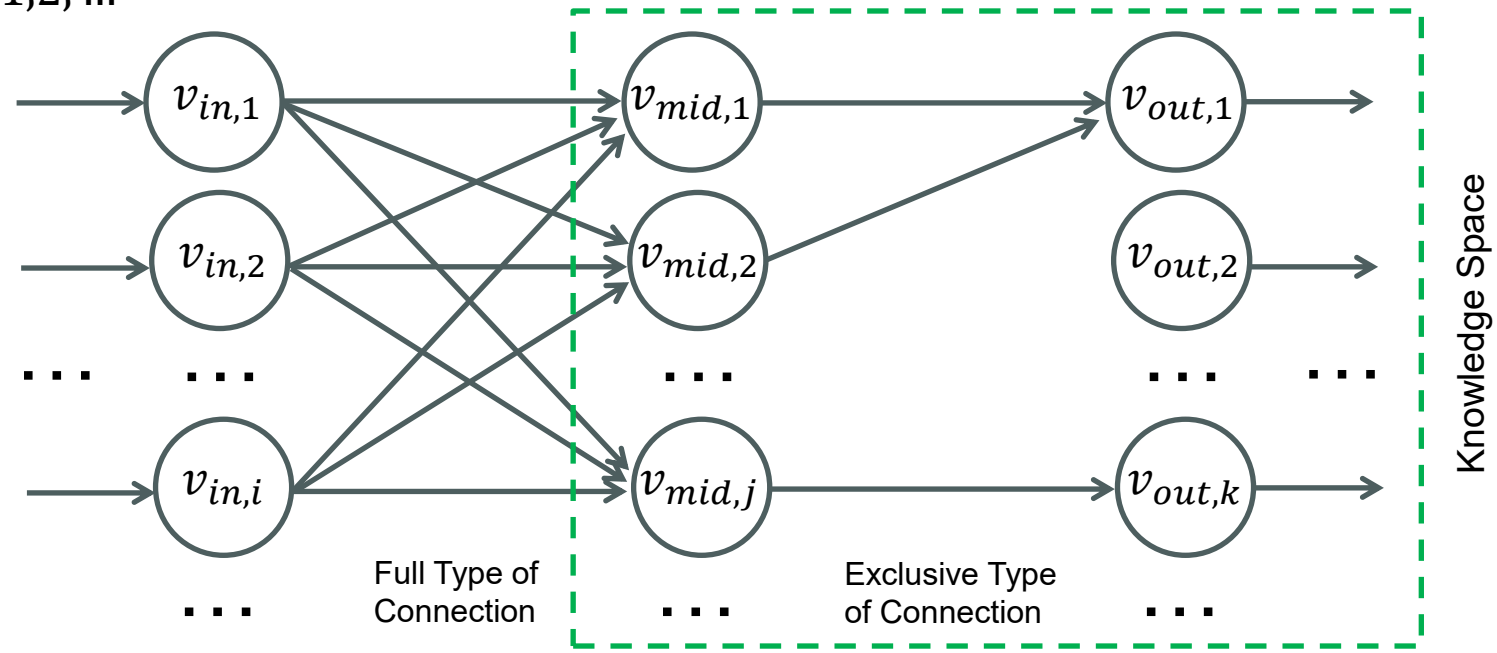
- Output from each node of middle layer is a possibility value:

$$v_{mid,j} = \{v_{mid,j,l}, l = 1,2, \ldots\} = \{\mu_j, \Sigma_j, p_j, conceptual\ labels, others\}$$

μ_j: means of training inputs
Σ_j: co-variance of training inputs

$$p_j = e^{-\frac{1}{2}\left(v_{in,i}-\mu_j\right)^t \Sigma_j^{-1}\left(v_{in,i}-\mu_j\right)}$$

$j = 1,2, \ldots$



Property 6 of RCE Neural Network

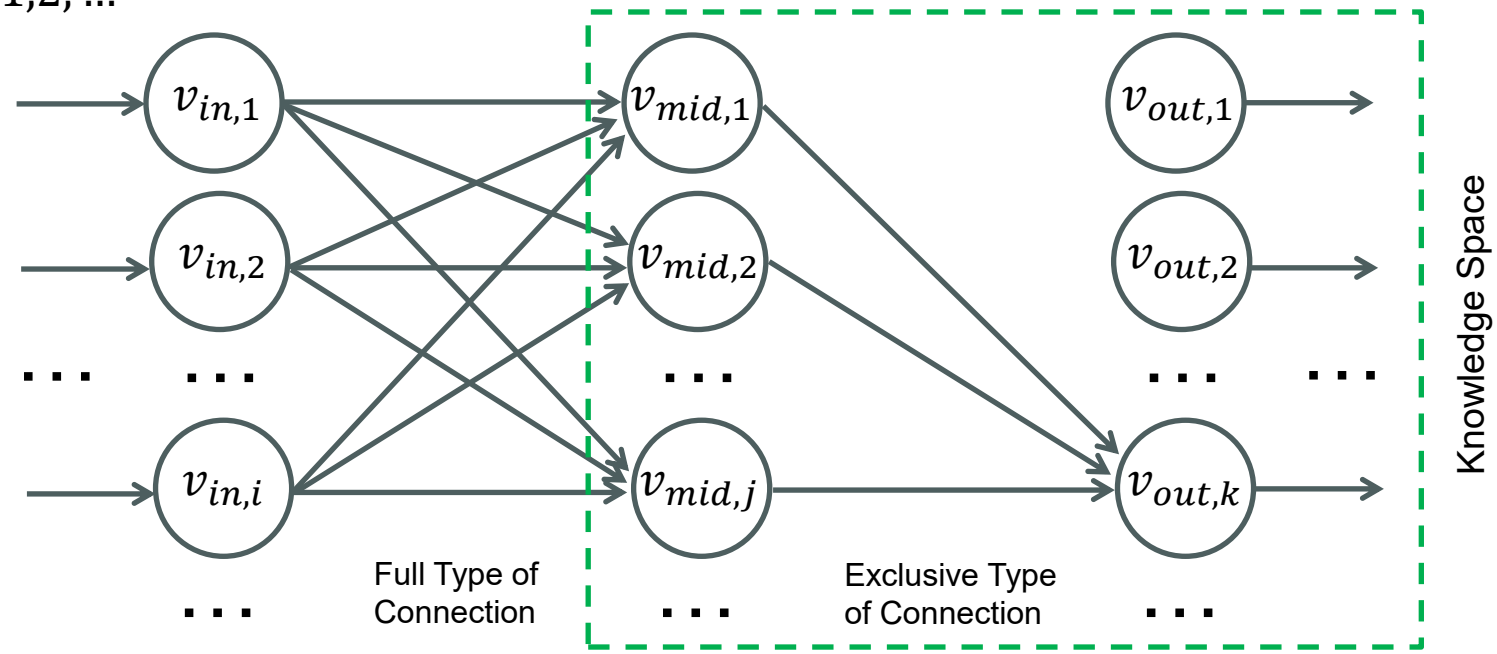
- Output from each node of output layer is the copy of the selected node from the middle layer:

$$v_{mid,j} = \{v_{mid,j,l}, l = 1,2, \ldots\} = \{\mu_j, \Sigma_j, p_j, conceptual\ labels, others\}$$

$$p_j = e^{-\frac{1}{2}\left(v_{in,i}-\mu_j\right)^t \Sigma_j^{-1}\left(v_{in,i}-\mu_j\right)}$$

$$\forall j, v_{out,k} = v_{mid,j}, if\ p_j = max$$

$j = 1,2, \ldots$



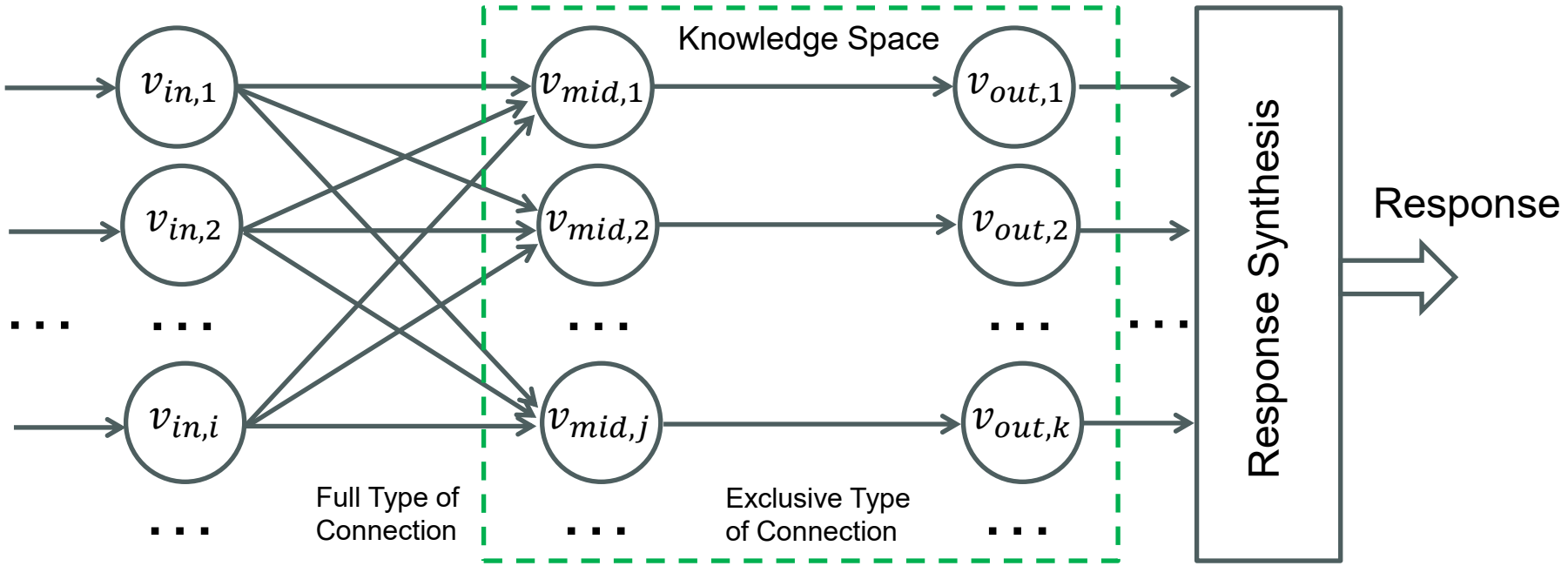
Property 7 of RCE Neural Network

- RCE neural network is a representation of knowledge space which is incrementally enriched by cognition or learning.
- RCE neural network is also an engine for knowledge recognition, which behaves like a sum of SISO systems.

$$\forall k, R_{reply} = S(v_{out,k}), if\ p_k = max$$

Use of Programming Language

$S(\quad)$: Any function of synthesis

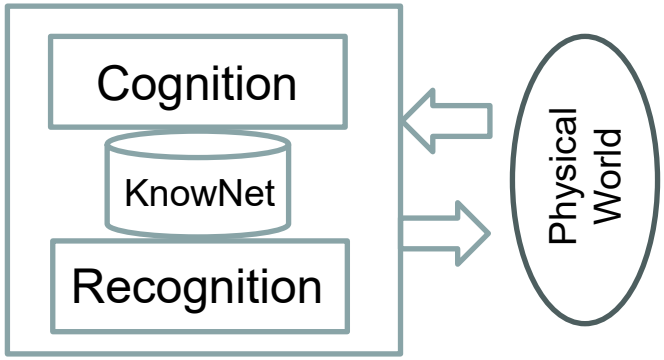


Outline of Lecture 4

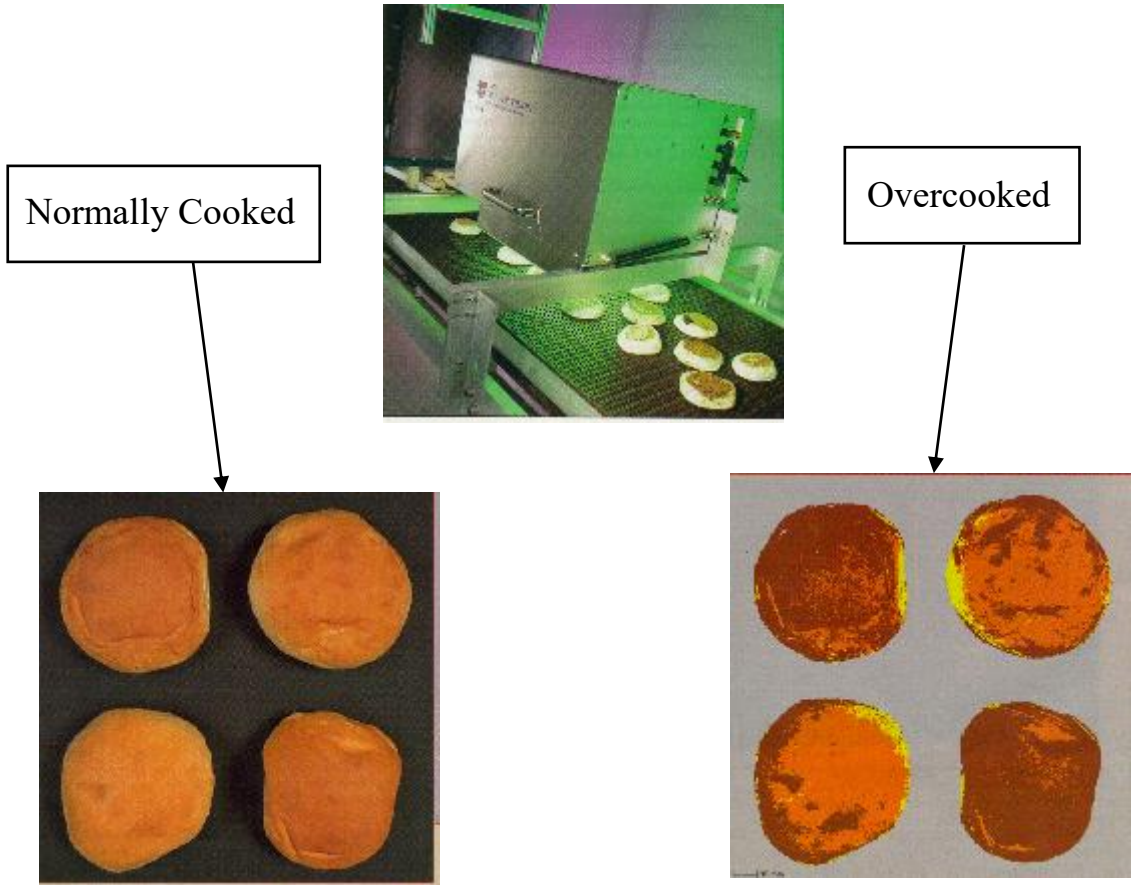
- Principle of Recognition (Understanding)
- Principle of Artificial Neural Network
- Principle of RCE Neural Network

Revision

- Recogniton of Colors
- Recognition of Curves
- Recognition of Patterns
- Practices in MATLAB

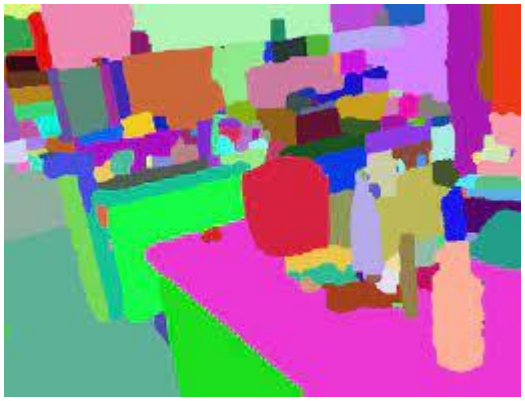


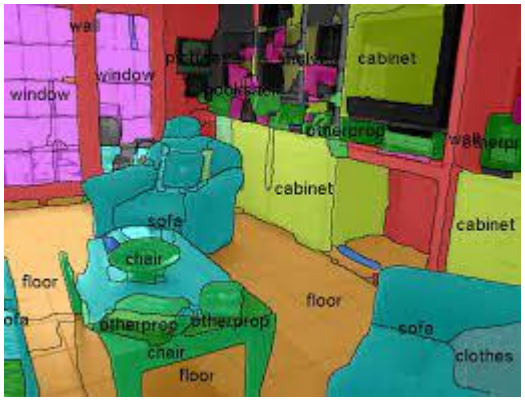
Example of Doing Recognition of Colors in Quality Control …

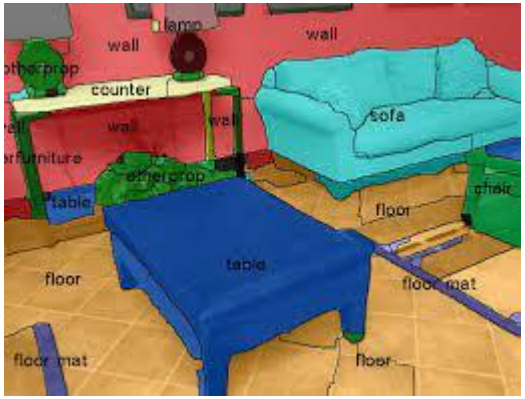


Quality Control of Food

Expected Outcomes from Recognition ...







Solution Toward Cognition and Recognition of Colors

Input 1: Any Image at Input

Images as 2D Matrices

- Image with Colors Inside

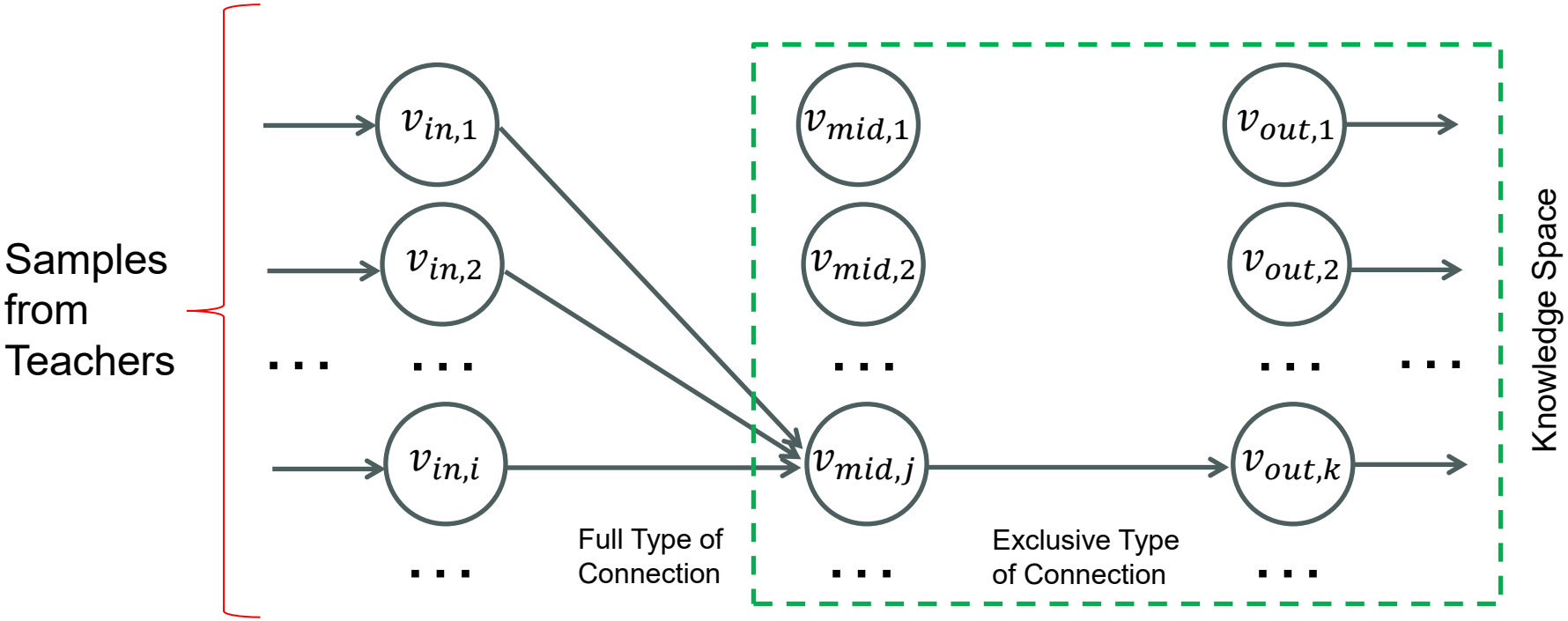
Data Structure of Pixel

```
typedef   struct
{
        short     r, g, b, I ;
        short     L, a, b ;
        short     u, v ;
        double    x, y, z ;
}
Pixel;
```

Input 2: Trained RCE Neural Network

- RCE neural network which has been trained by color cognition

$$v_{mid,j} = \{\mu_a, \mu_b, \sigma_a, \sigma_b, p_j, color\_name, others\}$$



Output

Images as 2D Matrices

- Image with Recognized Colors

Data Structure of Pixel

```
typedef   struct
{
        short     r, g, b, I ;
        short     L, a, b ;
        short     u, v ;
        double    x, y, z ;
        string    color_name ;
        double    feature_vector[ ];
}
Pixel;
```

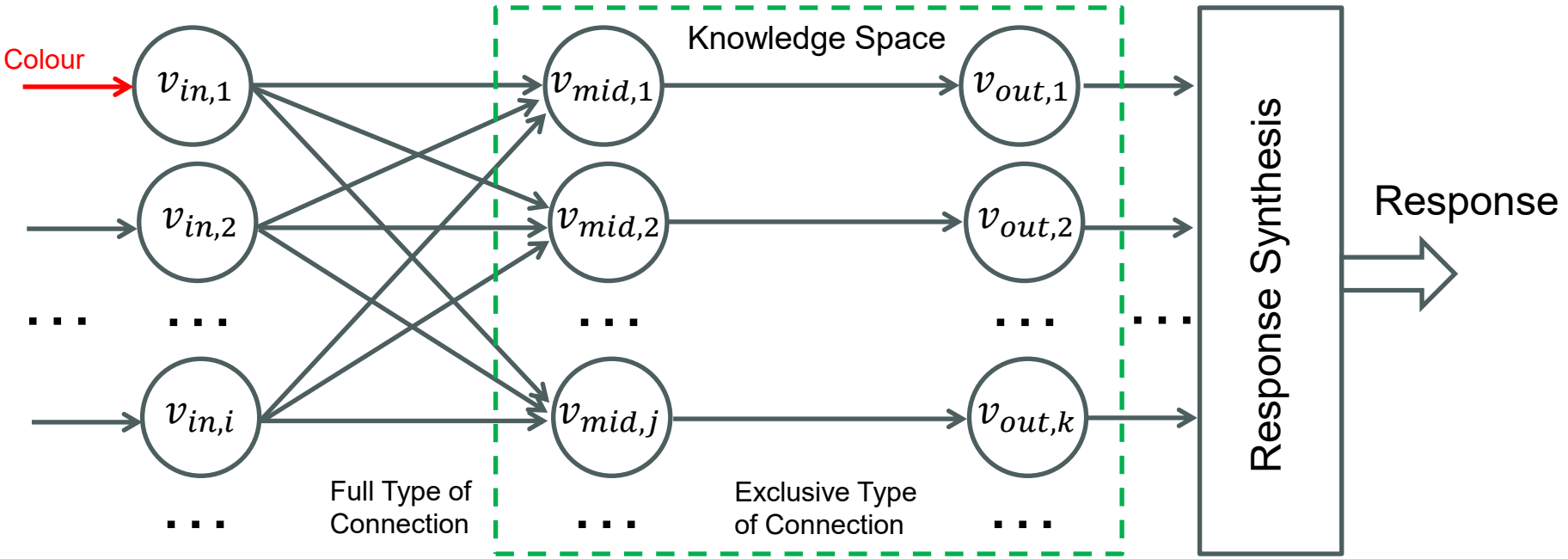
Question

- How to recognize or understand the color that a pixel inside an image belongs to?

$$\forall k, R_{reply} = S(v_{out,k}), if\ p_k = max$$

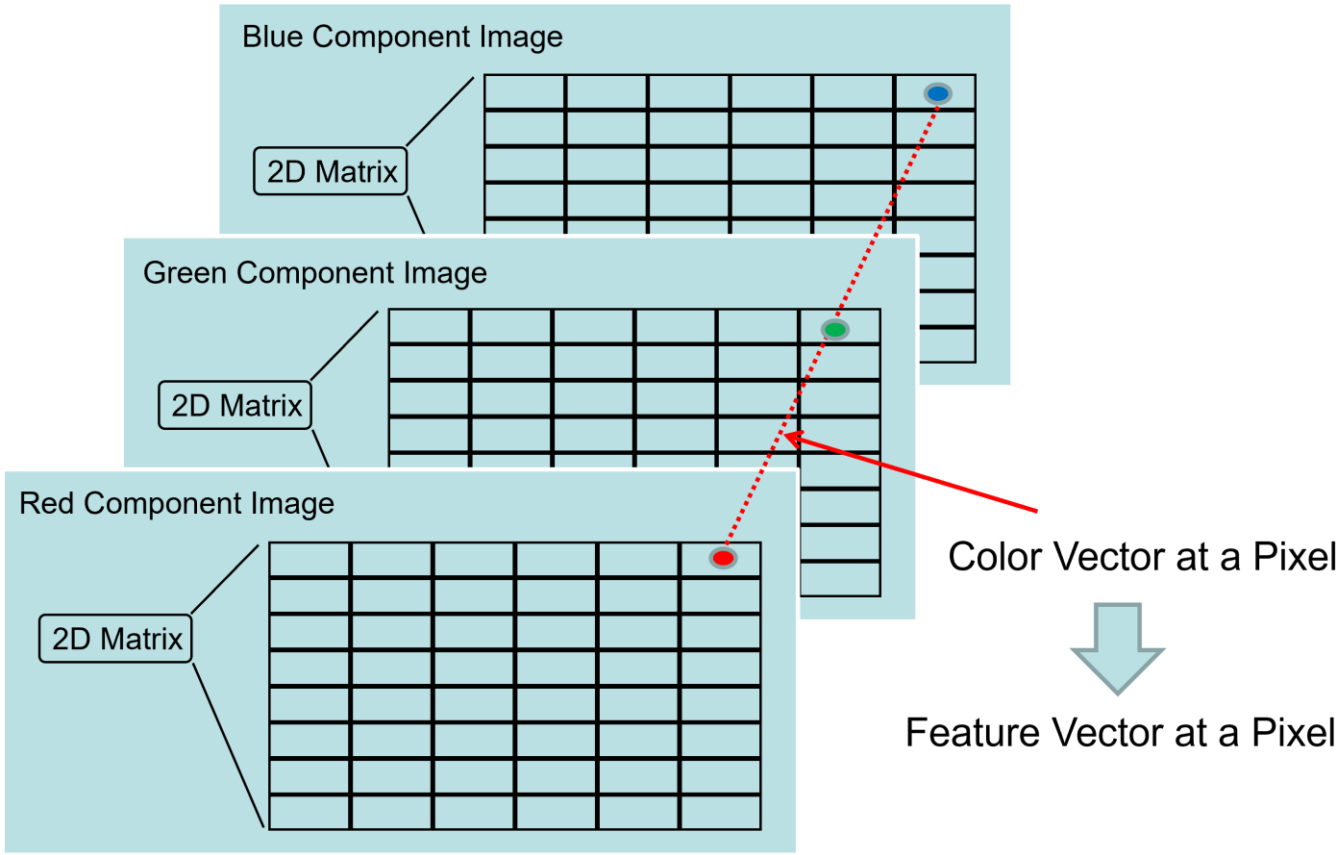
Use of Programming Language

$S(\quad)$: Any function of synthesis



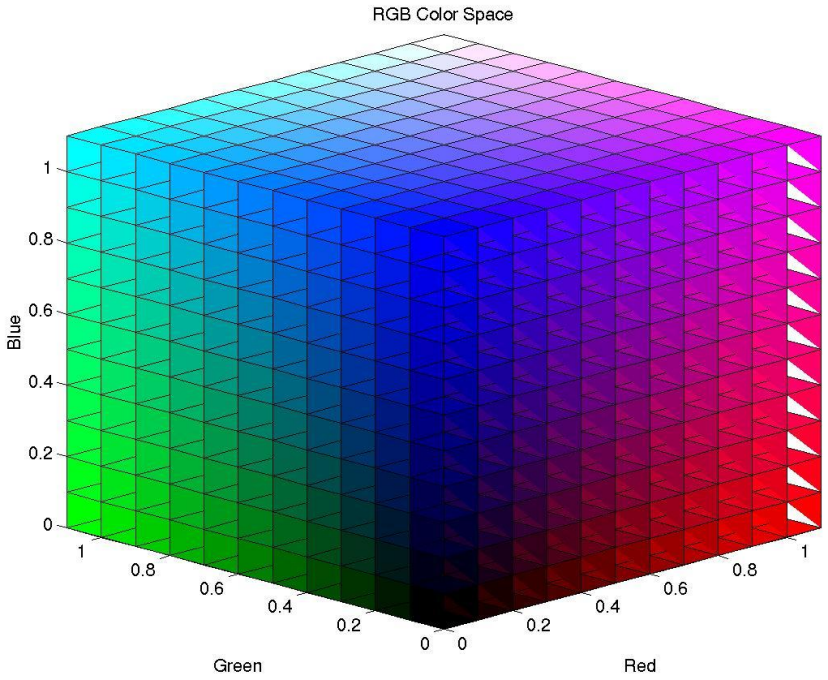
Step 1: To obtain visual samples (case of colors) from input image …

- Scan input image row by row and column by column.

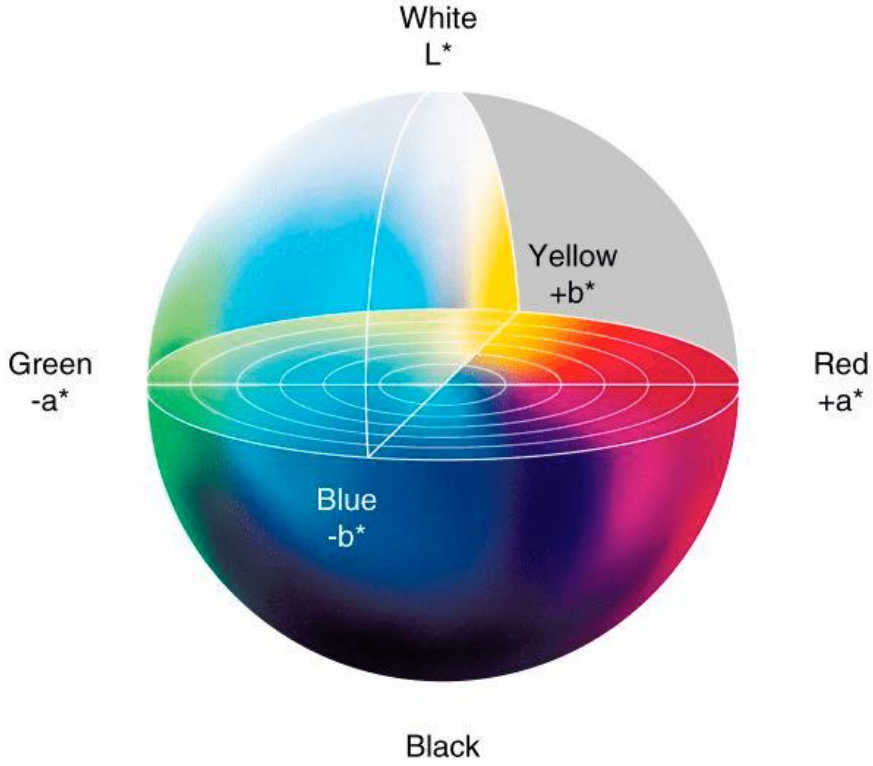


Step 2: To represent a color by a vector

- At each location of input image, represent the pixel by a vector in a color space such as:



RGB Color Space



L*a*b Color Space

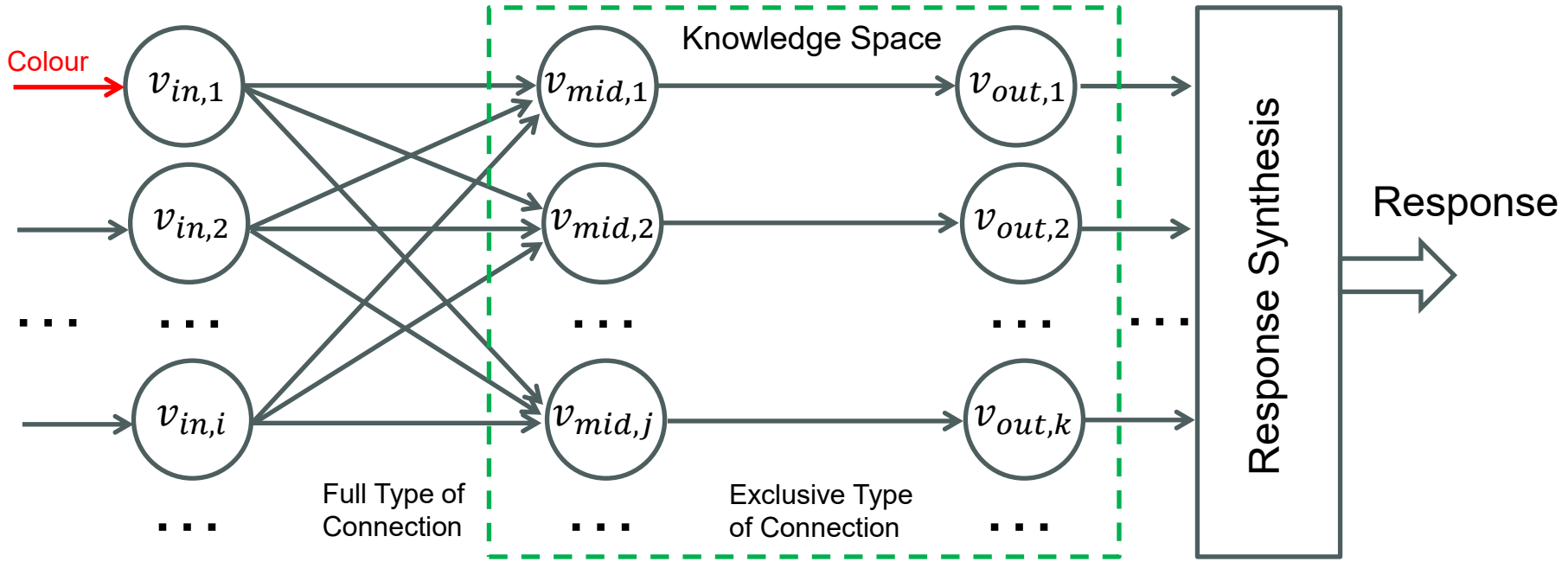
Step 3b: To determine output

- First, compute the possibility value at each node inside the middle layer of the trained RCE neural network.
- Subsequently, determine the output of color name which corresponds to the maximum value among the above-computed possibilities.

$$\forall k, R_{reply} = S(v_{out,k}), if\ p_k = max$$

Use of Programming Language

$S(\quad)$: Any function of synthesis



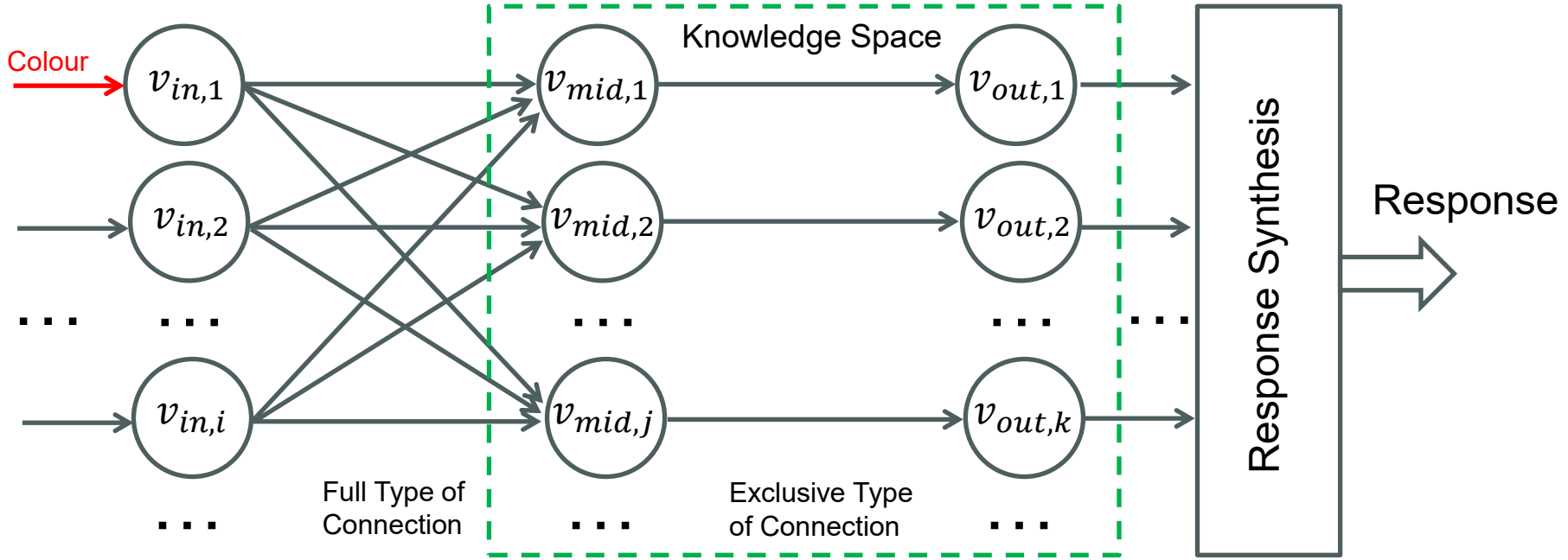
Step 3b (continued): To determine output

- Finally, synthesize the response using a natural language.

Use of Programming Language

$$\forall k, R_{reply} = S(v_{out,k}), if\ p_k = max$$

$S(\quad)$: Any function of synthesis



Simplified Example …

- Scan an input image row by row and column by column.
- At each location, compute the color vector: (L, a, b).
- Then, compute the possibility value at each node inside the middle layer of the trained RCE neural network:

$$p_k(a,b) = e^{-\frac{1}{2}(\frac{a-\mu_a}{\sigma_a})^2} \times e^{-\frac{1}{2}(\frac{b-\mu_b}{\sigma_b})^2}, k = 1,2,\ldots$$

- Subsequently, determine the color name with the maximum value among the above-computed possibilities:

$$P_{max} = \max\{p_k(a,b), k = 1,2,\ldots\}, \text{ if } P_{max} > P_{min}$$

Lower Limit of P_{max}

- Finally, synthesize the response using a natural language.

Exercise

- We pinpoint a location on an image. Its RGB value is (40, 40, 200). What is the possibility for it to belong to blue color?

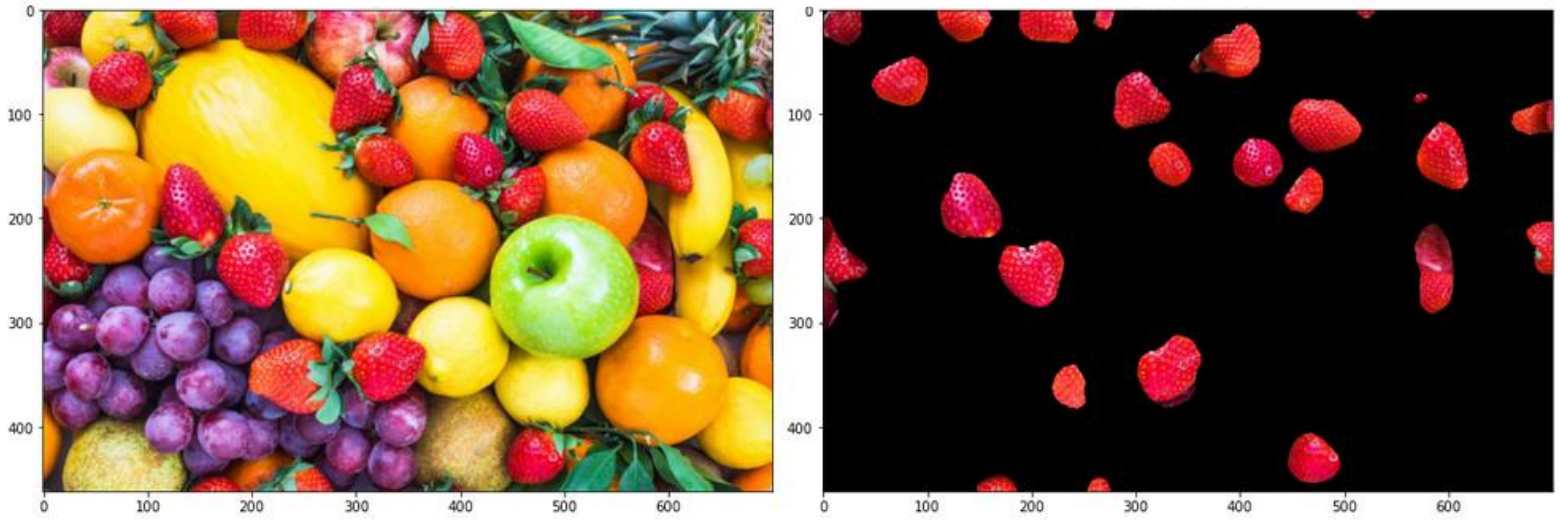
Solution: MATLAB Program

```
17 —    onergb = [40 40 200];
18 —    onexyz = rgbtoxyz*onergb';
19 —    onelab(1) = 25.0*(100.0*onexyz(2)/xyz_max(2))^(1/3) - 16.0;
20 —    onelab(2) = 500.0*[(onexyz(1)/xyz_max(1))^(1/3) - (onexyz(2)/xyz_max(2))^(1/3)];
21 —    onelab(3) = 200.0*[(onexyz(2)/xyz_max(2))^(1/3) - (onexyz(3)/xyz_max(3))^(1/3)];
22 —    pos = exp(-0.5*((onelab(2)-Lab_mean(2))/Lab_st(2))^2) ;
23 —    pos = pos*exp(-0.5*((onelab(3)-Lab_mean(3))/Lab_st(3))^2);
24 —    onergb
25 —    onelab
26 —    possibility_of_blue_color = pos
```

Solution: Results

```
onergb =

    40    40   200


onelab =

   54.8381   23.2977  -59.5676


possibility_of_blue_color =

    0.9480
```

More Examples of Doing Recognition of Colors

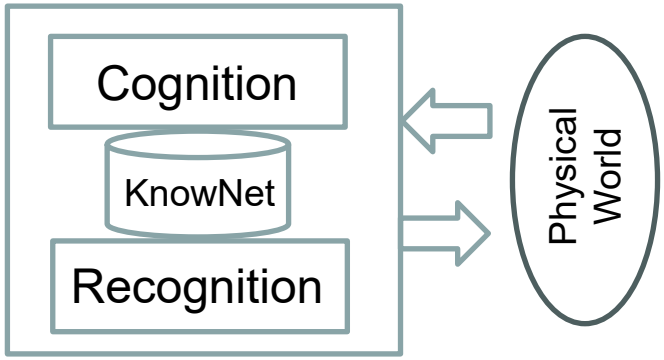


Outline of Lecture 4

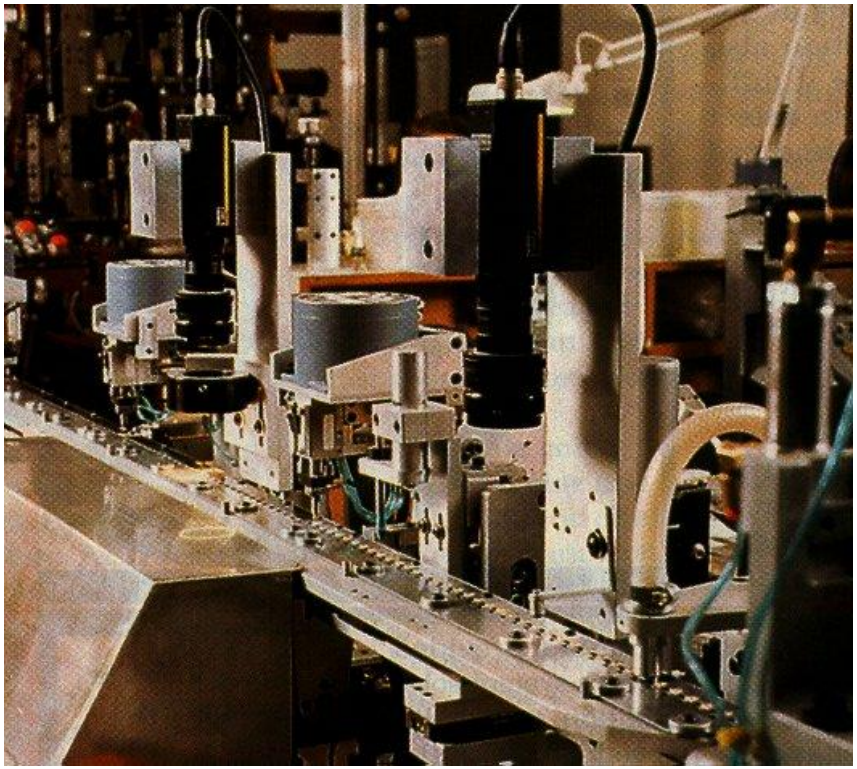
- Principle of Recognition (Understanding)
- Principle of Artificial Neural Network
- Principle of RCE Neural Network

Revision

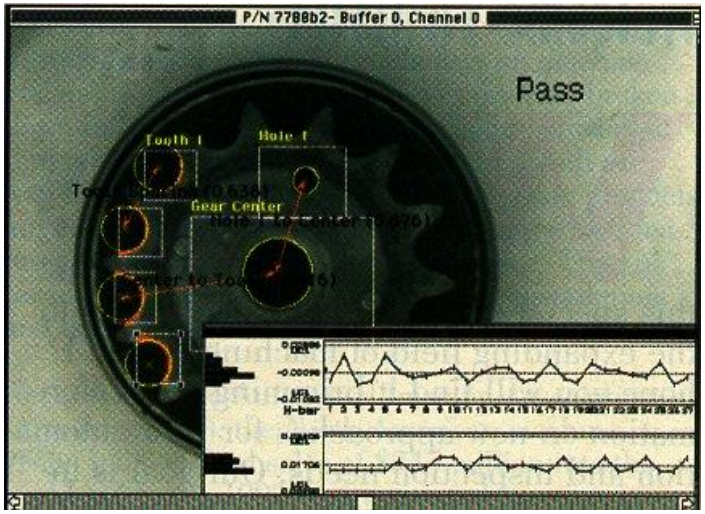
- Recogniton of Colors
- Recognition of Curves
- Recognition of Patterns
- Practices in MATLAB

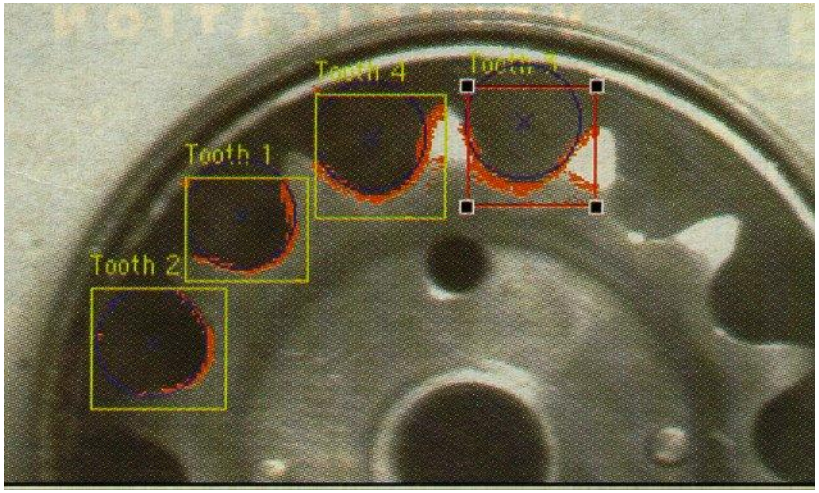


Example of Doing Recognition of Curves



Quality Control of Machined Parts

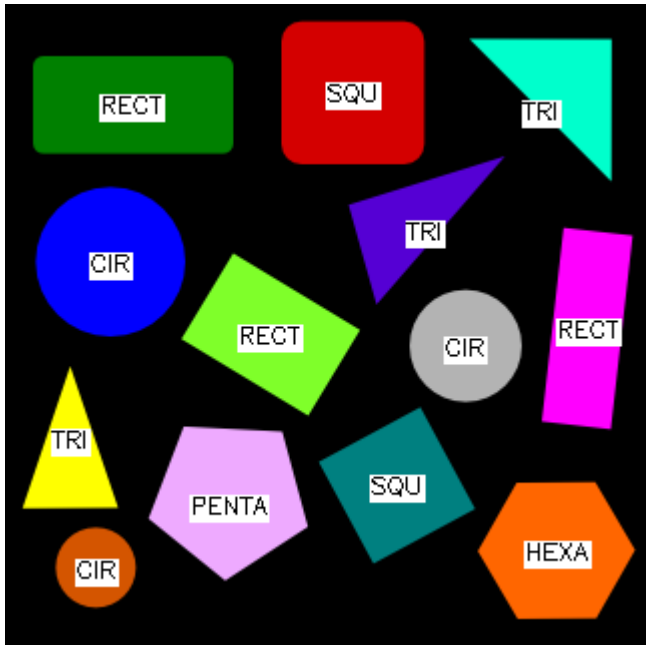




Expected Outcomes from Recognition ...

Input of Edge Pixels

Output of Polygons



Solution Toward Cognition and Recognition of Curves

Challenge

Digital Image

Step 1:
Edge Detection and Linking

Linked Edges

Step 2:
Curve Representation

Feature Vectors

Step 3a:
Supervised Training

Database of Learnt Curves

Step 3b:
Unsupervised Classification

Recognized Curves

Cognition

KnowNet

Recognition

Input 1: Any Image at Input

Images as 2D Matrices

- Image with Curves Inside

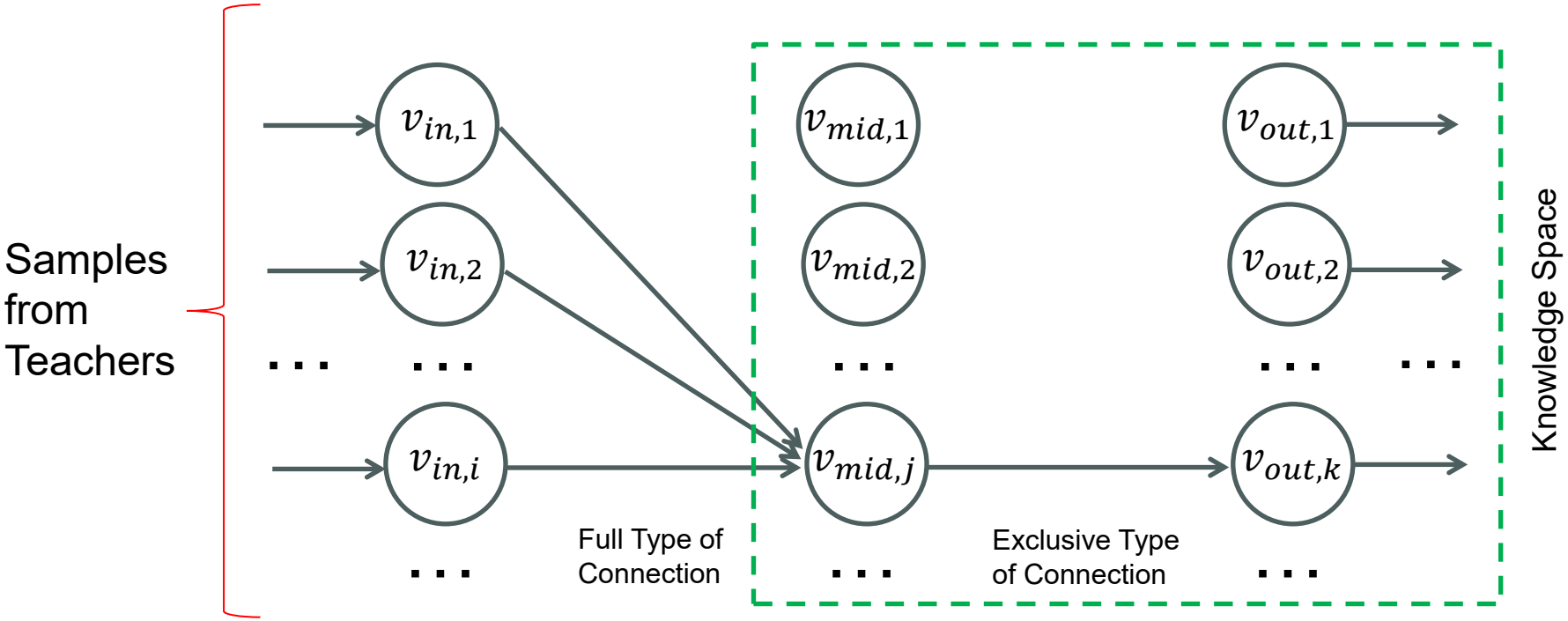
Data Structure of Pixel

```
typedef   struct
{
        short     r, g, b, I ;
        short     L, a, b ;
        short     u, v ;
        double    x, y, z ;
}
Pixel;
```

Input 2: Trained RCE Neural Network

- RCE neural network which has been trained by curve cognition

$$v_{mid,j} = \{a, b, \sigma, p_j, curve\_name, others\}$$



Output

Images as 2D Matrices

- Image with Recognized Curves

Data Structure of Pixel

```
typedef   struct
{
        short     r, g, b, i ;
        short     u, v ;
        double    x, y, z ;
        string    curve_name ;
        double    feature_vector[ ];
}
Pixel;
```

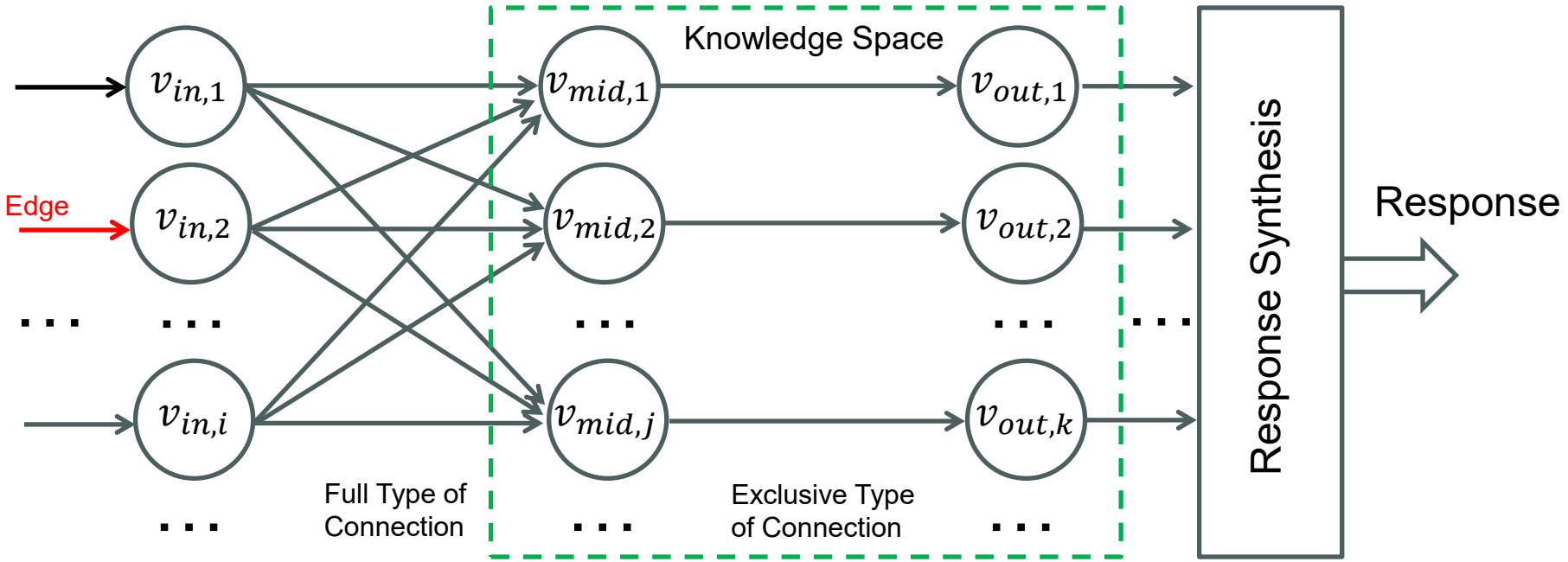
Question

- How to recognize or understand the curve that a pixel inside an image belongs to?

$$\forall k, R_{reply} = S(v_{out,k}), if\ p_k = max$$

Use of Programming Language

$S(\quad)$: Any function of synthesis

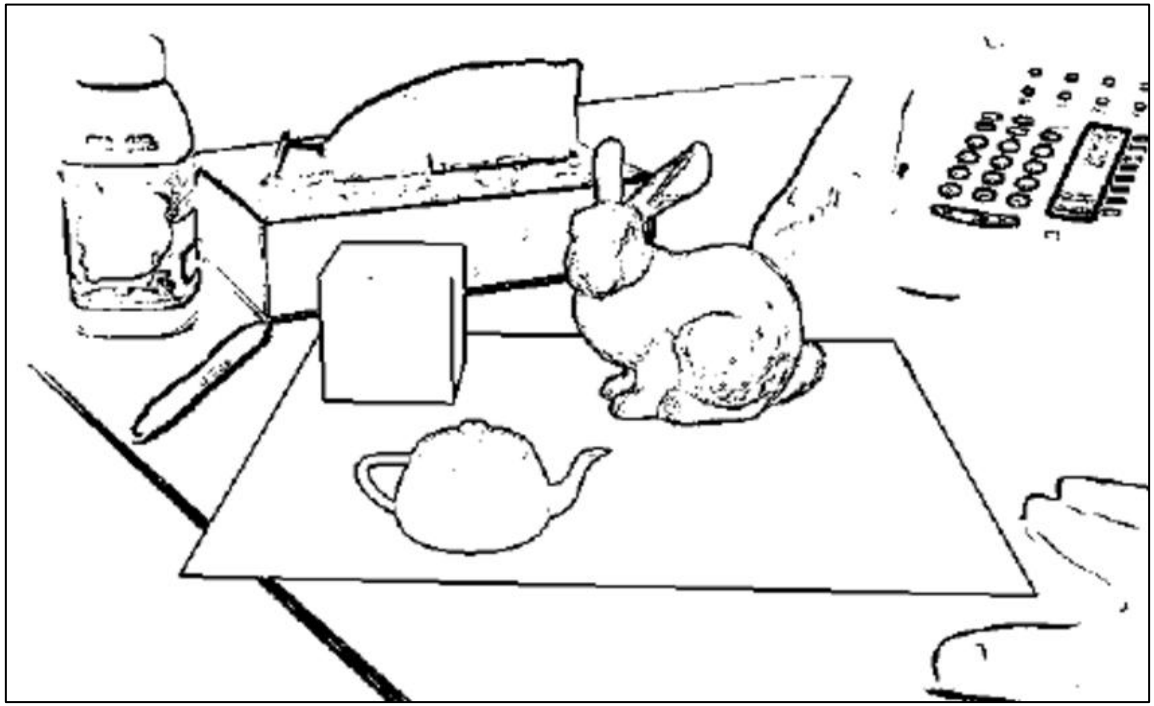


Step 1: To obtain visual samples (case of curves) from input image …

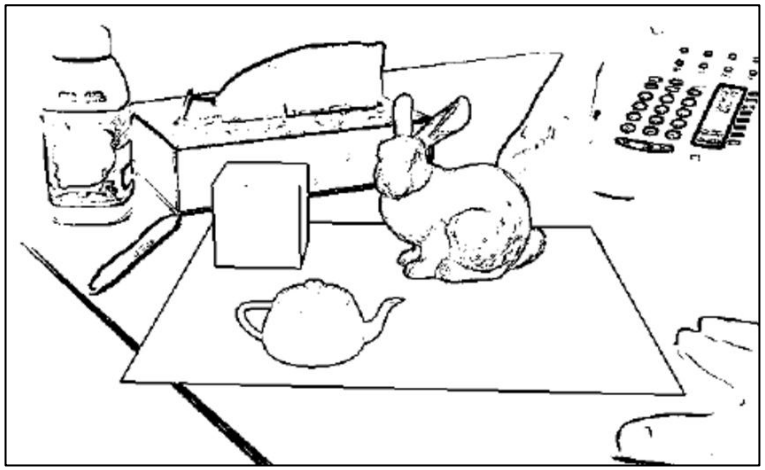
The general approach consists of two steps:

- Edge Detection
- Edge Linking

Color Image → Edge Image → Edge Samples



How to transform color image into edge image?



- Edge Detection Algorithm:

Step 1: Compute Horizontal Edges.

Step 2: Select the local maxima along the vertical axis as the edge pixels.

Step 3: Compute Vertical Edges.

Step 4: Select the local maxima along the horizontal axis as the edge pixels.

Step 5: Combine all the edge pixels to form the output.

Sample Program of Doing Edge Detection …

```
void EdgeDetection(void)
{
        int  i, j, v ;

        // Compute Horizontal Edges
        memset(gpBuffer1, 255, 512*512) ;
        for (i = 1 ; i < 511 ; i++)
        {
            for (j = 1 ; j < 511 ; j++)
            {
                v = abs(gpBuffer0[(i-1)*512+j-1] + pBuffer0[(i-1)*512+j] +  gpBuffer0[(i-1)*512+j+1] -
                    gpBuffer0[(i+1)*512+j-1] - gpBuffer0[(i+1)*512+j] - gBuffer0[(i+1)*512+j+1]) ;
                if (v >= 255)   gpBuffer1[i*512+j] = 0;
                if (v > 35 && v < 255) gpBuffer1[i*512+j] = (unsigned char) (255 - v) ; // change it to minimum
            }
        }
        SelectMaximaAlongColumn() ; // select local minima

        // Compute Vertical Edges
        memset(gpBuffer2, 255, 512*512) ;
        for (i = 1 ; i < 511 ; i++)
        {
            for (j = 1 ; j < 511 ; j++)
            {
                v = abs(gpBuffer0[(i-1)*512+j-1] + gpBuffer0[(i)*512+j-1] + gpBuffer0[(i+1)*512+j-1] -
                    gpBuffer0[(i-1)*512+j+1] - gpBuffer0[(i)*512+j+1] - gpBuffer0[(i+1)*512+j+1]) ;
                if (v >= 255)   gpBuffer2[i*512+j] = 0 ;
                if (v > 35 && v < 255) gpBuffer2[i*512+j] = (unsigned char) (255 - v) ; // change it to minimum
            }
        }
        SelectMaximaAlongRow() ; // select local minima

        // Display Horizontal Edges

        // Display Vertical Edges

        // Combined Edgemap
        memset(gpBuffer0, 255, 512*512) ;
        for (i = 0; i < 512*512 ; i++)
            if (gpBuffer1[i]  == 0 || gpBuffer2[i] == 0) gpBuffer0[i] = 0 ; // combine local minima
}
```

Set Background to be White

*

+1	+1	+1
0	0	0
-1	-1	-1

Ideal horizontal edge

Negative Edge Map

*

+1	0	-1
+1	0	-1
+1	0	-1

Ideal vertical edge

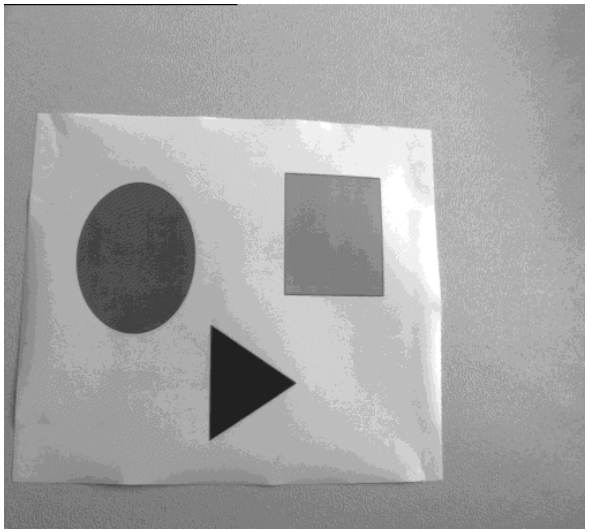
Negative Edge Map

```
static SelectMaximaAlongColumn()
{
        int         i, j, v ;
        unsigned short temporaryBuffer[512*512] ;

        // step 1: smoothing in vertical direction
        memset(temporaryBuffer, 255*5, 512*512) ;
        for (i = 2 ; i < 512 - 2; i++)
        {
            for (j = 0 ; j < 512; j++)   // compute weighted sum
            {
                    temporaryBuffer[i*512+j] = 1*gpBuffer1[(i-2)*512+j] +
                                               2*gpBuffer1[(i-1)*512+j] +
                                               3*gpBuffer1[i*512+j] +
                                               2*gpBuffer1[(i+1)*512+j] +
                                               1*gpBuffer1[(i+2)*512+j] ;
            }
        }

        // step 2: select local minima in vertical direction
        memset(gpBuffer1, 255, 512*512) ;
        for (i = 2 ; i < 512 - 2; i++)
        {
            for (j = 0 ; j < 512; j++)
            {
                    v = temporaryBuffer[i*512+j] ;
                    if ( v < temporaryBuffer[(i-2)*512+j] &&
                         v < temporaryBuffer[(i-1)*512+j] &&
                         v < temporaryBuffer[(i+1)*512+j] &&
                         v < temporaryBuffer[(i+2)*512+j])
                    {
                            gpBuffer1[i*512+j] = (unsigned char) 0 ;  //use 0 as minimum value
                    }
            }
        }
}
```

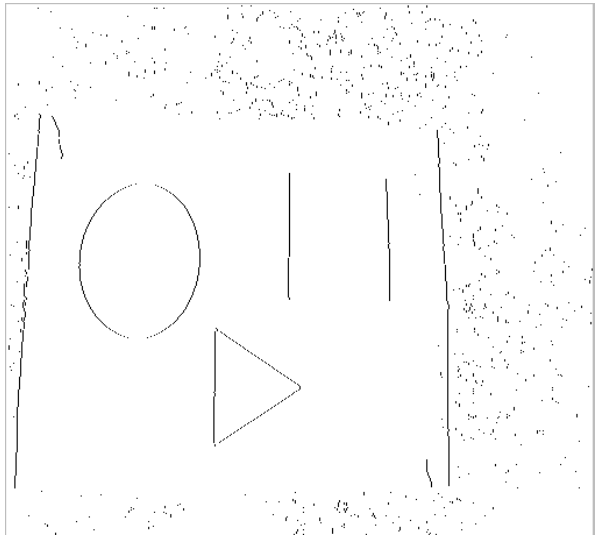
```
static SelectMaximaAlongRow()
{
        int         i, j, v ;
        unsigned short temporaryBuffer[512*512] ;

        // step 1: smoothing in horizontal direction
        memset(temporaryBuffer, 255*5, 512*512) ;
        for (i = 0 ; i < 512; i++)
        {
            for (j = 2 ; j < 512 - 2 ; j++)  // compute weighted sum
            {
                    temporaryBuffer[i*512+j] = 1*gpBuffer2[i*512+j-2] +
                                               2*gpBuffer2[i*512+j-1] +
                                               3*gpBuffer2[i*512+j] +
                                               2*gpBuffer2[i*512+j+1] +
                                               1*gpBuffer2[i*512+j+2] ;
            }
        }

        // step 2: select local minima in horizontal direction
        memset(gpBuffer2, 255, 512*512) ;
        for (i = 0 ; i < 512; i++)
        {
            for (j = 2 ; j < 512 - 2 ; j++)
            {
                    v = temporaryBuffer[i*512+j] ;
                    if ( v < temporaryBuffer[i*512+j-2] &&
                         v < temporaryBuffer[i*512+j-1] &&
                         v < temporaryBuffer[i*512+j+1] &&
                         v < temporaryBuffer[i*512+j+2])
                    {
                            gpBuffer2[i*512+j] = (unsigned char) 0 ; //use 0 as minimum value
                    }
            }
        }
}
```

Example of Results

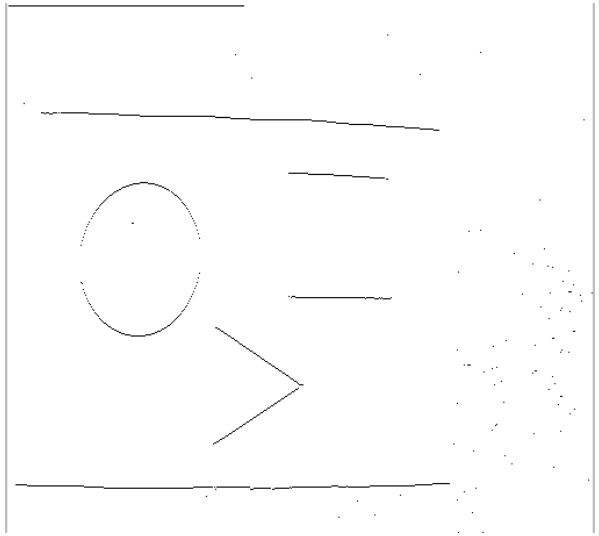
Input Image



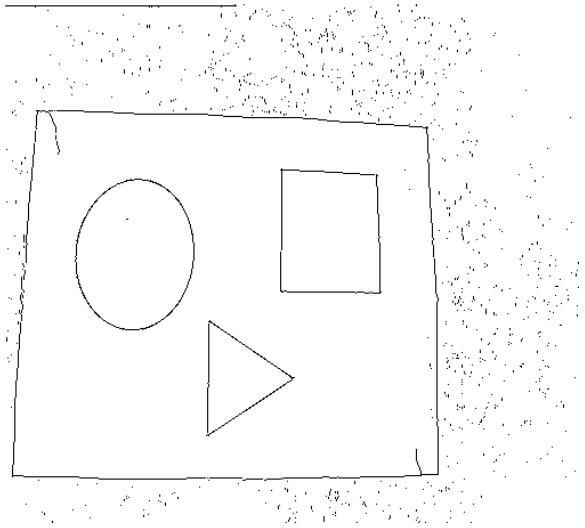
Vertical Edges



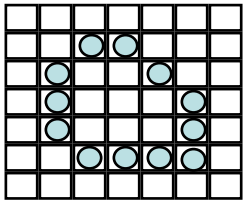
Horizontal Edges



Combined Edges



How to transform edge image into edge samples?

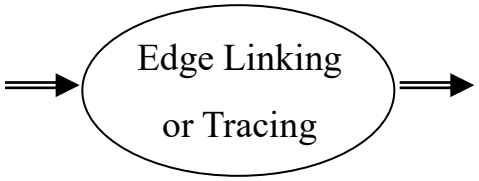


Main idea of doing edge linking:

- Scan input image of edges.
- If edge pixel at (i,j) is unlabeled, label it and define it as the currently labeled pixel.
- From the eight neighboring locations of (i,j), choose an edge pixel according to a user-defined condition (e.g. nearest to the center of unlabeled edges within a window).
  - If such a pixel exists, label it and define it as the currently-labeled edge pixel, and repeat the procedure;
  - Otherwise, a sample curve has been traced (i.e. found) and labeled.

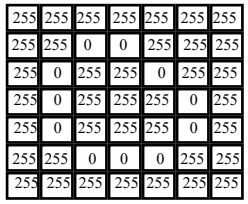
Illustration of Linked Edges (i.e. Curves)

	0	1	2	3	4	5	6	7	8	9	10	11
0	255	255	255	255	255	255	255	255	255	255	255	255
1	255	0	0	0	0	255	255	255	0	0	255	255
2	255	0	255	255	0	255	255	0	255	255	0	255
3	255	0	255	255	0	255	255	0	255	255	0	255
4	255	0	255	255	0	255	255	0	255	255	0	255
5	255	0	255	255	0	255	255	255	0	0	255	255
6	255	0	255	255	0	255	255	255	255	255	255	255
7	255	0	0	0	0	255	255	255	255	255	255	255
8	255	255	255	255	255	255	255	255	255	255	255	255
9	255	255	255	255	255	255	255	255	255	255	255	255



	0	1	2	3	4	5	6	7	8	9	10	11
0	255	255	255	255	255	255	255	255	255	255	255	255
1	255	1	1	1	255	255	255	255	2	2	255	255
2	255	1	255	255	1	255	255	2	255	255	2	255
3	255	1	255	255	1	255	255	2	255	255	2	255
4	255	1	255	255	1	255	255	2	255	255	2	255
5	255	1	255	255	1	255	255	255	2	2	255	255
6	255	1	255	255	1	255	255	255	255	255	255	255
7	255	255	1	1	255	255	255	255	255	255	255	255
8	255	255	255	255	255	255	255	255	255	255	255	255
9	255	255	255	255	255	255	255	255	255	255	255	255

An Edge Linking Algorithm …



255	255	255	255	255	255	255
255	255	0	0	255	255	255
255	0	255	255	0	255	255
255	0	255	255	255	0	255
255	0	255	255	255	0	255
255	255	0	0	0	255	255
255	255	255	255	255	255	255

Step 1: Initialize the label map and set the label variable to 1 (n = 1).

Step 2: Scan the edge map row by row and column by column.

Step 3: If encounter an unlabeled edge-pixel, define it as the current edge-pixel.
Compute the estimated center of unlabeled edges within a window which is centered at the current edge-pixel.

Step 4: Assign the label "n" to the current edge-pixel. Now, this edge is labeled.

Step 5: Find all the unlabeled neighbors around the current edge-pixel.
If there is no unlabeled neighbor, a curve has been found and go to Step 7.

Step 6: Among the unlabeled neighbors, choose the one which is the nearest edge to the estimated center. Define it as the new current edge-pixel.
Repeat Step 4, Step 5 and Step 6.

Step 7: Increment the label (n = n + 1) and continue from Step 2.

Xie M. and Thonnat M. 1992, An Algorithm for Finding Closed Curves, Pattern Recognition Letters, Vol. 13, No. 1, pp. 73-81.

Sample Program of Doing Edge Linking ...

Edgemap

255	255	255	255	255	255	255
255	255	0	0	255	255	255
255	0	255	255	0	255	255
255	0	255	255	255	0	255
255	0	255	255	255	0	255
255	255	0	0	0	255	255
255	255	255	255	255	255	255

Labelmap

255	255	255	255	255	255	255
255	255	255	255	255	255	255
255	255	255	255	255	255	255
255	255	255	255	255	255	255
255	255	255	255	255	255	255
255	255	255	255	255	255	255
255	255	255	255	255	255	255

```
/* assuming that the pixel values of all the edge pixels are "0" */

unsigned char edgemap[512*512] ;
unsigned char labelmap[512*512] ;

static  int     cx, cy ;        /* the estimated center of unlabeled edges      */
static  int     newx, newy ;  /* the location of next "current edge-pixel"     */
static  double  min_distance ;
static  int     sx, sy ;

static  void  ComputeCenterWithinWindowAt(int x0, int y0)
{
        /* update the center's coordinates (pcx, pcy) */
}

static  void  VerifyNextUnlabeledEdge(int x, int y)
{
       /* verify the neighbor whether it is the nearest one to (pcx, pcy) */
}

static void  LinkEdge(int x, int y, int n)
{
      /* recursively link the edge-pixels */
}

main(int argc, char **argv)
{
      int    x,  y, n ;

      n = 1 ;   sx = 10 ;      sy = 10 ;          // set the size of window to be 10x10
      memset(labelmap, 255, 512*512) ;   // clear the content of labelmap

      for (y = 1 ; y < 511 ; y++)
            for (x = 1 ; x < 511 ; x++)
                  if (edgemap[y*512+x] == 0 && labelmap[y*512+x] == 255)
                  {
                        ComputeCenterWithinWindowAt(x, y);
                        LinkEdgeAtCurrentLocation(x, y, n) ; // do edge linking
                        n++ ;
                  }
}
```

```
static  void   VerifyNextUnlabeledEdge(int x, int y)
{
        double  distance ;
        // the candidate must be unlabeled edge
        if (edgemap[y*512+x] != 0 || labelmap[y*512+x] != 255)
              return ;

        distance = sqrt((cx - x)*(cx - x) + (cy - y)*(cy - y)) ;
        if (distance < min_distance)
        {
                    min_distance = distance;
                    newx = x ;
                    newy = y ;
        }
}
```

Edge Map

255	255	255	255	255	255	255
255	255	0	0	255	255	255
255	0	255	255	0	255	255
255	0	255	255	255	0	255
255	0	255	255	255	0	255
255	255	0	0	0	255	255
255	255	255	255	255	255	255

Label Map

255	255	255	255	255	255	255
255	255	255	255	255	255	255
255	255	255	255	255	255	255
255	255	255	255	255	255	255
255	255	255	255	255	255	255
255	255	255	255	255	255	255
255	255	255	255	255	255	255

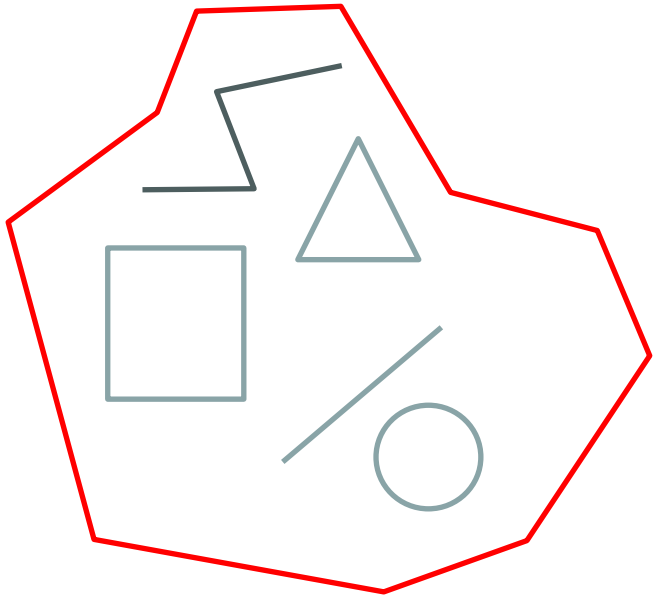
```
static  void  ComputeCenterWithinWindowAtCentre(int x0, int y0)
{
        int  r, c, n ;
        // use unlabeled edges to compute the center inside window
        cx = 0 ; cy = 0 ; n = 0 ;
        for (r = -sy/2 ; r < sy/2 ; r++)  // r is the increment of row index
              // c is the increment of column index
              for (c = -sx/2; c < sx/2 ; c++)
                    if (edgemap[(y0+r)*512+x0+c] == 0 &&
                        labelmap[(y0+r)*512+x0+c] == 255)
                    {
                        cx = cx + x0 + c ;
                        cy = cy + y0 + r ;
                        n = n + 1 ;
                    }
        if (n != 0)
        {
             cx = cx/n ;
             cy = cy/n ;
        }
}
```

```
static void  LinkEdgeAtCurrentLocation(int x, int y, int n)
{
        labelmap[y*512+x] = n ;    /*  assign label to the pixel  */

        min_distance = 1000000.0 ;

        // what should be the next edge to be linked?
        VerifyNextUnlabeledEdge(x-1, y-1) ;   // neighbor 1
        VerifyNextUnlabeledEdge(x, y-1) ;     // neighbor 2
        VerifyNextUnlabeledEdge(x+1,y-1);     // neighbor 3
        VerifyNextUnlabeledEdge(x-1,y) ;      // neighbor 4
        VerifyNextUnlabeledEdge(x+1,y) ;      // neighbor 5
        VerifyNextUnlabeledEdge(x-1, y+1) ;   // neighbor 6
        VerifyNextUnlabeledEdge(x, y+1) ;     // neighbor 7
        VerifyNextUnlabeledEdge(x+1, y+1) ; // neighbor 8

        if (min_distance != 1000000.0)
              LinkEdgeAtCurrentLocation(newx, newy, n) ;
}
```

(NOTE: This is called Dynamic Programming)

What are the possible outputs from the process of edge linking?

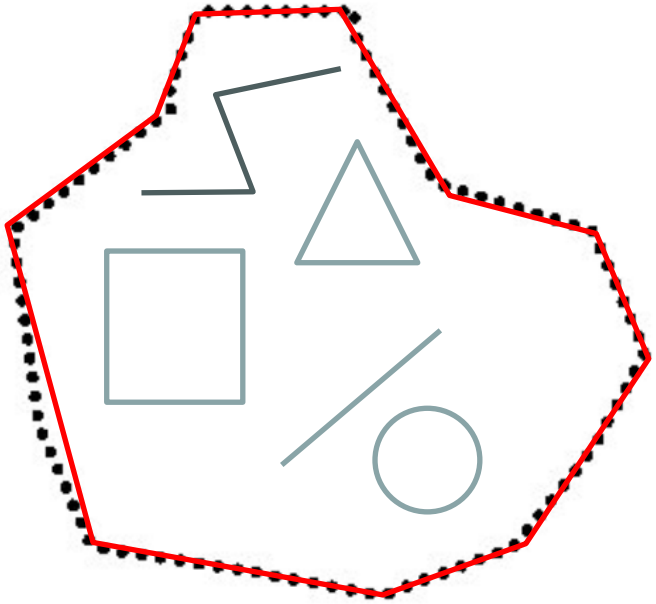
- Linked Edges from Line Segments
- Linked Edges from Curve Segments
- Linked Edges from Circles
- Linked Edges from Ellipses



- Linked Edges from Polylines or Closed Contours (e.g. triangles, rectangles, polygons, etc)

Step 2: To represent each sample of linked edges

- Solution is to use a set of pixel coordinates: $\{(u_i, v_i), i = 1, \ldots, n\}$



	0	1	2	3	4	5	6	7	8	9	10	11
0	255	255	255	255	255	255	255	255	255	255	255	255
1	255	1	1	1	255	255	255	255	2	2	255	255
2	255	1	255	255	1	255	255	2	255	255	2	255
3	255	1	255	255	1	255	255	2	255	255	2	255
4	255	1	255	255	1	255	255	2	255	255	2	255
5	255	1	255	255	1	255	255	255	2	2	255	255
6	255	1	255	255	1	255	255	255	255	255	255	255
7	255	255	1	1	255	255	255	255	255	255	255	255
8	255	255	255	255	255	255	255	255	255	255	255	255
9	255	255	255	255	255	255	255	255	255	255	255	255

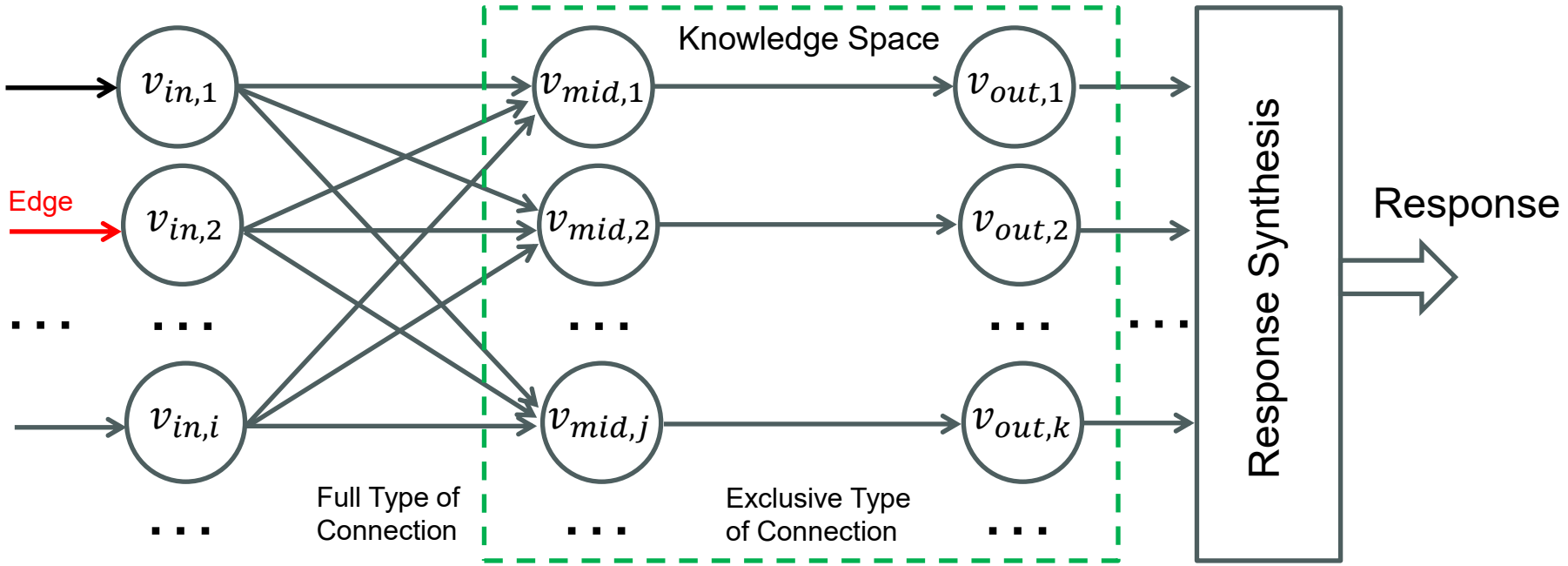
Step 3b: To determine output

- For each pixel on a sample of linked edges (e.g., a line or a polyline),
  - First, compute the possibility value at each node inside the middle layer of the trained RCE neural network.
  - Subsequently, determine the output of curve name which corresponds to the maximum value among the above-computed possibilities.

$$\forall k, R_{reply} = S(v_{out,k}), if\ p_k = max$$

Use of Programming Language

$S(\quad)$: Any function of synthesis



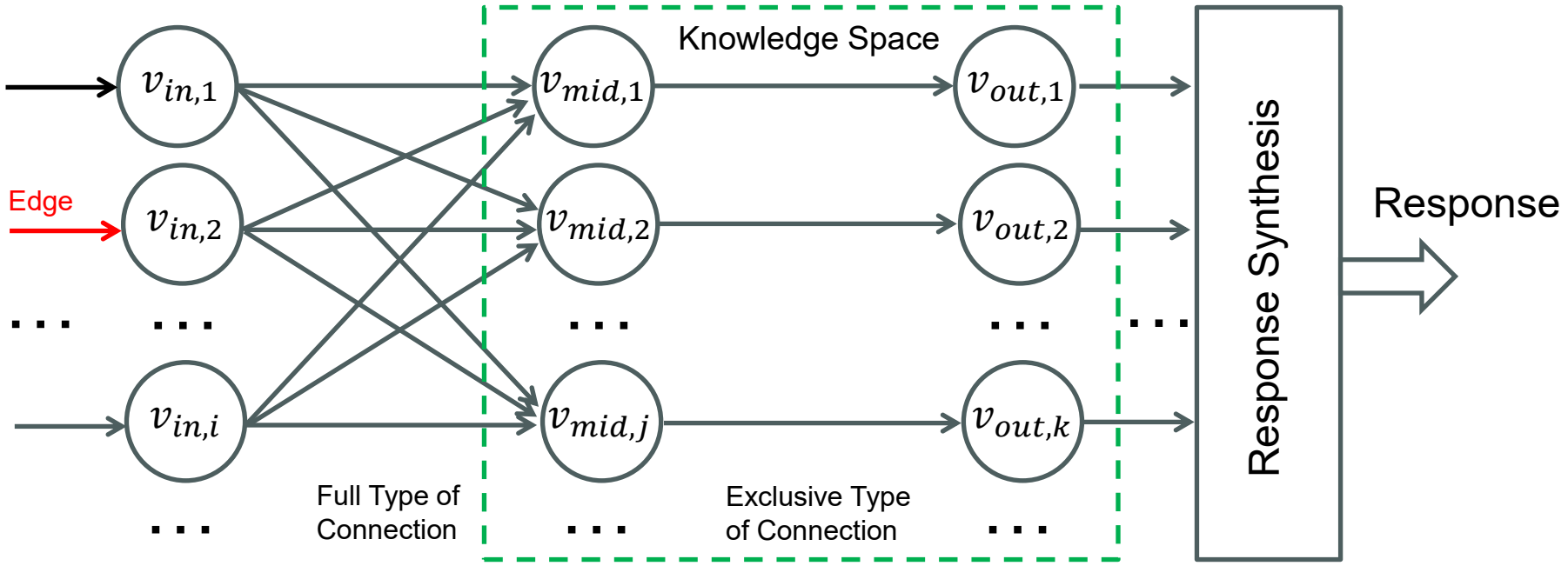
Step 3b (continued): To determine output

- Finally, synthesize the response using a natural language.

$$\forall k, R_{reply} = S(v_{out,k}), if\ p_k = max$$

Use of Programming Language

$S(\quad)$: Any function of synthesis



Simplified Example

- Do Edge Detection and Edge Linking.
- For each pixel of a sample of linked edges: $\{(u_i, v_i), i = 1,2, \dots\}$
  - Then, compute the possibility value at each node inside the middle layer of the trained RCE neural network:

$$p_k(u_i, v_i) = e^{-\frac{1}{2}\left(\frac{a \times u_i - v_i + b}{\sigma\sqrt{1+a^2}}\right)^2}, k = 1,2, \dots$$

  - Subsequently, determine the curve name with the maximum value among the above-computed possibilities:

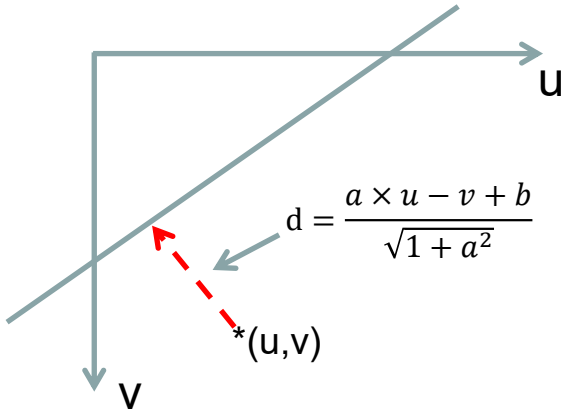
$$P_{max} = \max\{p_k(u_i, v_i), k = 1,2, \dots\}, \text{ if } P_{max} > P_{min}$$

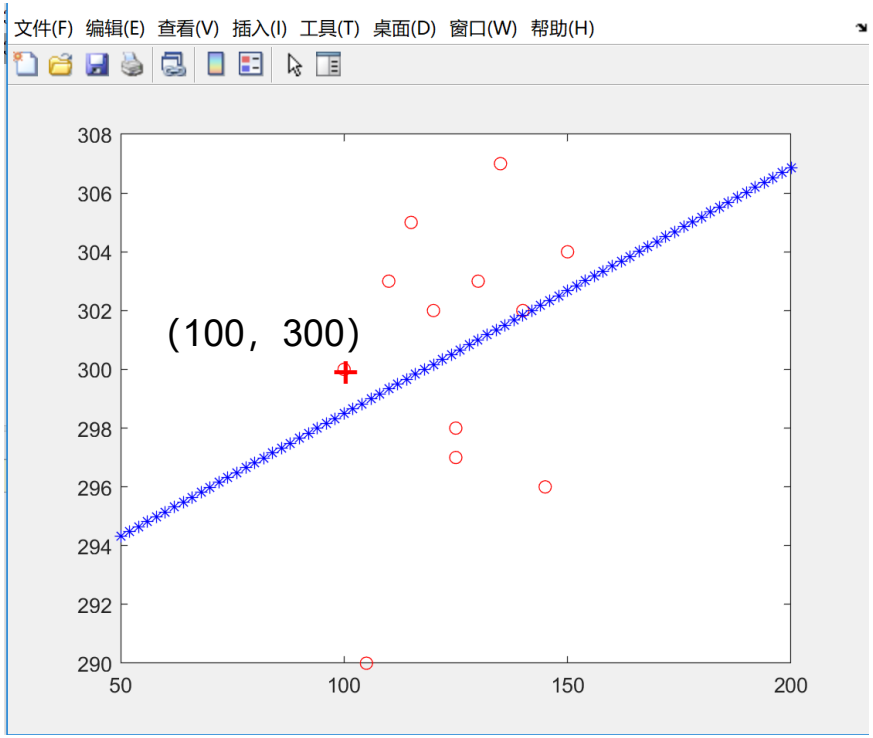
Lower Limit of P_{max}

- Finally, synthesize the response using a natural language.

Exercise

- We scan an image row by row and column by column. When we are at the location (100, 300), what is the possibility for this location to belong to a learnt line in which $(a, b, u_d) = (0.0836, 290.1288, 2.5134)$?



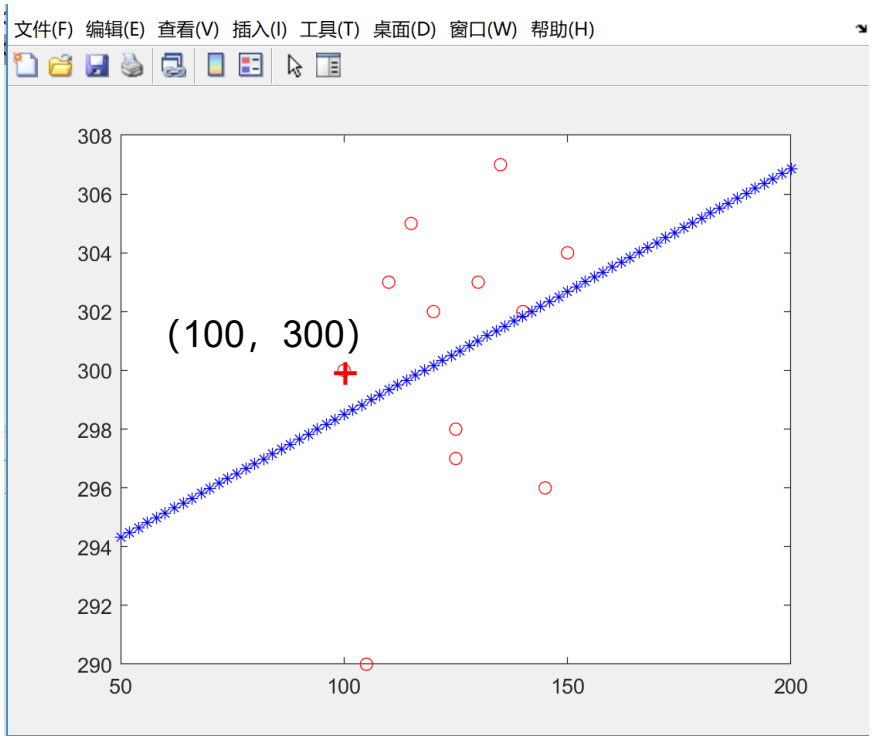


Solution

$$P_k(u,v) = e^{-\frac{1}{2}\left(\frac{a\times u - v + b}{\sigma\sqrt{1+a^2}}\right)^2} > P_{k,min}$$

```
17 —    u = 100; v = 300;
18 —    pos = exp(-0.5*((a*u-v+b)/(sigma*sqrt(1+a^2)))^2);
19 —    possibility = pos
```

```
possibility =

    0.8364
```

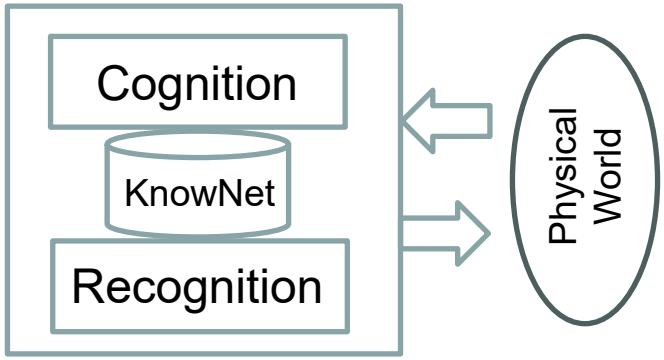


Outline of Lecture 4

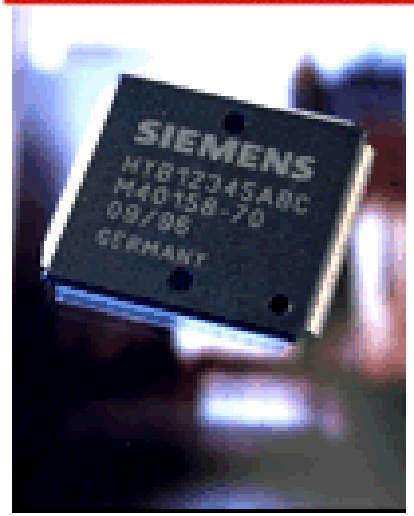
- Principle of Recognition (Understanding)
- Principle of Artificial Neural Network
- Principle of RCE Neural Network

Revision

- Recogniton of Colors
- Recognition of Curves
- Recognition of Patterns
- Practices in MATLAB

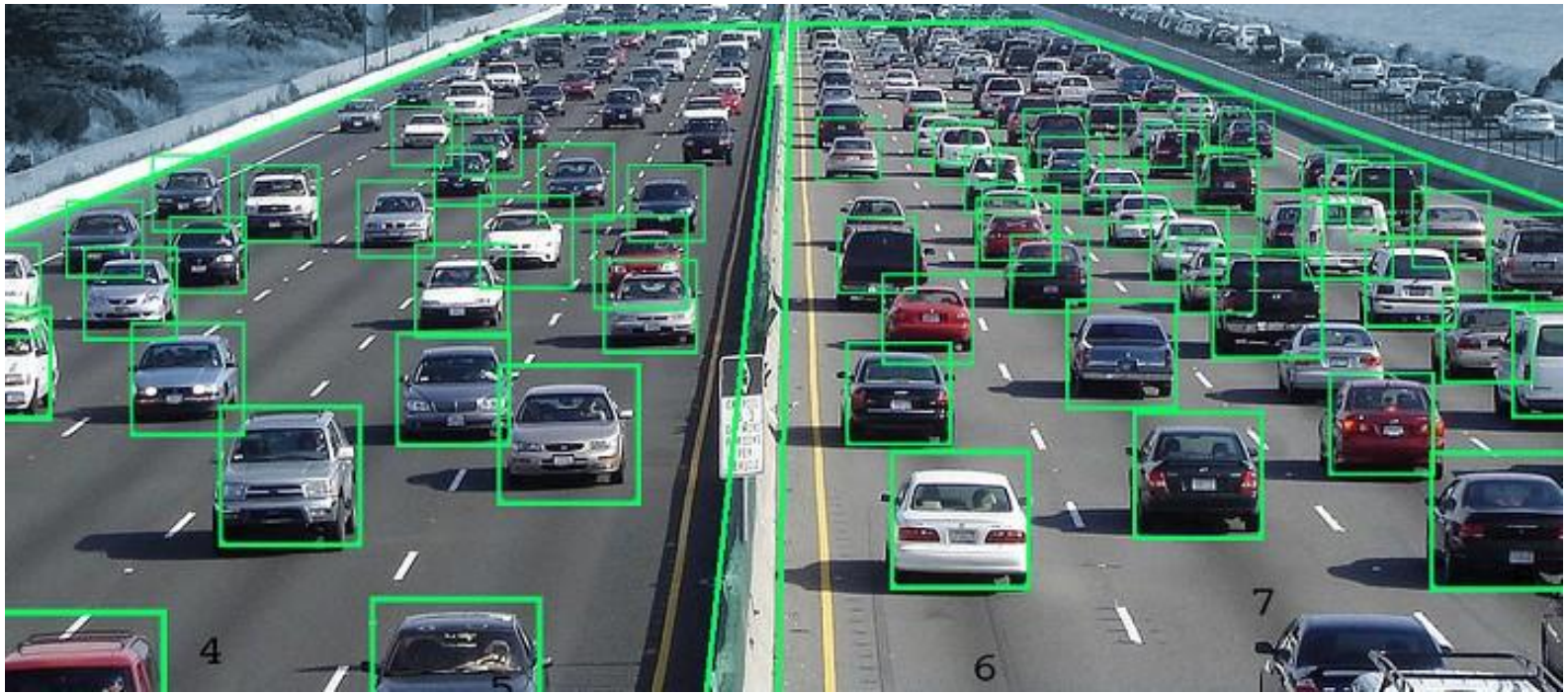


Example of Doing Recognition of Patterns in Quality Control …

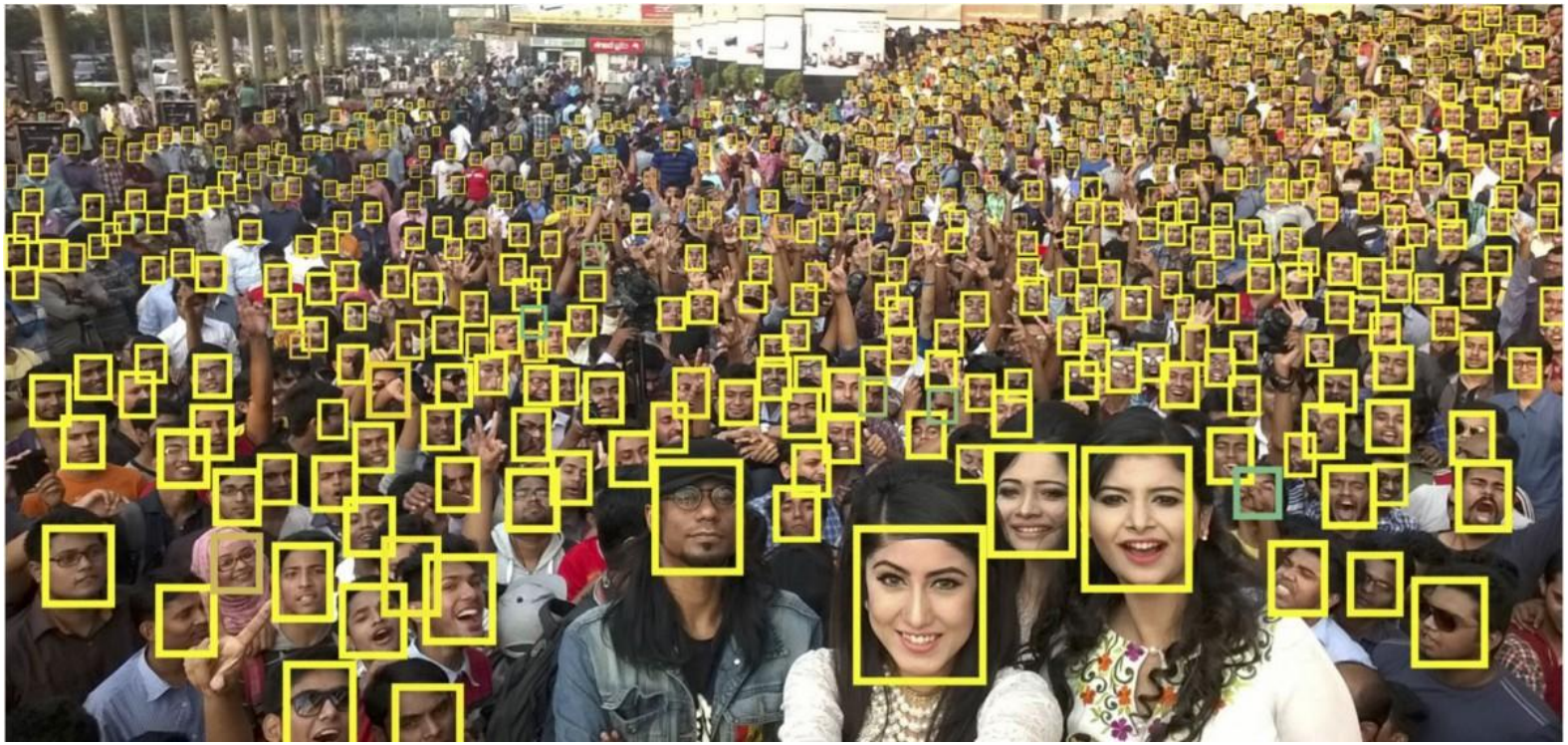


Quality Control of Printed Characters

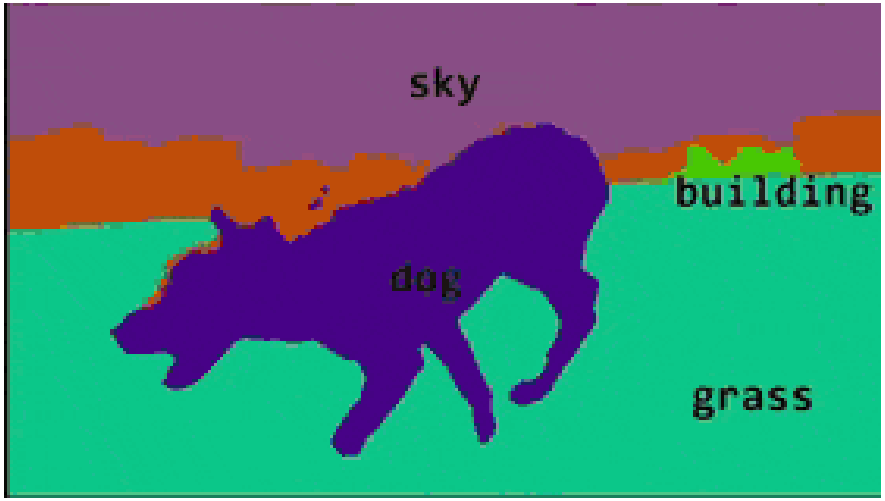
Example of Doing Recognition of Vehicles in Traffic Control ...

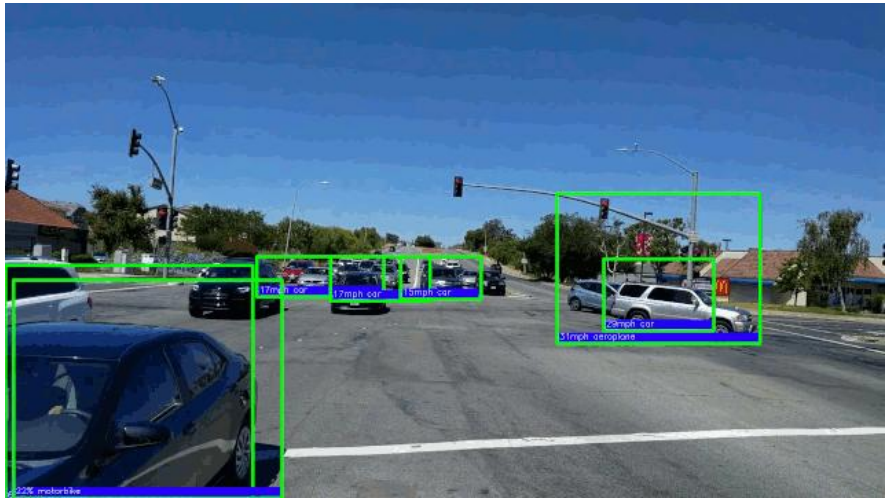


Example of Doing Recognition of Human Faces in Security Control …

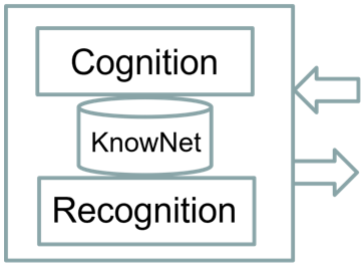


Expected Outcomes from Recognition ...





Solution Toward Cognition and Recognition of Patterns



Input 1: Any Image at Input

Images as 2D Matrices

- Image with Patterns Inside

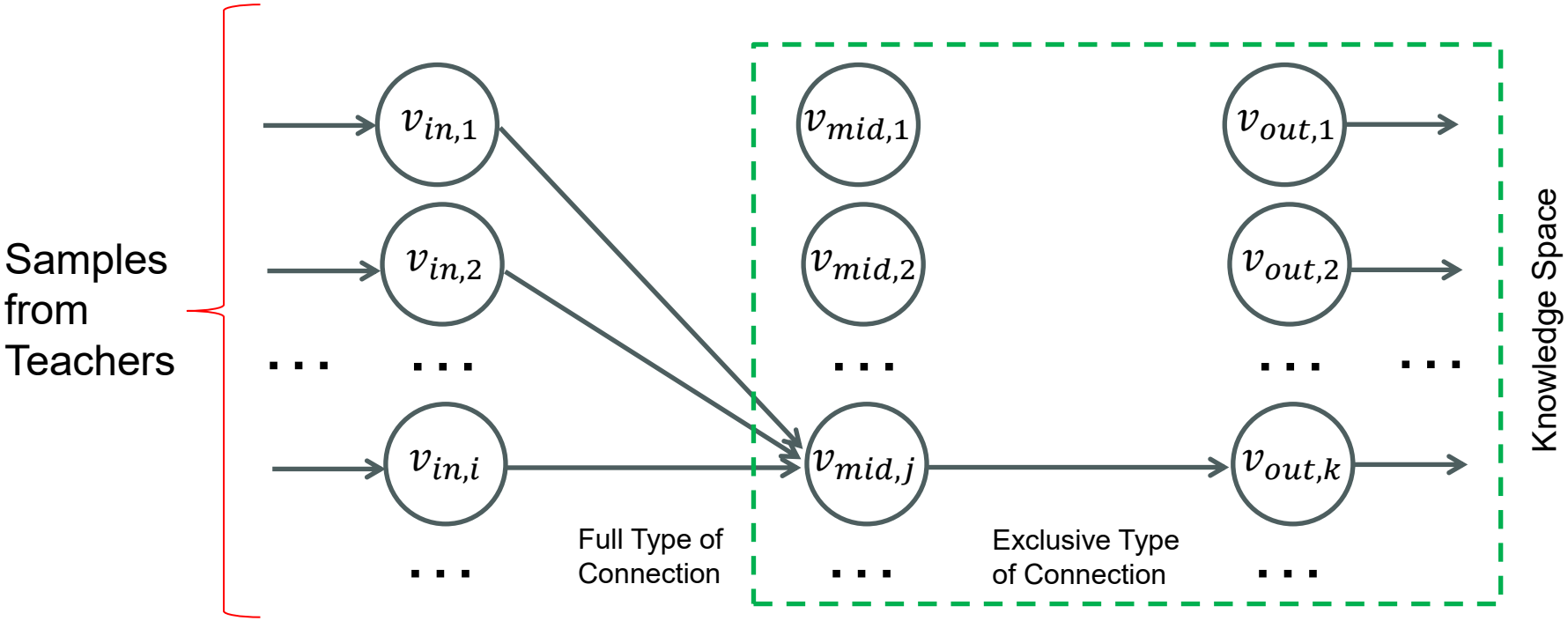
Data Structure of Pixel

```
typedef   struct
{
        short     r, g, b, I ;
        short     L, a, b ;
        short     u, v ;
        double    x, y, z ;
}
Pixel;
```

Input 2: Trained RCE Neural Network

- RCE neural network which has been trained by pattern cognition

$$v_{mid,j} = \{\mu_f, \sigma_f, p_j, pattern\_name, others\}$$



Output

Images as 2D Matrices

- Image with Recognized Patterns

Data Structure of Pixel

```
typedef   struct
{
        short     r, g, b, i ;
        short     u, v ;
        double    x, y, z ;
        string    pattern_name ;
        double  feature_vector[ ];
}
Pixel;
```

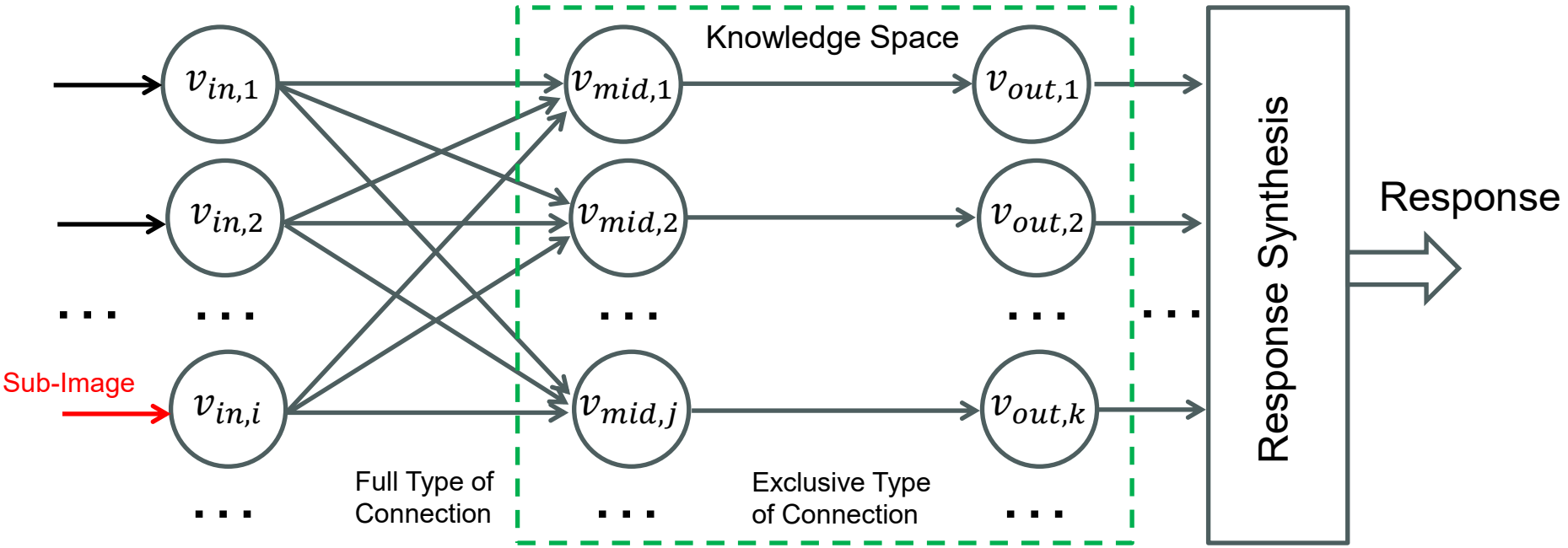
Question

- How to recognize or understand the pattern that the sub-image at a pixel inside an input image belongs to?

Use of Programming Language

$\forall k, R_{reply} = S(v_{out,k}), if\ p_k = max$

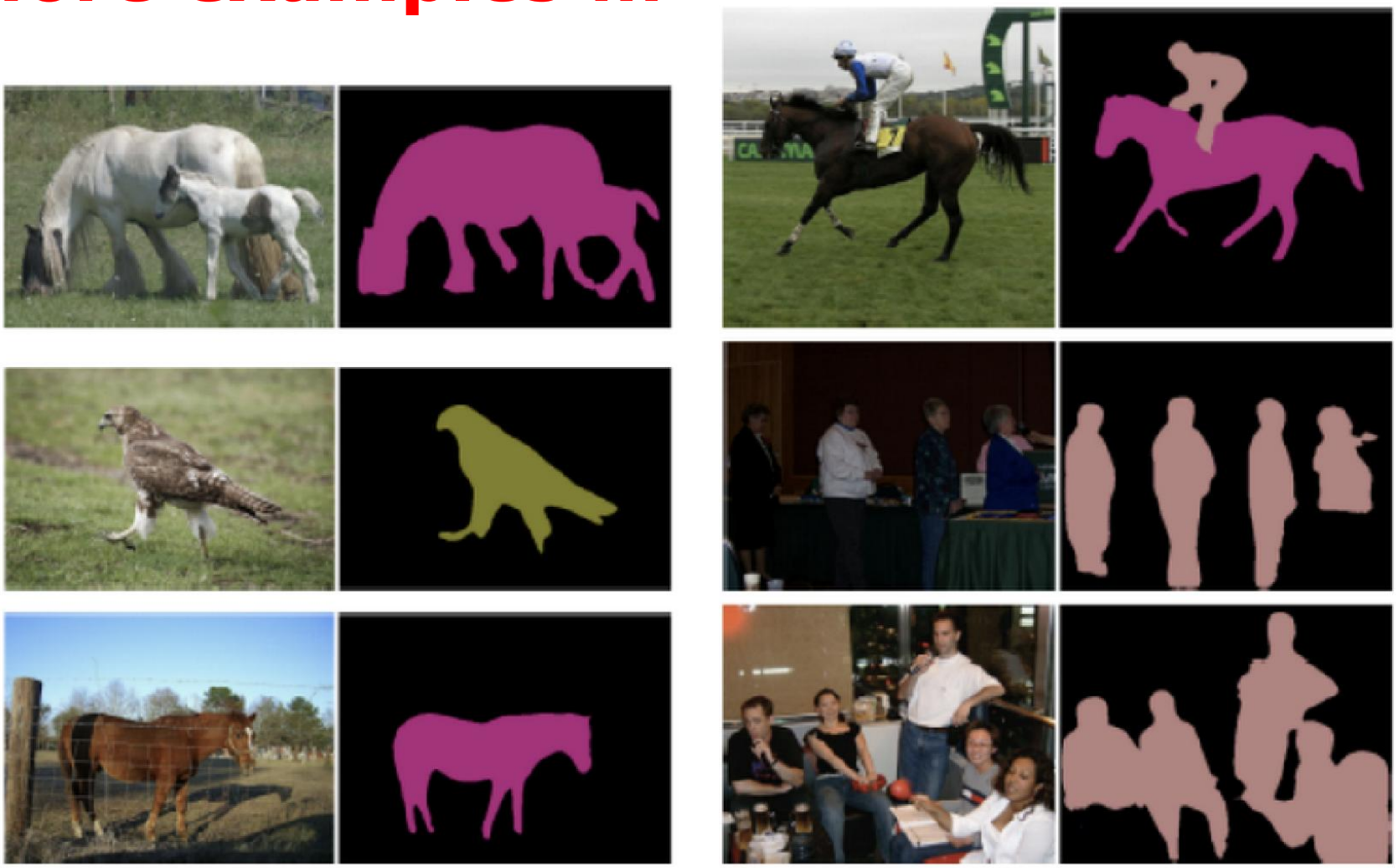
$S(\quad)$: Any function of synthesis



Step 1: To obtain visual samples (case of patterns) from input image?

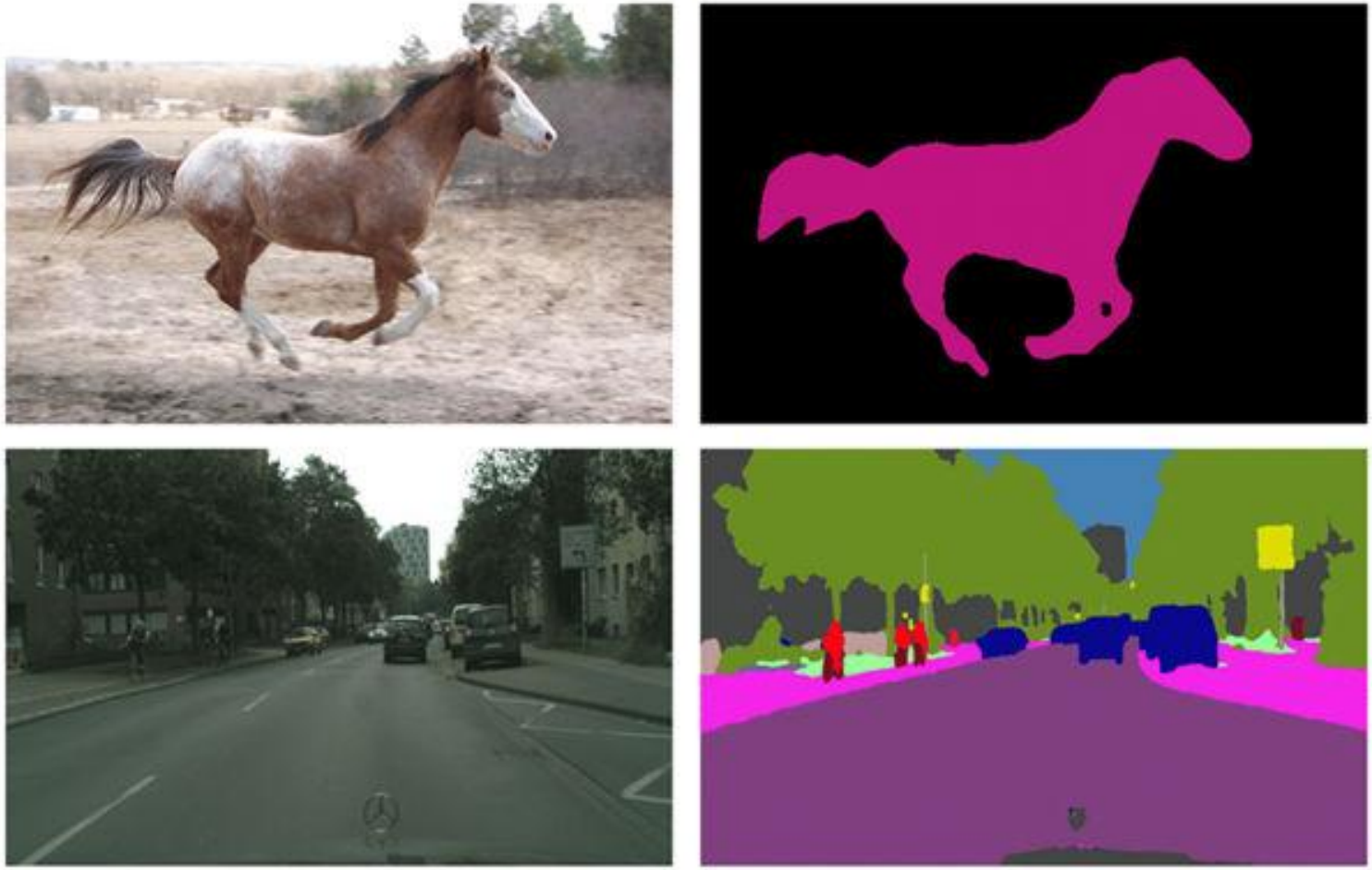
- Method 1: Thresholding (using intensities, colors, textures, motions, …)

More examples …

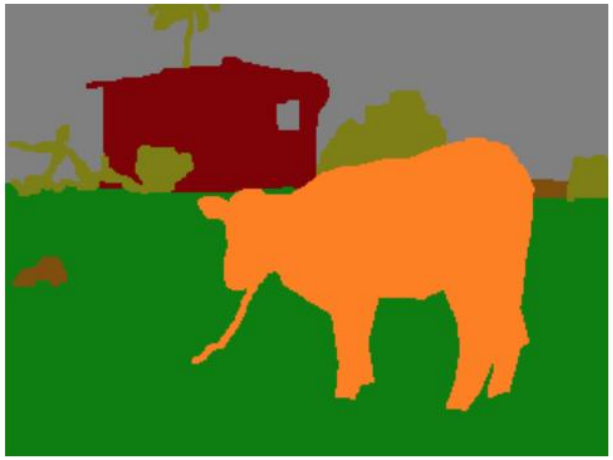


Step 1: Obtain visual samples (case of patterns) from input image? (continued)

- Method 2: Bottom-Up Grouping (e.g., using feature vectors …)



More examples ...

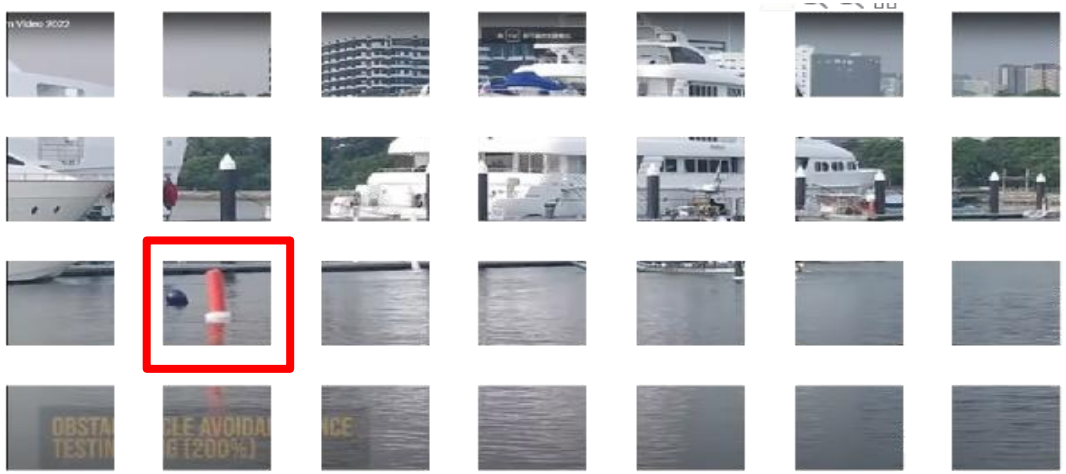


Step 1: To obtain visual samples (case of patterns) from input image? (continued)

- Method 3: Top-Down Sampling (e.g., doing sub-divisions …)

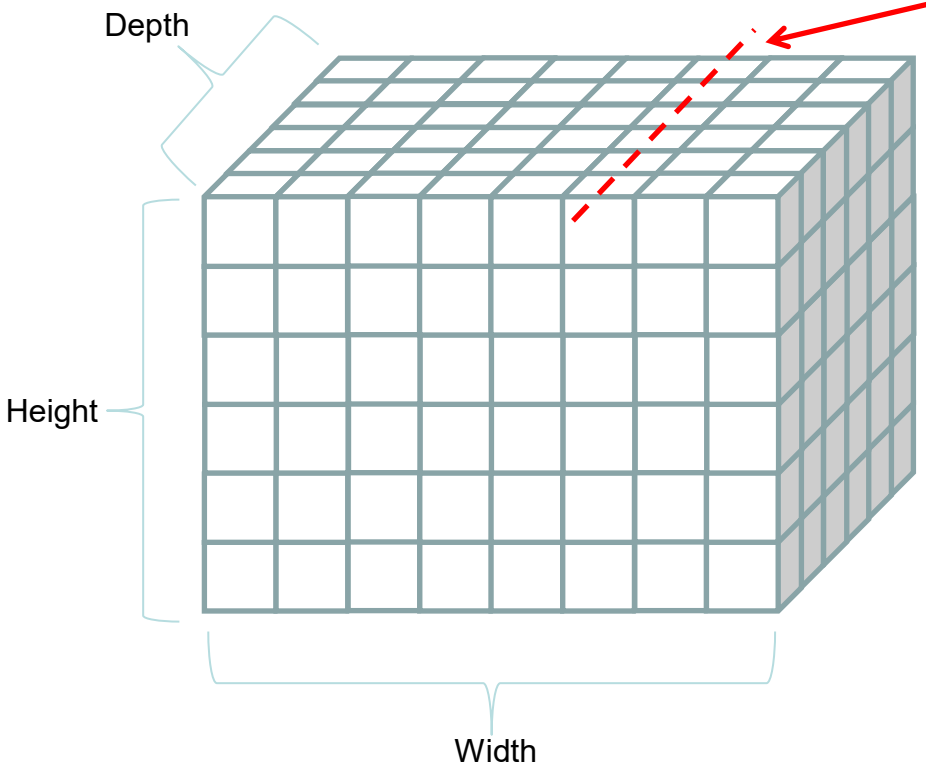
4x7 image samples in set S_{28}

Image Sampling



Step 2: To represent a sample pattern by a vector

- Increase the depth of visual pattern by doing Deep Convolution or Fourier Transform.
- Convert the 2D matrix of pixels' feature vectors into a single vector which is called Apparent/Deep Feature Vectors.



This series of values belongs to a pixel's feature vector

Size of visual pattern: N x M

Size of a Pixel's Feature Vector: L x 1

Size of a Pattern's Feature Vector:

$$(N \times M \times L) \times 1$$

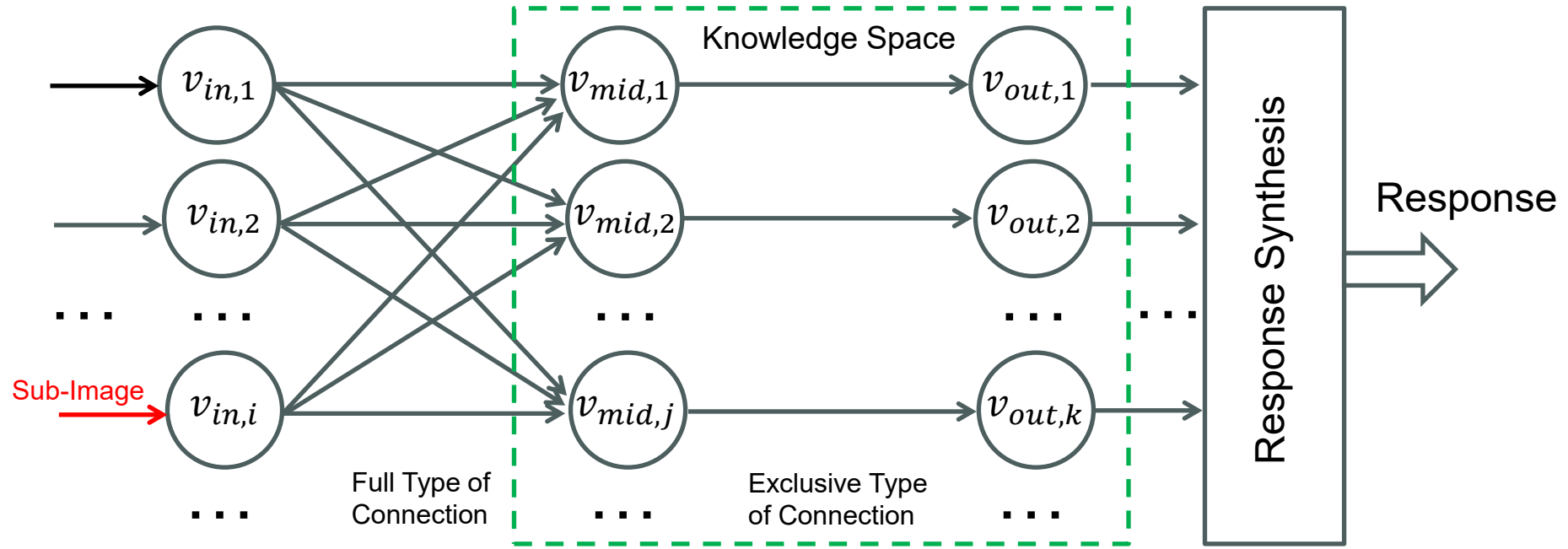
Step 3b: To determine output

- First, compute the possibility value at each node inside the middle layer of the trained RCE neural network.
- Subsequently, determine the output of pattern name which corresponds to the maximum value among the above-computed possibilities.

$$\forall k, R_{reply} = S(v_{out,k}), if\ p_k = max$$

Use of Programming Language

$S(\quad)$: Any function of synthesis



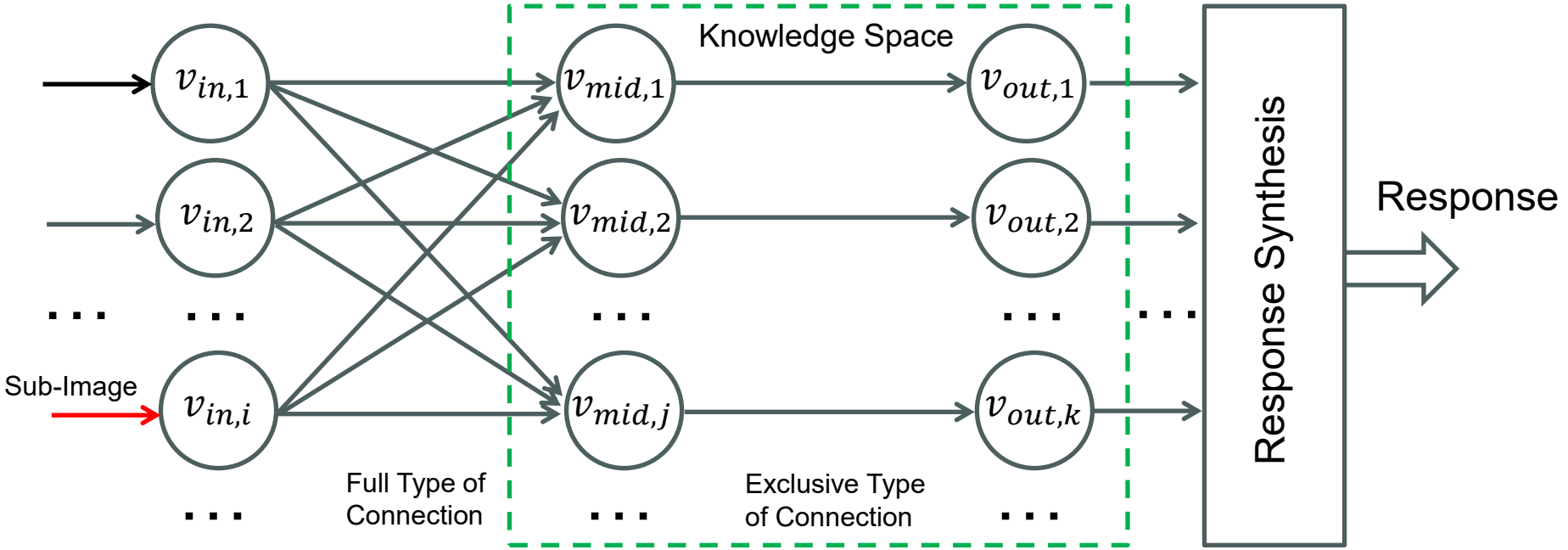
Step 3b (continued): To determine output

- Finally, synthesize the response using a natural language.

Use of Programming Language

$$\forall k, R_{reply} = S(v_{out,k}), if\ p_k = max$$

$S(\quad)$: Any function of synthesis



Simplified Example …

- Detect visual samples of patterns to be recognized.

Deep Vector

- For each detected pattern, represent it as a vector: $\{f_i, i = 1,2, \dots\}$

- Then, compute the possibility value at each node inside the middle layer of the trained RCE neural network:

$$p_k(f_i) = e^{-\frac{1}{2\sigma_f^2}(f_i-\mu_f)^t\,(f_i-\mu_f)}, k = 1,2, \dots$$

- Subsequently, determine the color name with the maximum value among the above-computed possibilities:

$$P_{max} = \max\{p_k(f_i), k = 1,2, \dots\}, \text{ if } P_{max} > P_{min}$$

Lower Limit of P_{max}

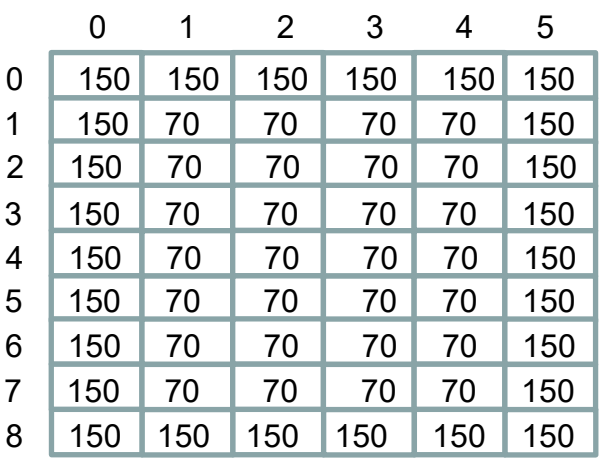
- Finally, synthesize the response using a natural language.

Exercise

- Given a learnt pattern of “rectangle” with the following mean vector and the variance of distances to the mean vector:

```
mu_f =

  Elements 1 to 17

    150   150   150   150   150   150   150    60    60    60    60   150   150

     60    60    60    60

  Elements 18 to 34

    150   150    60    60    60    60   150   150    60    60    60    60   150

    150    60    60    60

  Elements 35 to 51

     60   150   150    60    60    60    60   150   150    60    60    60    60

    150   150   150   150

  Elements 52 to 54

    150   150   150

sigma_f =

   86.4099
```

- What is the possibility for the image below to belong to the learnt pattern of “rectangle”?



	0	1	2	3	4	5
0	150	150	150	150	150	150
1	150	70	70	70	70	150
2	150	70	70	70	70	150
3	150	70	70	70	70	150
4	150	70	70	70	70	150
5	150	70	70	70	70	150
6	150	70	70	70	70	150
7	150	70	70	70	70	150
8	150	150	150	150	150	150

Solution:

MATLAB Program

```
42 —    n = 3 ;
43 —    for i=1:9
44 —        for j=1:6
45 —            f1((i-1)*6+j) = image1(i, j);
46 —            f2((i-1)*6+j) = image2(i, j);
47 —            f3((i-1)*6+j) = image3(i, j);
48 —            f4((i-1)*6+j) = image4(i, j);
49 —        end
50 —    end
51 —    f1 = f1'; % Change to column vector
52 —    f2 = f2'; % Change to column vector
53 —    f3 = f3'; % Change to column vector
54 —    f4 = f4'; % Change to Column vector
55
56 —    mu_f = (1/n)*(f1+f2+f3); % mean of three columns vectors
57 —    disp(mu_f');
58 —    e1 = f1 - mu_f; % error vector from image 1
59 —    e2 = f2 - mu_f; % error vector from image 2
60 —    e3 = f3 - mu_f; % error vector from image 3
61 —    variance_f = (1/n)*(e1'*e1+e2'*e2+e3'*e3);
62 —    sigma_f = sqrt(variance_f)
63
64 —    e4 = f4 - mu_f ;
65 —    pos = exp(-0.5*(f4-mu_f)'*(f4-mu_f)/variance_f)
66
```

For Cognition

For Recognition

Solution:

Results

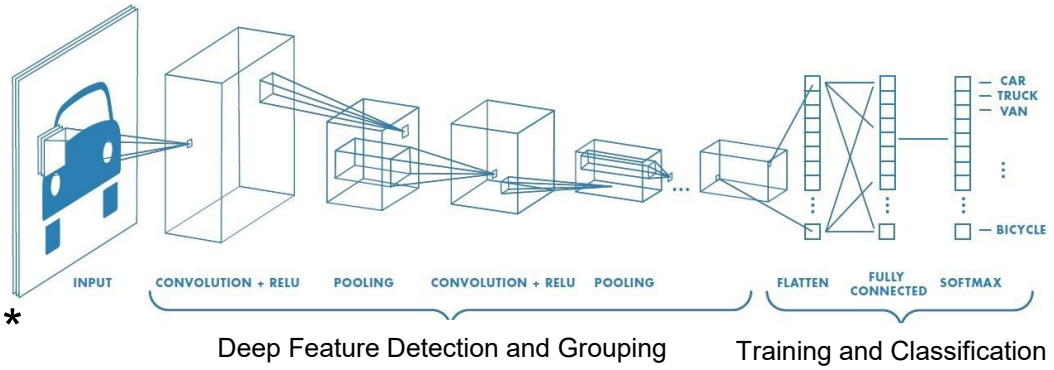
```
32 —    image4 = [150   150   150   150   150   150
33              150    70    70    70    70   150
34              150    70    70    70    70   150
35              150    70    70    70    70   150
36              150    70    70    70    70   150
37              150    70    70    70    70   150
38              150    70    70    70    70   150
39              150    70    70    70    70   150
40              150   150   150   150   150   150];
```

```
pos =

    0.8290
```

82.9% of possibility!

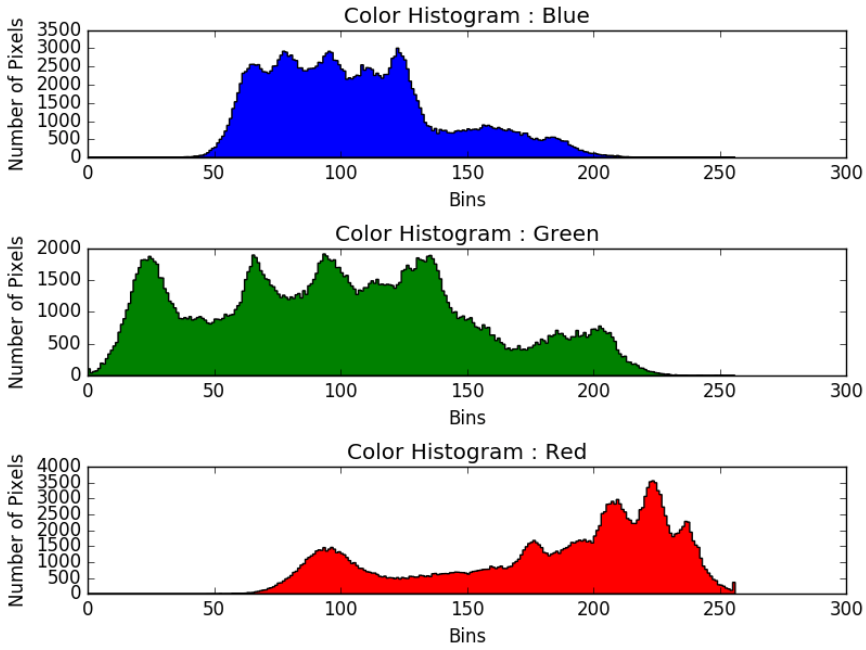
There are other ways of doing implementations …

- Forward Template Matching
- Inverse Template Matching
- Prediction/Verification Methods*



What to use?

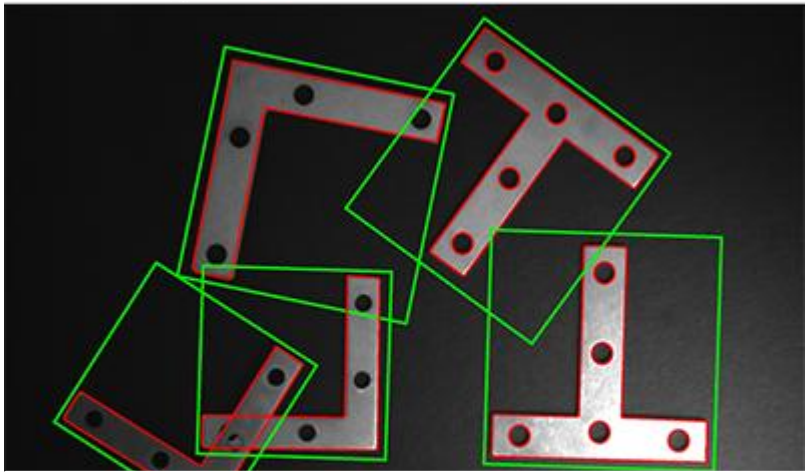
- Apparent/Deep Feature Vectors
- Histogram vector of colors/Intensities
- Histogram vector of gradients
- Histogram vector of textures, etc.



Principle of Forward Template Matching

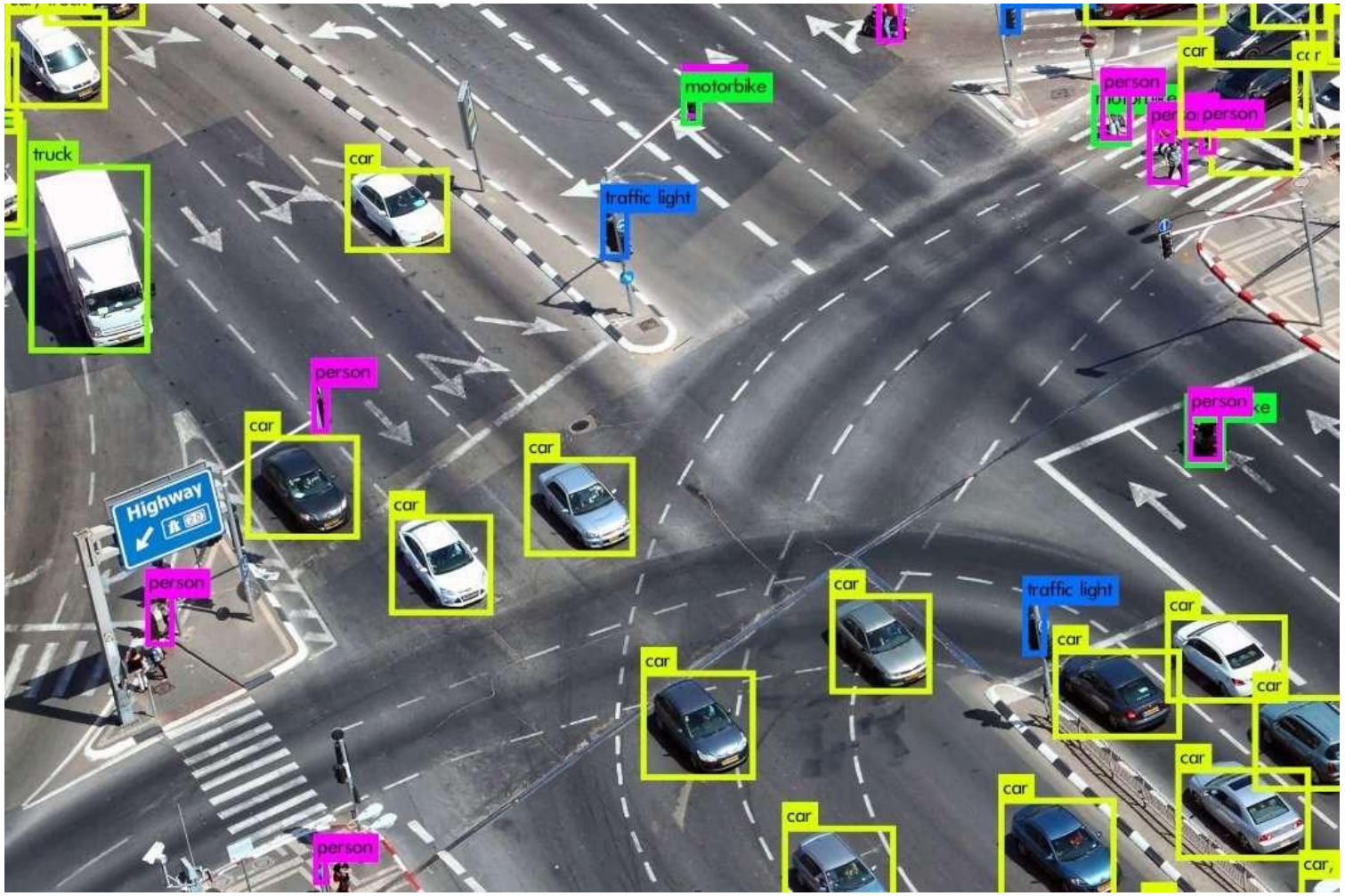
- Input:
  - Images with Patterns to be Recognized
  - Templates (References) of Learnt Patterns
- Output:
  - Poses of Detected Patterns
- Procedure:
  - Create template-images at selected discrete orientations.
  - Divide an input image into a matrix of sub-images, each of which has the same size of template-image.
  - Compute the dissimilarity between each pair of sub-image and template-image.
  - Output the ones with the minimum values of dissimilarity.

Two Templates



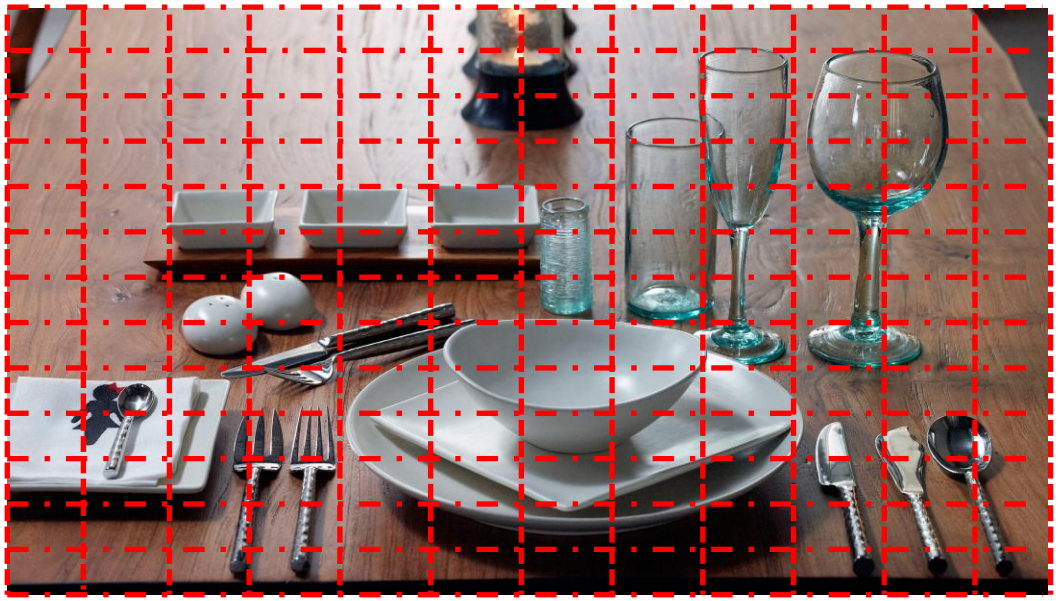
Results of Template Matching

Example of Results ...

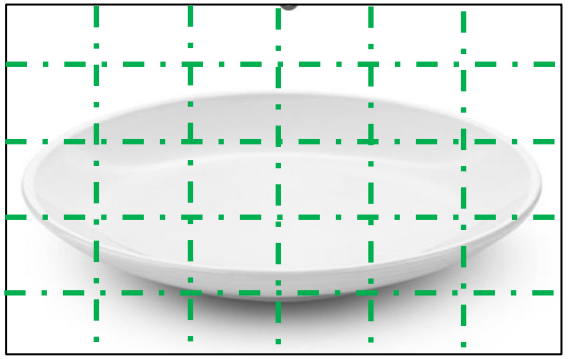


Principle of Inverse Template Matching

Input Image



- Input:
  - Image (i.e. original image)
  - Template (i.e. original template)
- Output:
  - Poses of Detected Patterns
- Procedure:
  - Consider an input image as a template and Divide it into a matrix of so-called sub-template-images.
  - Consider an original template as an input image and Divide it into a matrix of so-called sub-input-images.
  - Find the best matches between sub-template-images and sub-input-images.
  - Connect the sub-template-images (from original input image) which match with the sub-input-images (from original template image).

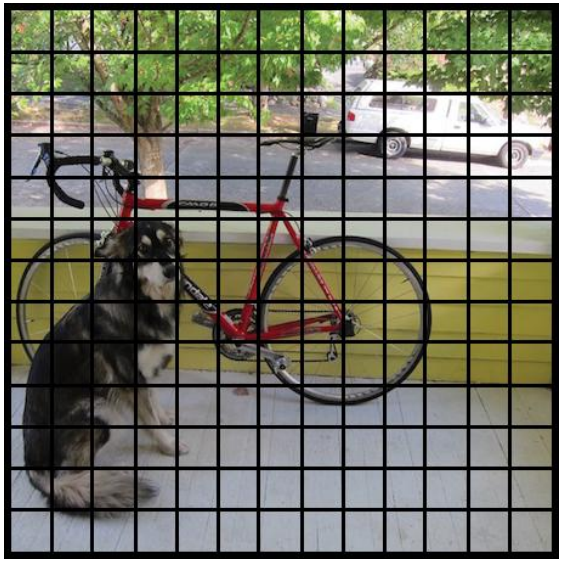


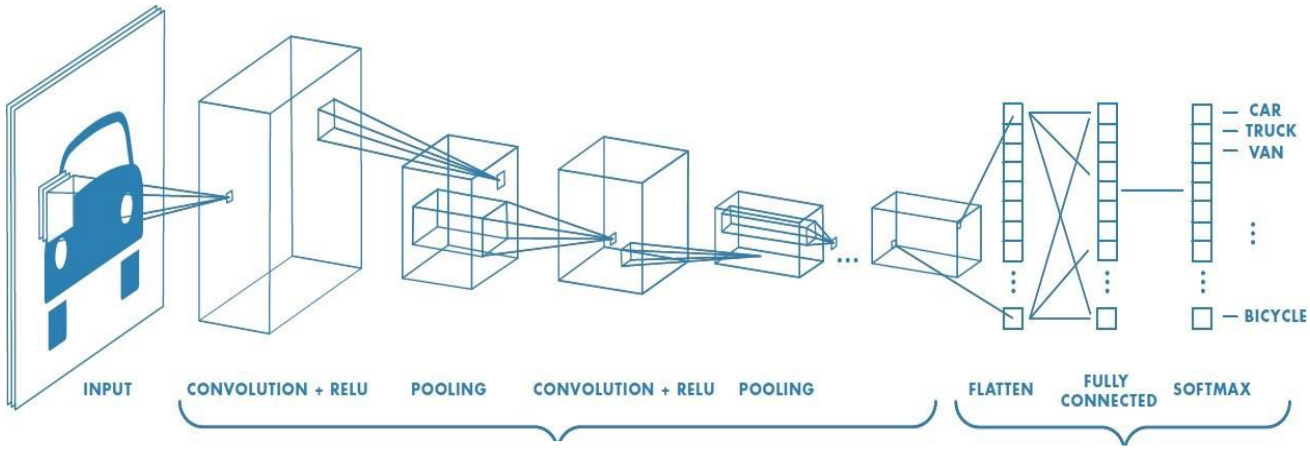
Template Image

Prediction/Verification Methods …

- Bottom-up Strategy (e.g. YOLO Algorithm):
  - Sliding Windows with Variable Strides
  - Spatial Division

- Top-down Strategy:
  - Prediction of one sample, two samples, three samples …





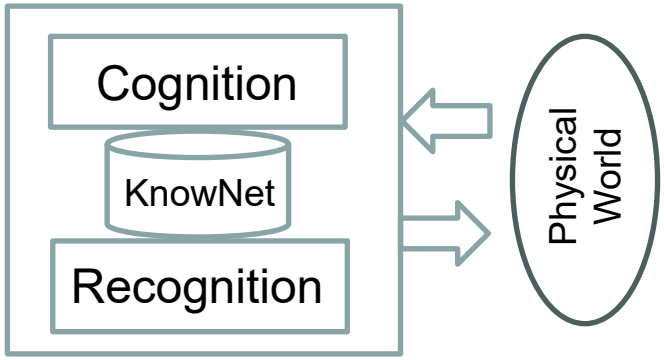
(e.g. YOLO Algorithm)

Outline of Lecture 4

- Principle of Recognition (Understanding)
- Principle of Artificial Neural Network
- Principle of RCE Neural Network

Revision

- Recogniton of Colors
- Recognition of Curves
- Recognition of Patterns
- Practices in MATLAB



Practice 1: Offline Face Classification

```
clear all; clc; %% clear workspace and memory
[file, path] = uigetfile('*.*', 'Select an image');
imagefile = strcat(path, file);
rgbimage = imread(imagefile);
subplot(3,1,1) ;  imshow(rgbimage);
grayimage = rgb2gray(rgbimage);
subplot(3,1,2);   imshow(grayimage);

faceDetector = vision.CascadeObjectDetector();
faceDetector.MergeThreshold = 5 ; %% you can change this value
boundingboxes = step(faceDetector, grayimage);
faceimage = insertObjectAnnotation(rgbimage, ...
                                   'Rectangle', ...
                                    boundingboxes, ...
                                   'Face');
subplot(3,1,3);
imshow(faceimage);
```

Demonstration video ...

Practice 2: Real-time Face Detection

```
clear all; close all; clc;
cam = webcam(2);
rgbimage = snapshot(cam) ;
imshow(imresize(rgbimage, 0.2)) ;
faceDetector = vision.CascadeObjectDetector();
faceDetector.MergeThreshold = 5;
yes = 1 ; counter = 100; % counter change could be changed
if yes == 1
    while yes
        rgbimage = snapshot(cam);
        rgbimage = imresize(rgbimage, 0.2);
        grayimage = rgb2gray(rgbimage);
        boundingboxes = step(faceDetector, grayimage);
        faceimage = insertObjectAnnotation(rgbimage, ...
                                           'Rectangle', ...
                                           boundingboxes, ...
                                           'Face');
        imshow(faceimage);
        counter = counter - 1;
        if counter == 0
            yes = input('Continue (1/0)');
        end
        couter = 100; % each time, perform 100 loops.
    end
end
clear cam ; clear all;
```

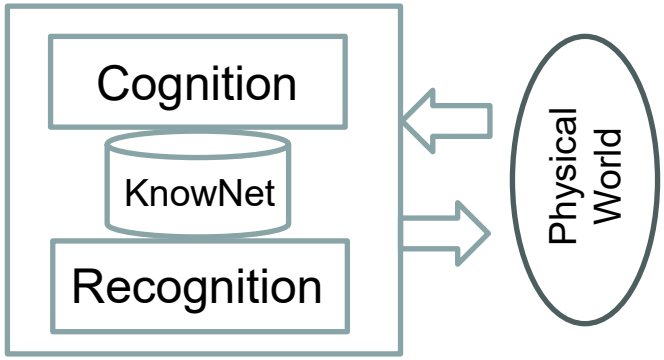
Demonstration video ...

Summary of Lecture 4

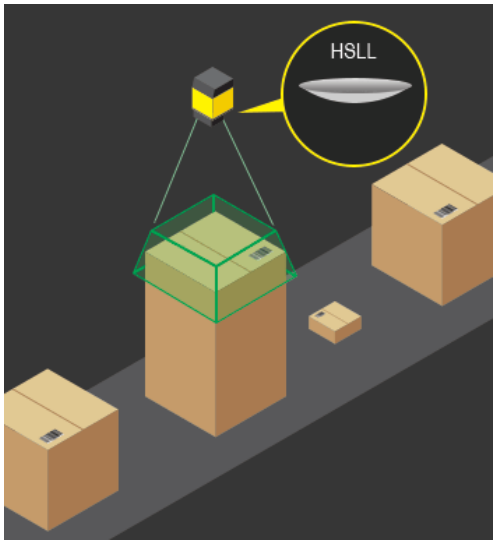
- Principle of Recognition (Understanding)
- Principle of Artificial Neural Network
- Principle of RCE Neural Network

Revision

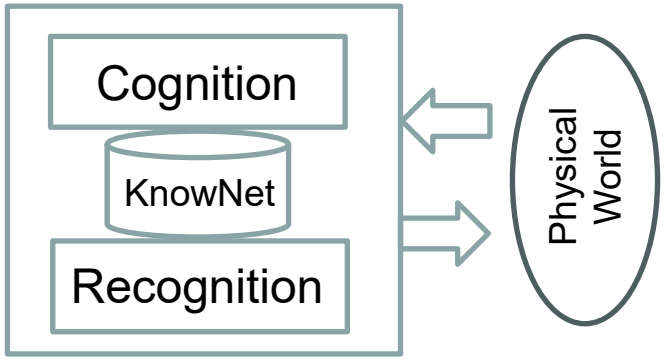
- Recogniton of Colors
- Recognition of Curves
- Recognition of Patterns
- Practices in MATLAB



Outline of Module 4



- Lecture 1:
  - Geometry-Driven Computation
- Lecture 2:
  - Fuzziness-Driven Computation
- Lecture 3:
  - Cognition-Driven Computation
- Lecture 4:
  - Recognition-Driven Computation
- Lecture 5:
  - Computation Using Monocular Vision
- Lecture 6:
  - Computation Using Binocular Vision



NANYANG TECHNOLOGICAL UNIVERSITY

School of Mechanical & Aerospace Engineering

Design, Machine, Control, Intelligence

Lecture 5 of Module 4

AI 3.0

MA4829 Machine Intelligence

Computation Using Monocular Vision

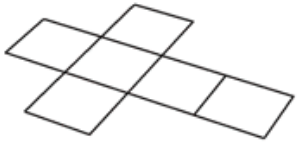
XIE Ming, PhD (France)

http://personal.ntu.edu.sg/mmxie

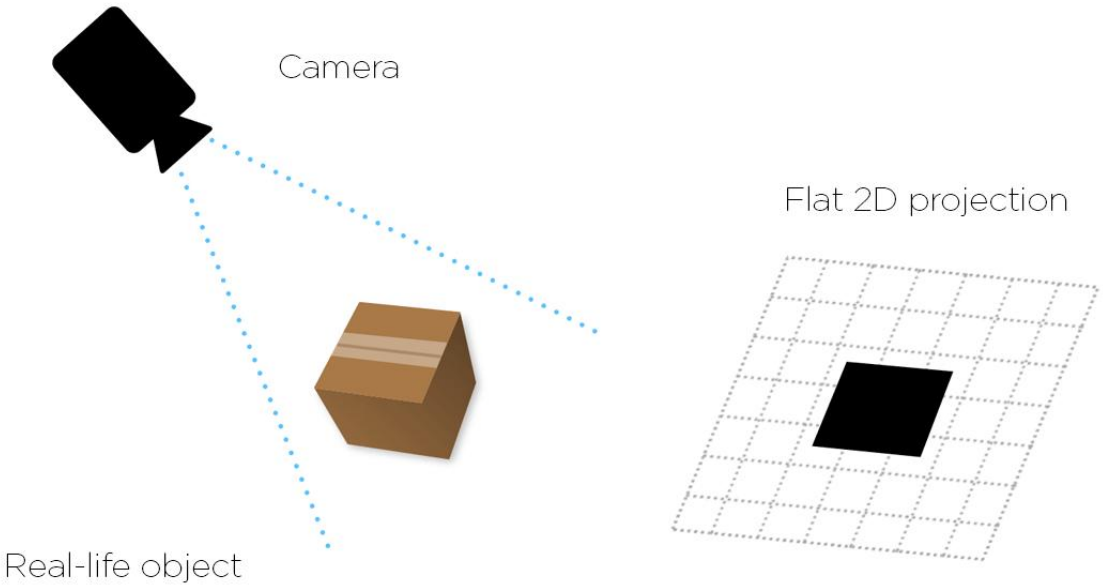
"We are living inside an ocean of signals"

What is a system with intelligence inside?

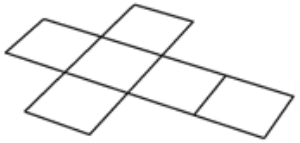
Outline of Lecture 5



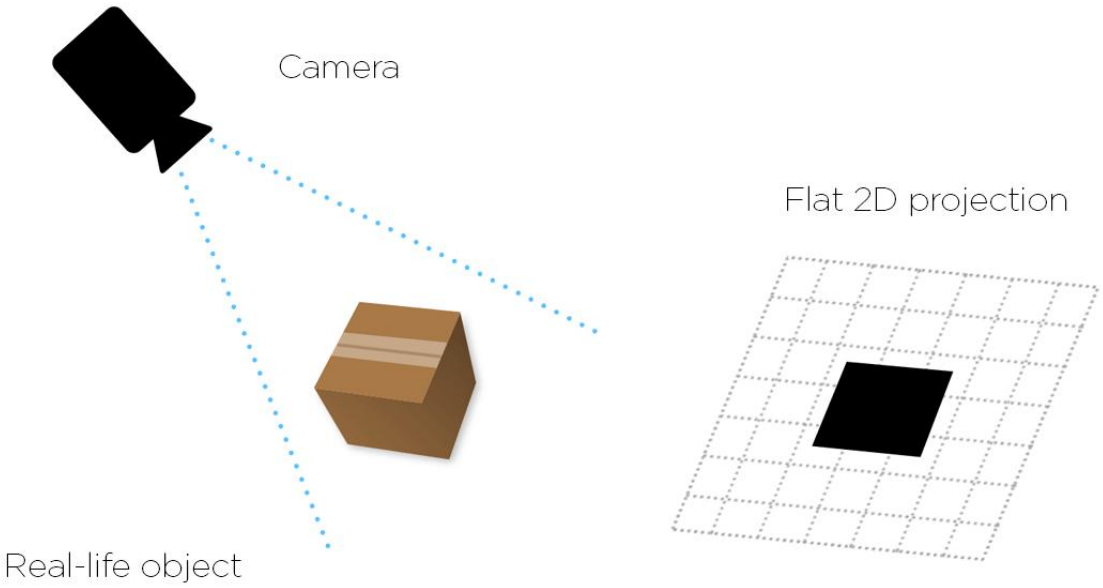
- Concept of Coordinate Transformation
- Transformation from Scene to Camera Space
- Transformation from Camera Space to Image Space
- Equation of Monocular Vision
- Calibration of Monocular Vision



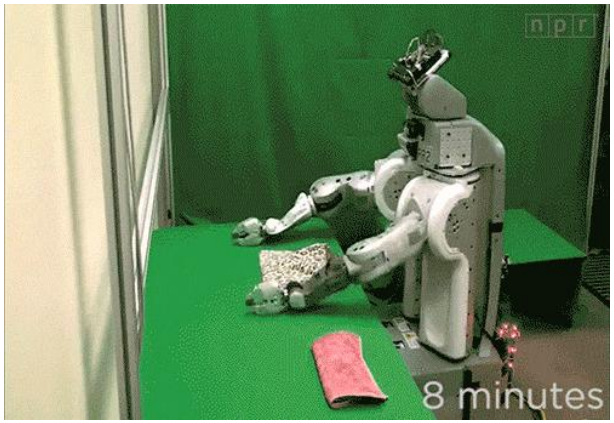
Outline of Lecture 5

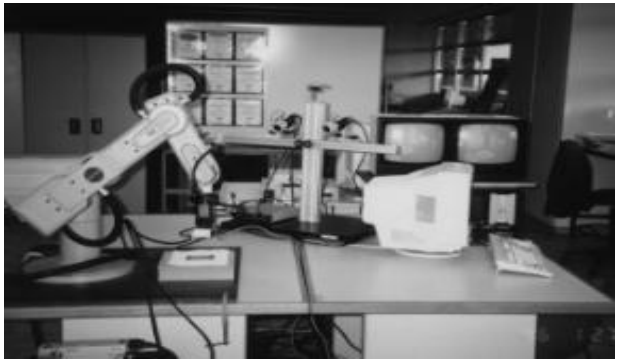


- Concept of Coordinate Transformation
- Transformation from Scene to Camera Space
- Transformation from Camera Space to Image Space
- Equation of Monocular Vision
- Calibration of Monocular Vision



Vision-Guided Manipulation ...

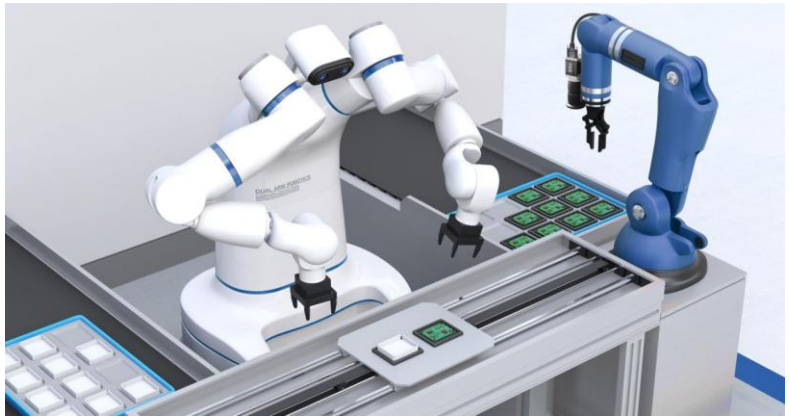


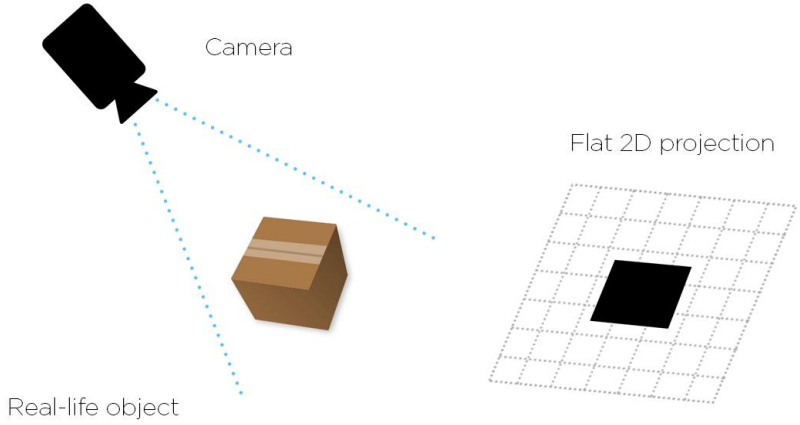


Vision-Guided Mobile Manipulation …

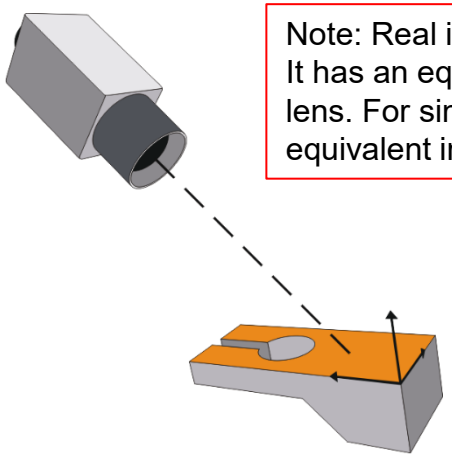
A vision system itself is a mechanical system …

- A mechanical system consists of many rigid elements, devices or components.
- Hence, it is an important topic to study the spatial relationship among rigid elements, devices and components.
- As a result, each element, device or component will be assigned a local coordinate system.

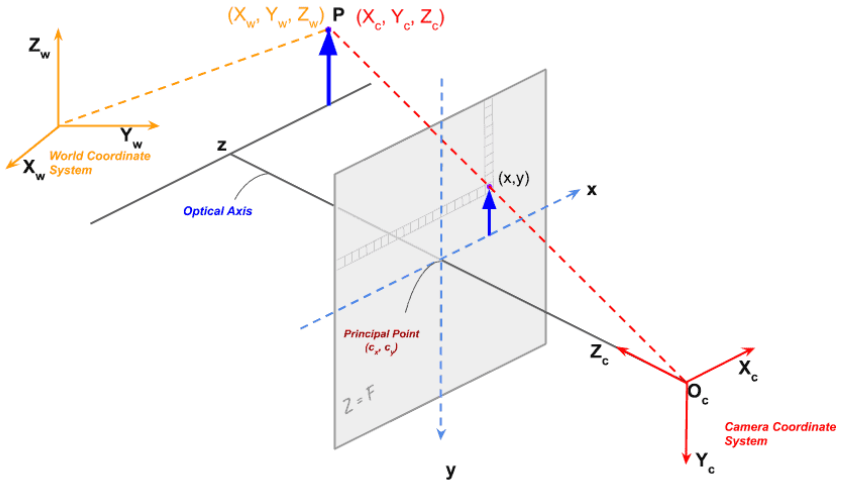


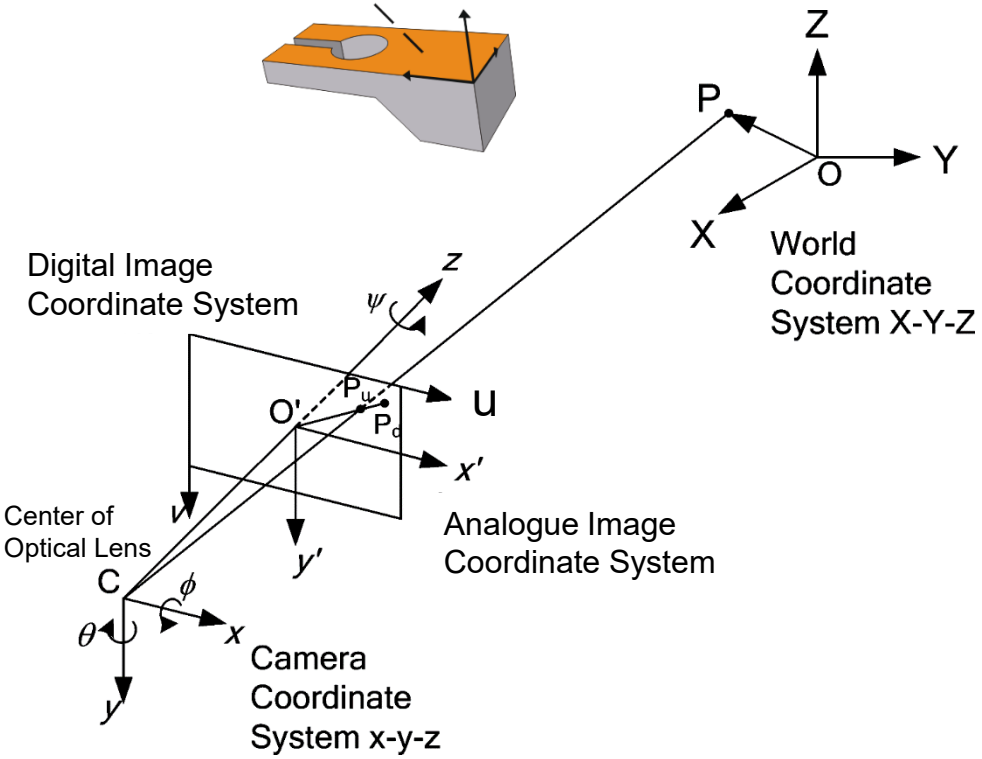


Example of Coordinate Systems Assigned to a Camera and its Images Inside …



Note: Real image is inside the camera. It has an equivalent image in front of the lens. For simplicity, we use a real image's equivalent image.





What is the scenario of coordinate transformations?

Basics of Homogeneous Transformation

- Pose = Position + Orientation

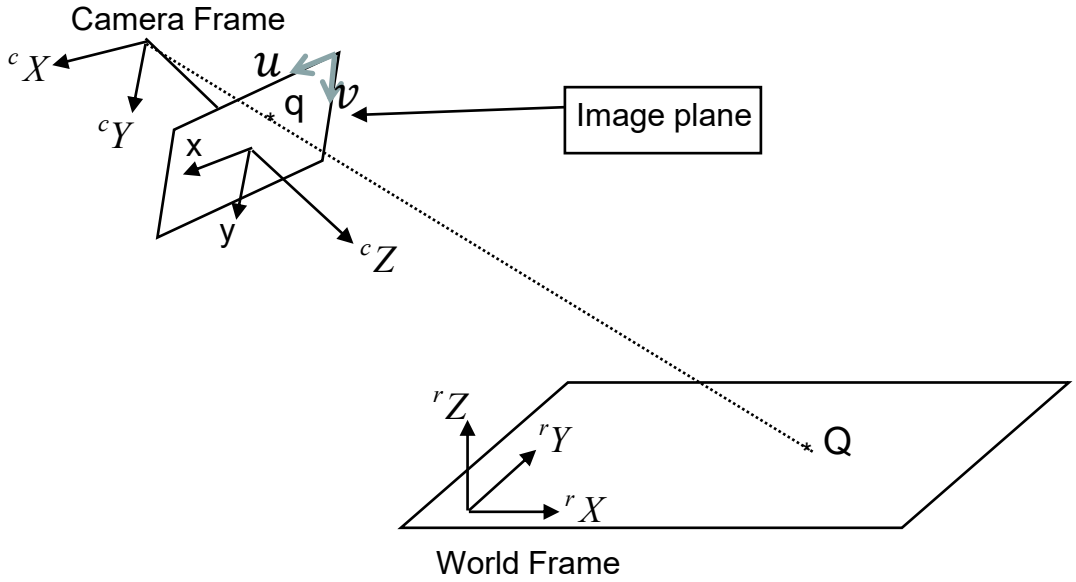
$$H_{camera} = \begin{bmatrix} R_{camera} & T_{camera} \\ 0 \quad 0 \quad 0 & 1 \end{bmatrix}$$

This is the pose of the camera frame with respect to the world frame.

$$H_{world} = \begin{bmatrix} R_{camera}^{-1} & -R_{camera}^{-1} \times T_{camera} \\ 0 \quad 0 \quad 0 & 1 \end{bmatrix}$$

(This is the pose of the world frame with respect to the camera frame)

(Monocular Vision)



$$H_{camera} \times H_{world} = I_{4\times4}$$

$$H_{camera} = H_{world}^{-1}$$

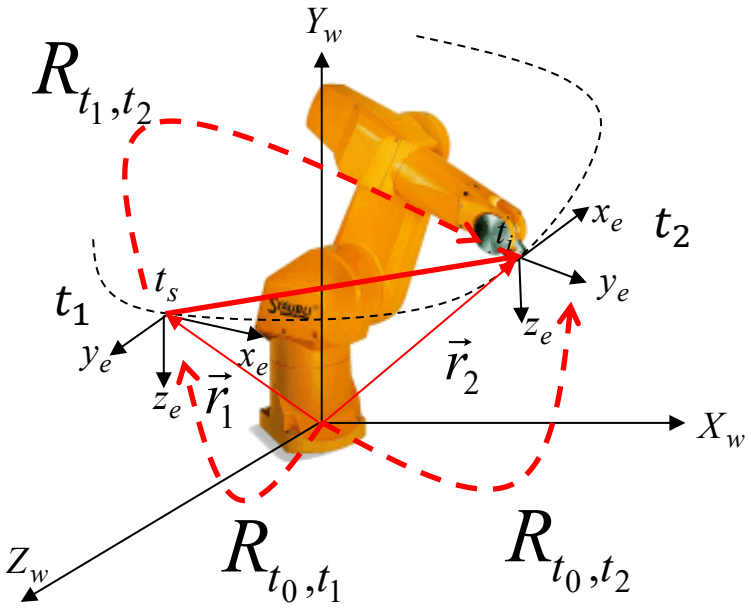
Basics of Homogeneous Transformation

This is the pose of the tooltip frame with respect to the world frame

$$H_{tooltip} = \begin{bmatrix} R_{tooltip} & T_{tooltip} \\ 0 \quad 0 \quad 0 & 1 \end{bmatrix}$$

This is the pose of the world frame with respect to the tooltip frame

$$H_{world} = \begin{bmatrix} R_{tooltip}^{-1} & -R_{tooltip}^{-1} \times T_{tooltip} \\ 0 \quad 0 \quad 0 & 1 \end{bmatrix}$$



(Robot Arm)

$$H_{tooltip} \times H_{world} = I_{4\times4}$$

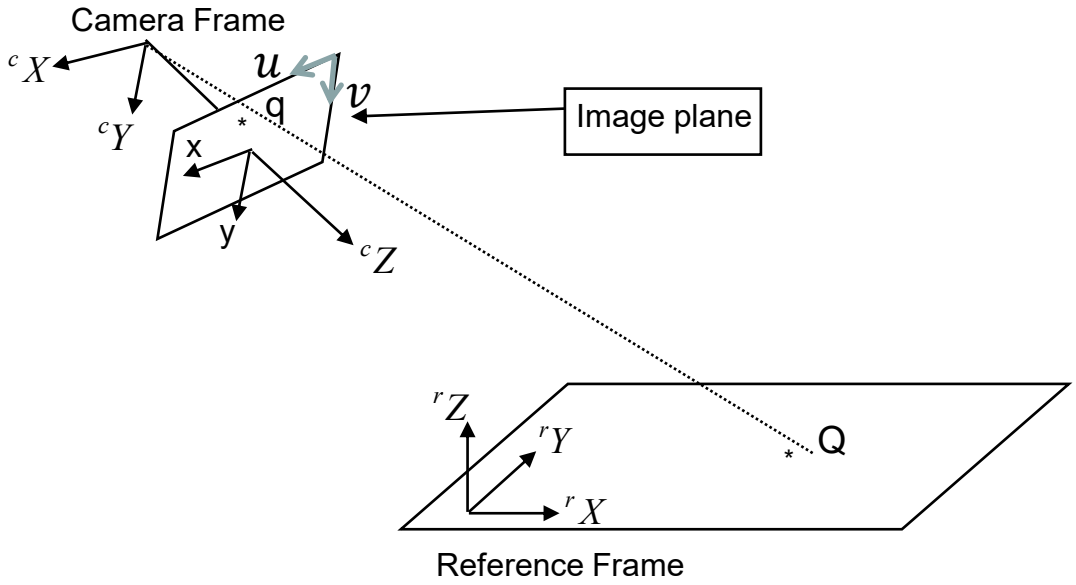
$$H_{tooltip} = H_{world}^{-1}$$

Transformation of Coordinates in 3D Space

- The coordinates in the reference frame can be transformed into the coordinates in the camera frame.

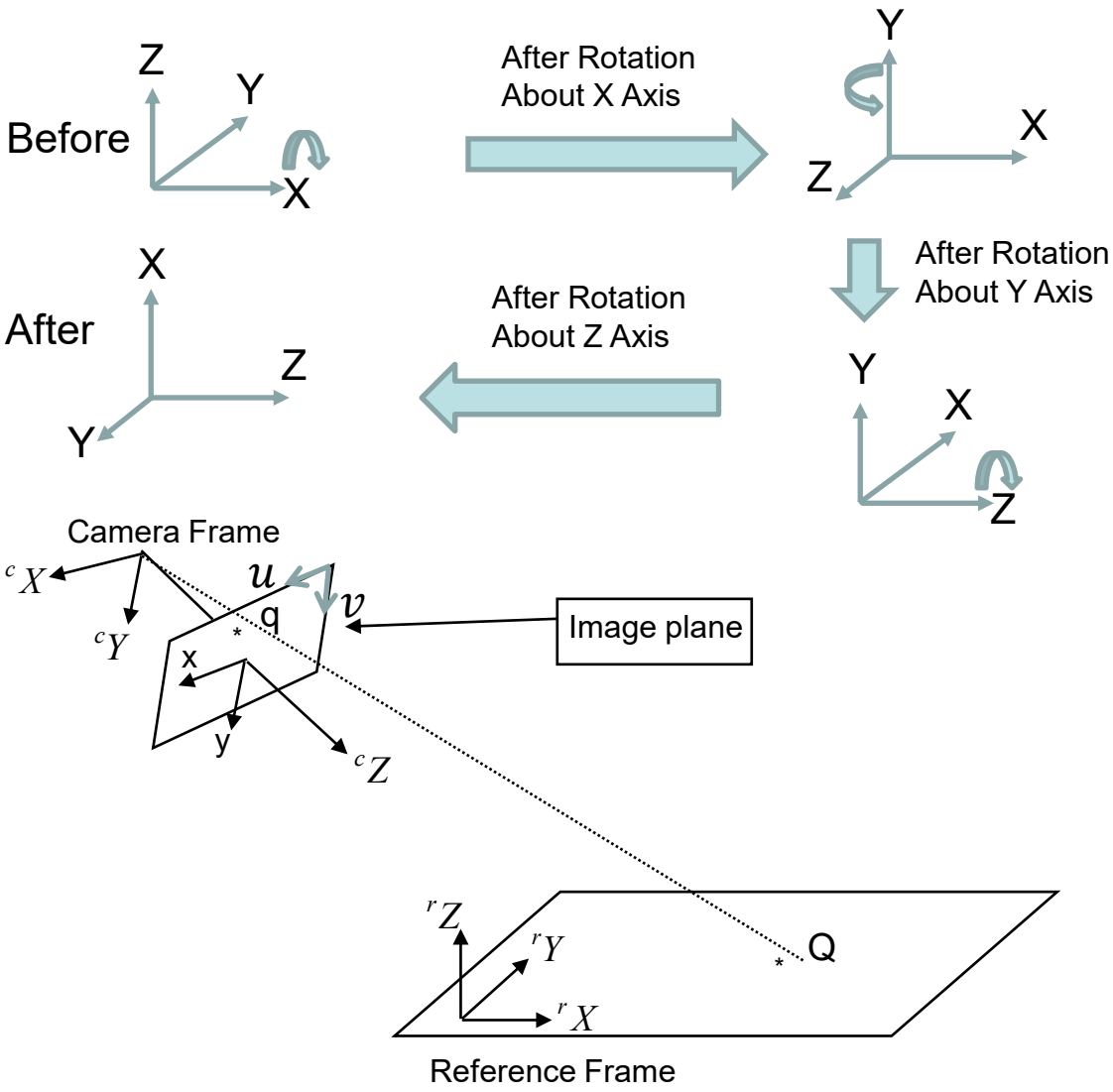
This is the pose of the reference frame with respect to the camera frame

$$ {}^{c}H_{r} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} $$



$$ \begin{bmatrix} {}^{c}X \\ {}^{c}Y \\ {}^{c}Z \\ 1 \end{bmatrix} = {}^{c}H_{r} \bullet \begin{bmatrix} {}^{r}X \\ {}^{r}Y \\ {}^{r}Z \\ 1 \end{bmatrix} $$

Practices with Rotational Transformation



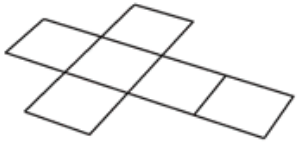

```
1
2 -   ax = 90*pi/180;
3 -   rx = [1   0          0 ;
4           0  cos(ax)   -sin(ax);
5           0  sin(ax)    cos(ax)];
6
7 -   ay = 90*pi/180;
8 -   ry = [cos(ay)   0    sin(ay);
9             0       1    0 ;
10           -sin(ay) 0    cos(ay)];
11
12 -  az = 90*pi/180;
13 -  rz = [cos(az)   -sin(az)   0;
14          sin(az)    cos(az)   0;
15          0          0         1];
16
17 -  r = rx*ry*rz
18
19
```

Command Window

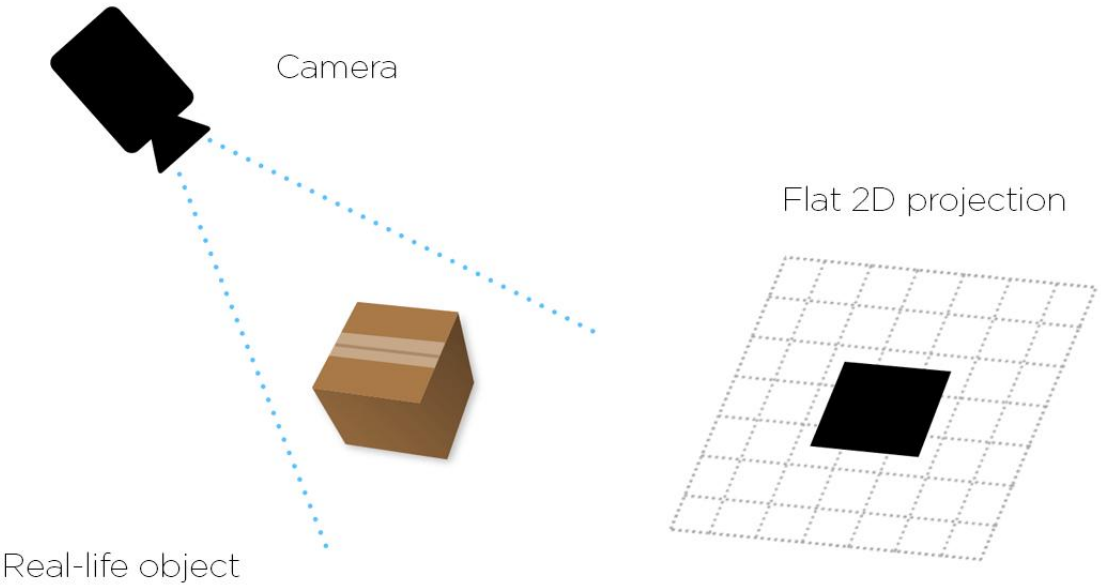
New to MATLAB? See resources for Getting Started.

```
r =

    0.0000   -0.0000    1.0000
    0.0000   -1.0000   -0.0000
    1.0000    0.0000    0.0000
```

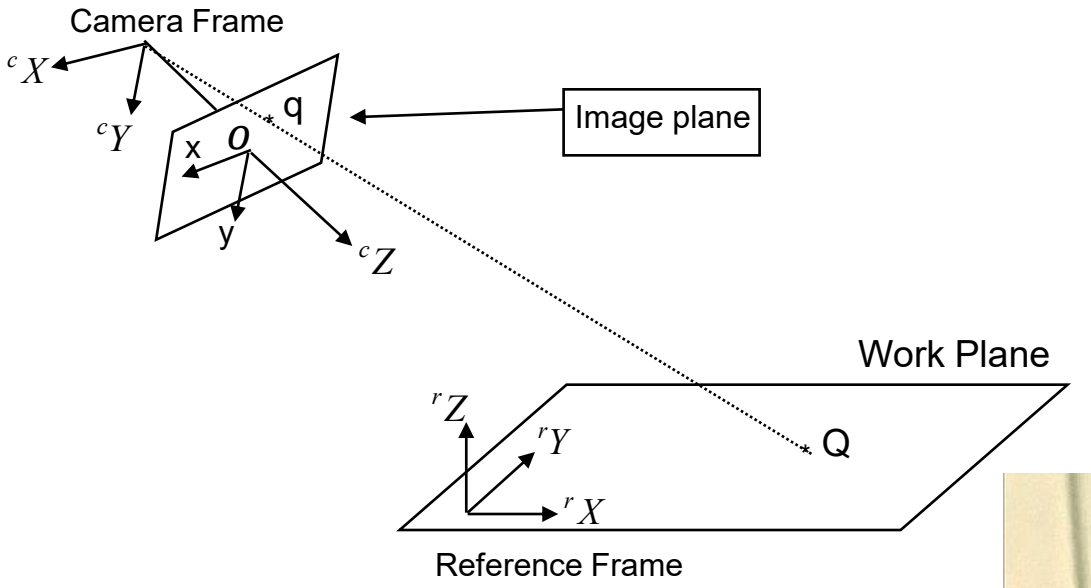
Outline of Lecture 5

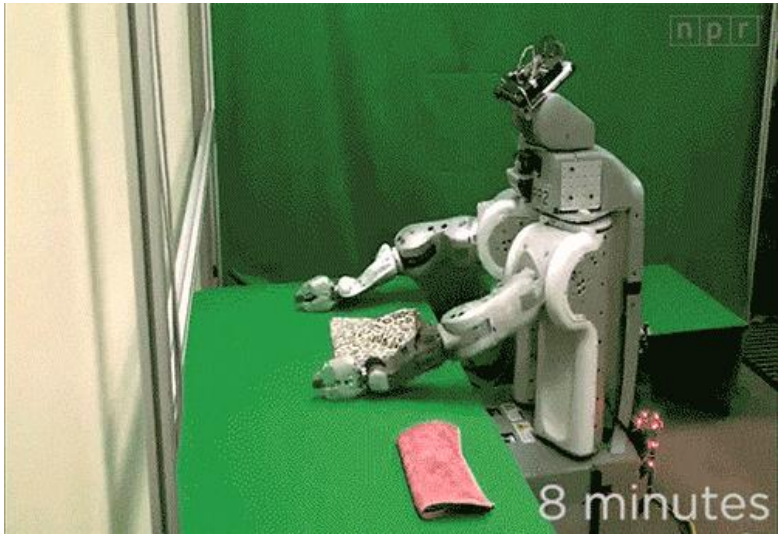


- Concept of Coordinate Transformation
- Transformation from Scene to Camera Space
- Transformation from Camera Space to Image Space
- Equation of Monocular Vision
- Calibration of Monocular Vision



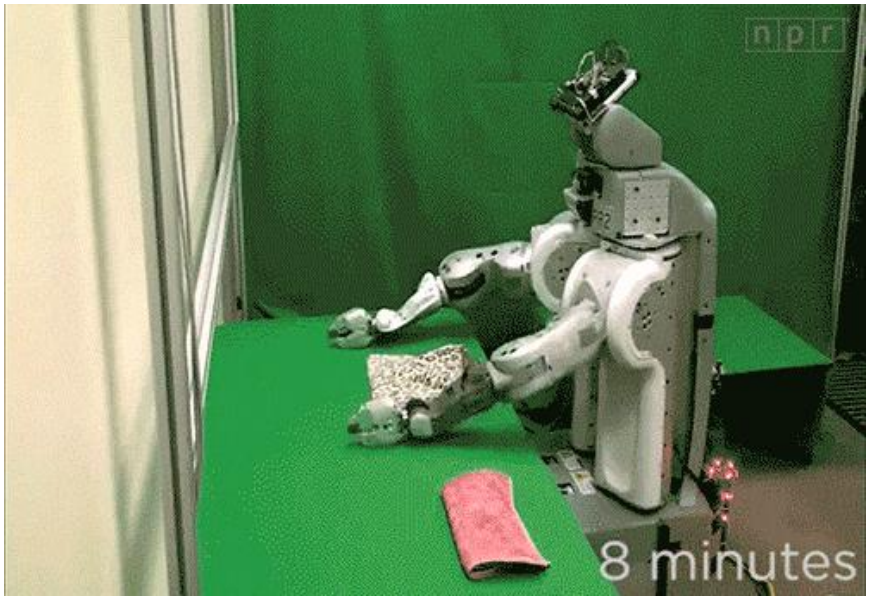
Setup of Coordinate Systems …

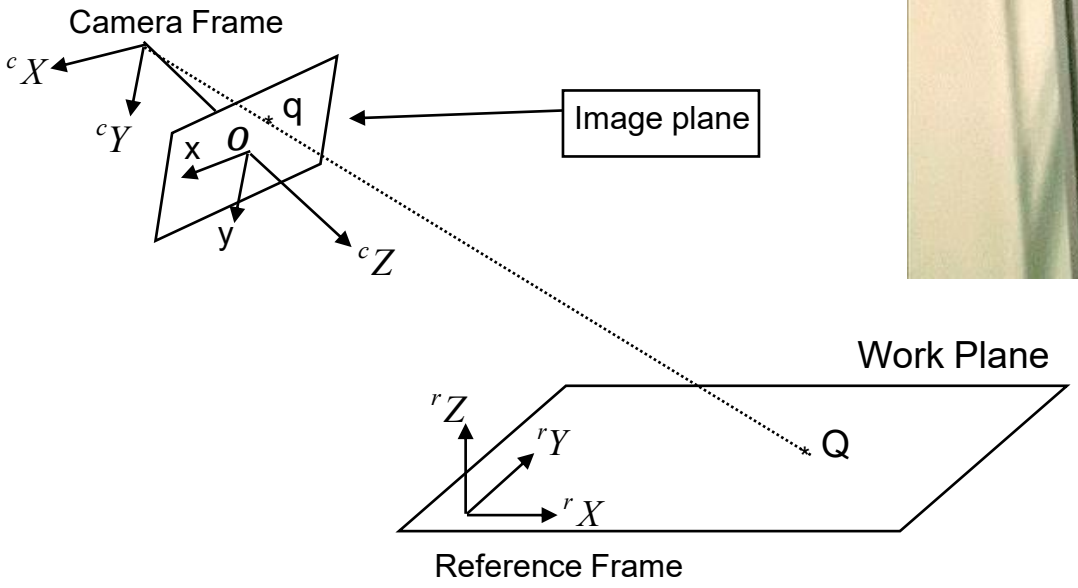




Representation of Scene with Respect to Camera's Coordinate System …

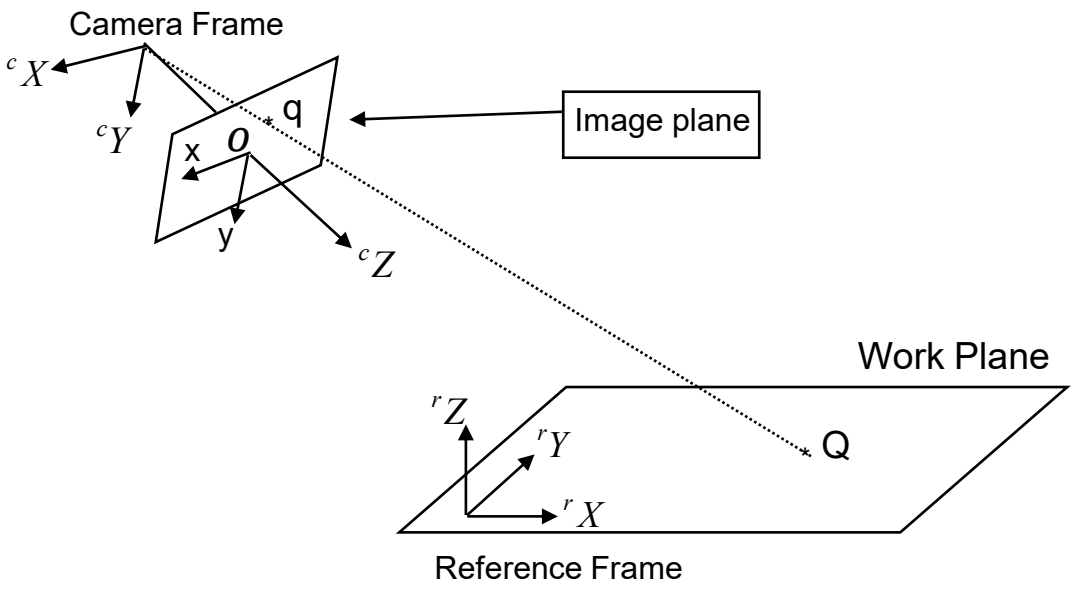
1. Position
2. Orientation





$$ {}^{c}H_{r} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

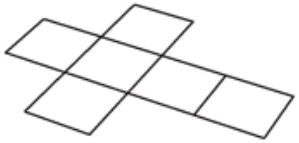
Transformation from Scene to Camera Space …



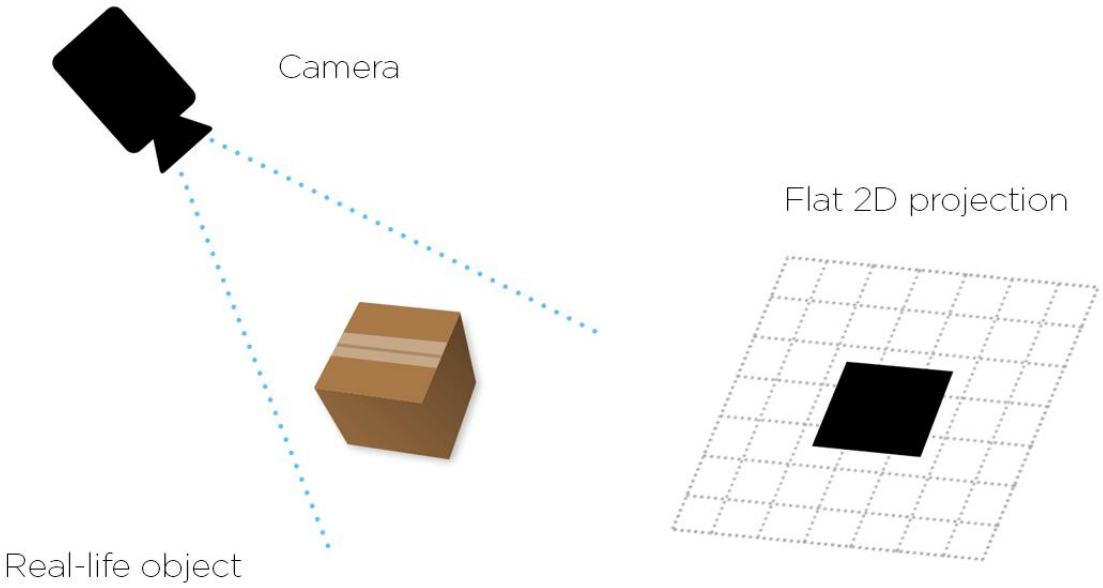
$$^cH_r = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} ^cX \\ ^cY \\ ^cZ \\ 1 \end{bmatrix} = {}^cH_r \bullet \begin{bmatrix} ^rX \\ ^rY \\ ^rZ \\ 1 \end{bmatrix}$$

Outline of Lecture 5



- Concept of Coordinate Transformation
- Transformation from Scene to Camera Space
- Transformation from Camera Space to Image Space
- Equation of Monocular Vision
- Calibration of Monocular Vision



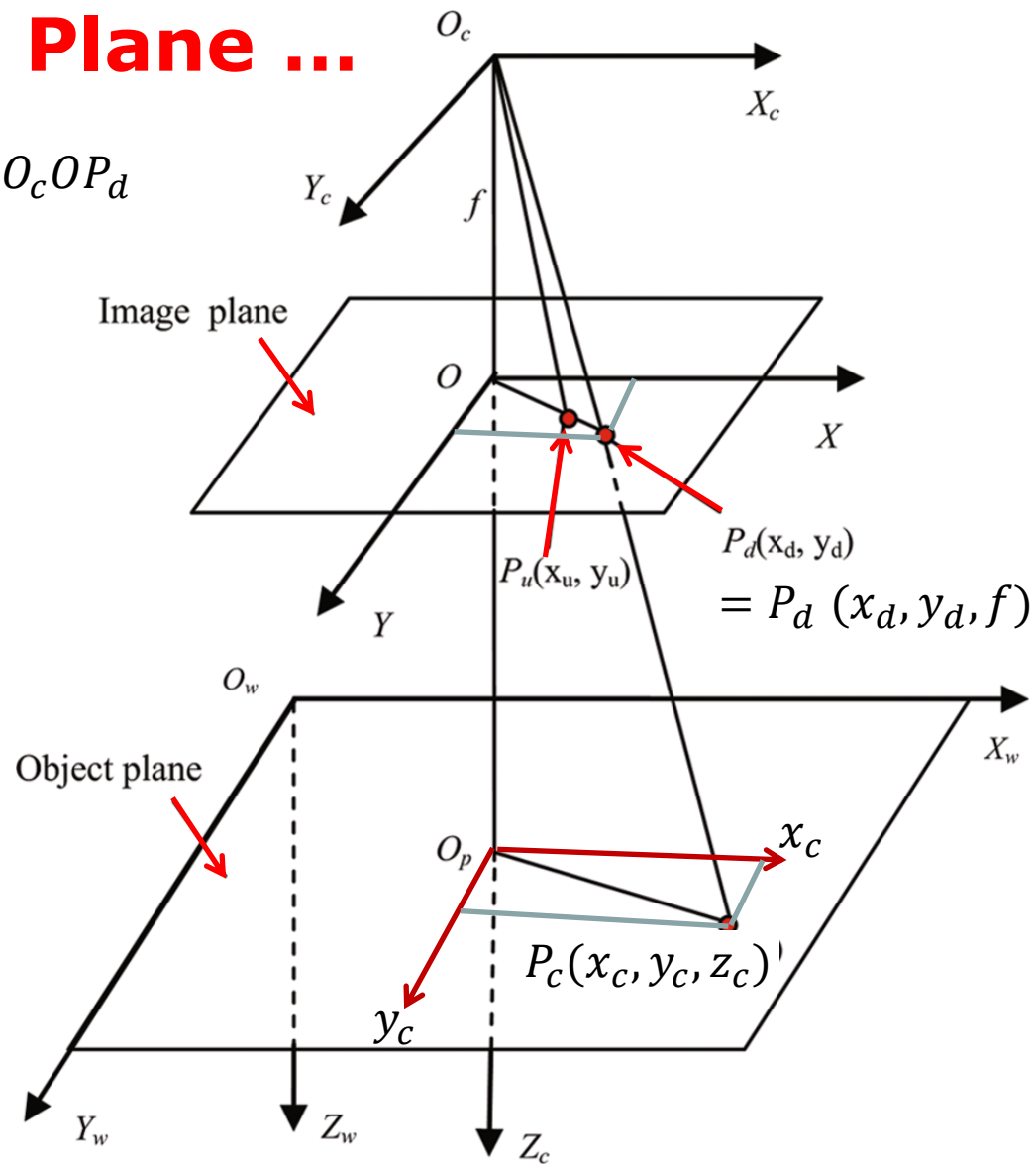
Projection to Image Plane …

Triangle $O_c O_p P_c$ is similar to Triangle $O_c O P_d$

$$\frac{O_p O_c}{O O_c} = \frac{P_c O_p}{P_d O} = \frac{z_c}{f} = \frac{x_c}{x_d} = \frac{y_c}{y_d}$$

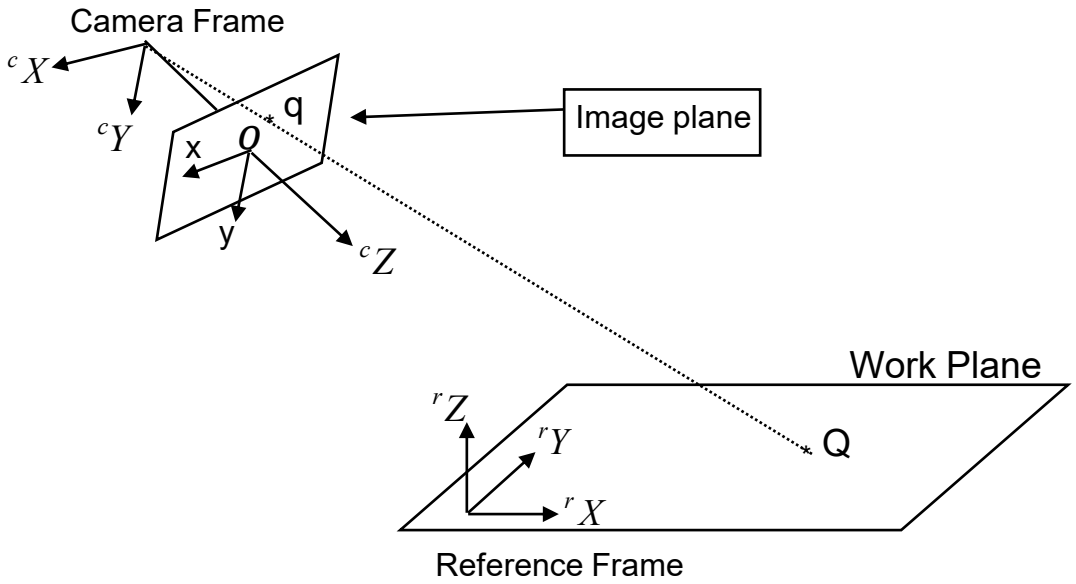
$$\frac{z_c}{f} = \frac{x_c}{x_d} \Rightarrow x_d = f\frac{x_c}{z_c}$$

$$\frac{z_c}{f} = \frac{y_c}{y_d} \Rightarrow y_d = f\frac{y_c}{z_c}$$



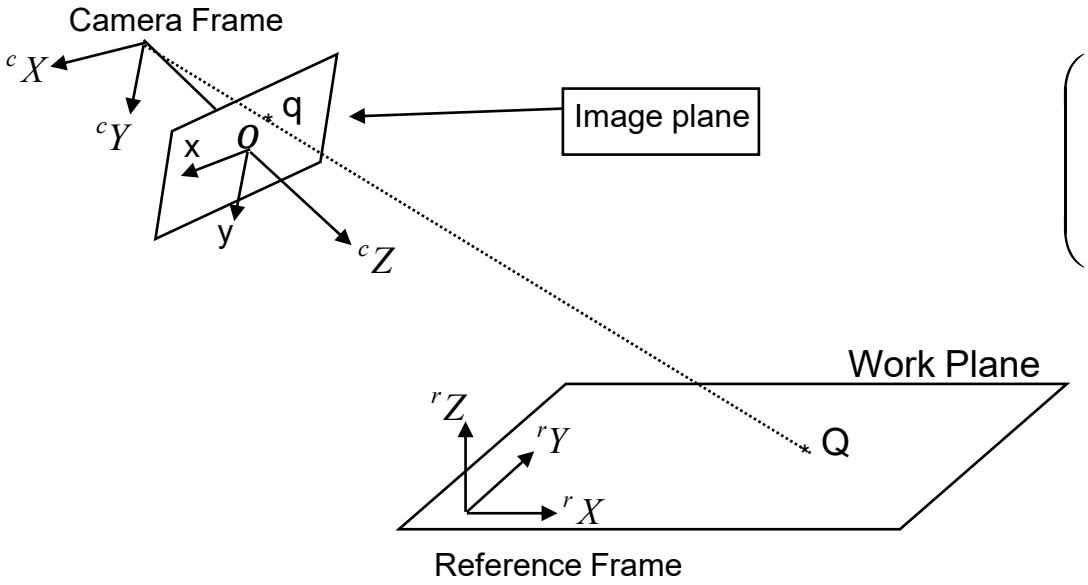
Projection to Image Plane …

$$x = f \bullet \frac{{}^{c}X}{{}^{c}Z} \qquad \text{and} \qquad y = f \bullet \frac{{}^{c}Y}{{}^{c}Z}$$



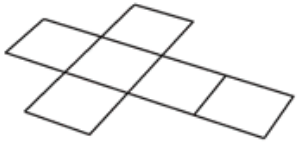
Matrix Equation of Projection ...

$$x = f \bullet \frac{{}^{c}X}{{}^{c}Z} \quad \text{and} \quad y = f \bullet \frac{{}^{c}Y}{{}^{c}Z}$$

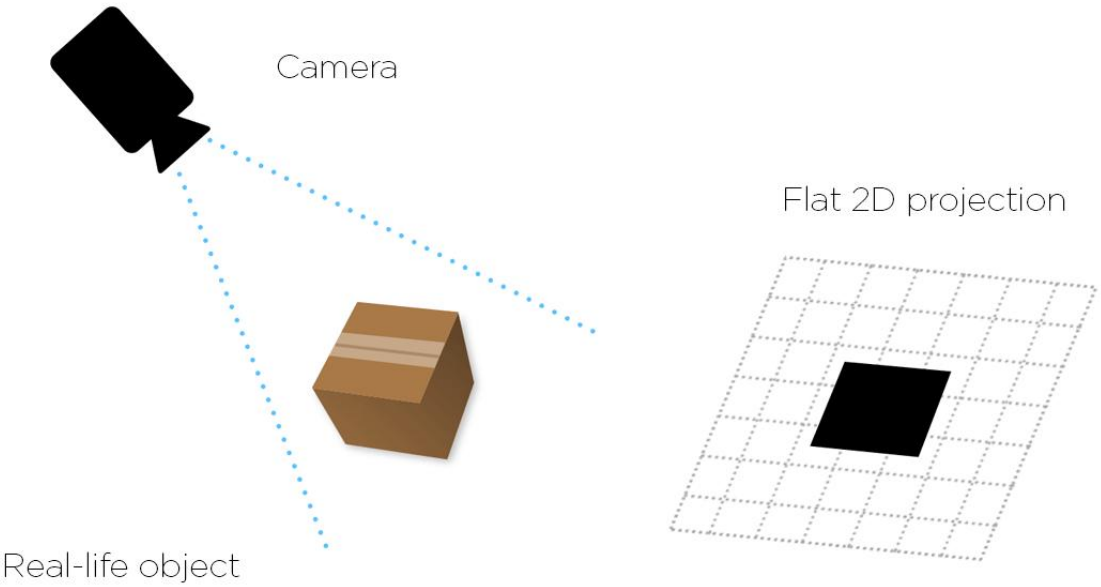


$$\begin{pmatrix} s \bullet x \\ s \bullet y \\ s \end{pmatrix} = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \bullet \begin{pmatrix} {}^{c}X \\ {}^{c}Y \\ {}^{c}Z \\ 1 \end{pmatrix}$$

Outline of Lecture 5



- Concept of Coordinate Transformation
- Transformation from Scene to Camera Space
- Transformation from Camera Space to Image Space
- Equation of Monocular Vision
- Calibration of Monocular Vision



Coordinate Transformation from Scene to Analogue Image …

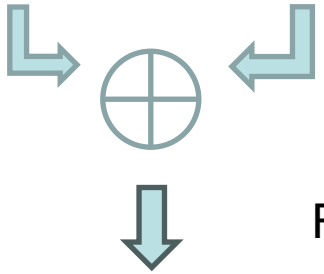
From Scene to Camera

$$\begin{bmatrix} {}^{c}X \\ {}^{c}Y \\ {}^{c}Z \\ 1 \end{bmatrix} = {}^{c}H_{r} \bullet \begin{bmatrix} {}^{r}X \\ {}^{r}Y \\ {}^{r}Z \\ 1 \end{bmatrix}$$

From Camera to Analogue Image

$$\begin{pmatrix} s \bullet x \\ s \bullet y \\ s \end{pmatrix} = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \bullet \begin{pmatrix} {}^{c}X \\ {}^{c}Y \\ {}^{c}Z \\ 1 \end{pmatrix}$$

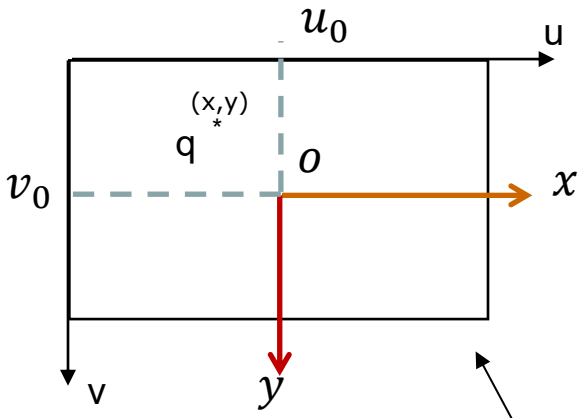
$$ {}^{c}H_{r} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



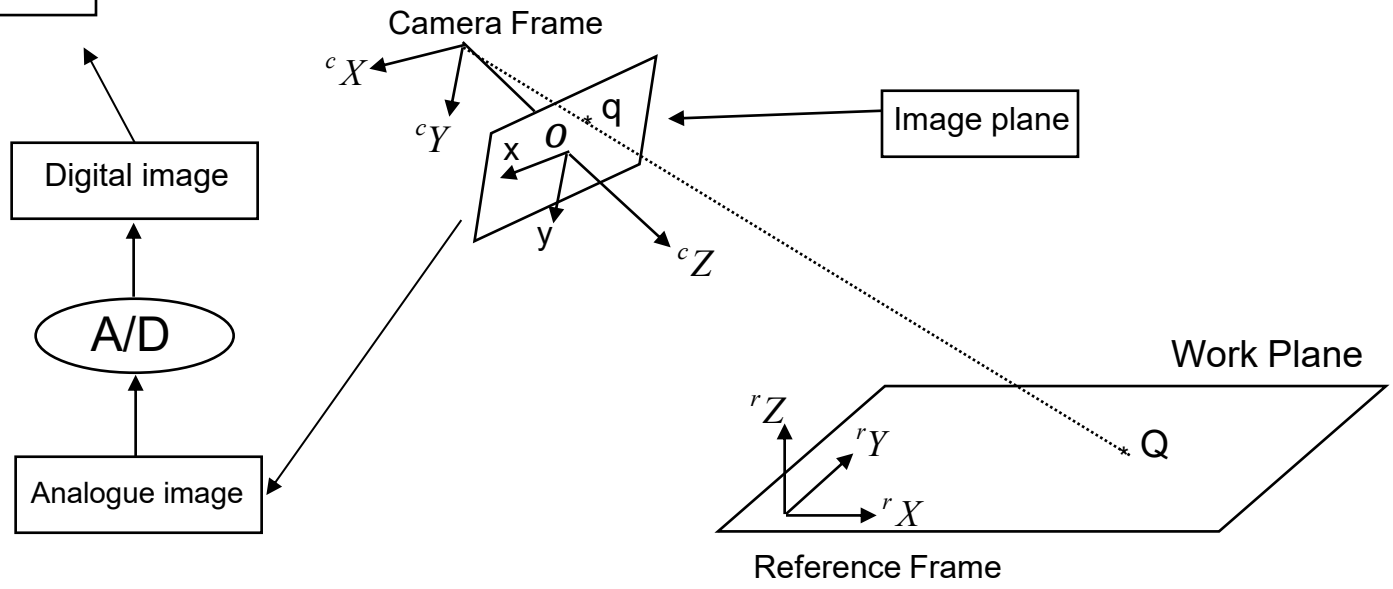
From Scene to Analogue Image

$$\begin{pmatrix} s \bullet x \\ s \bullet y \\ s \end{pmatrix} = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} {}^{r}X \\ {}^{r}Y \\ {}^{r}Z \\ 1 \end{pmatrix}$$

How to transform scene to digital image?

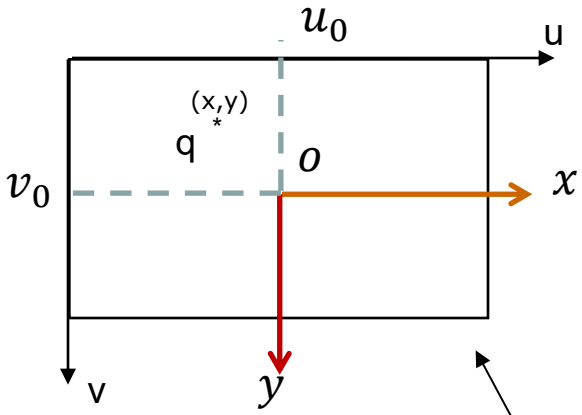


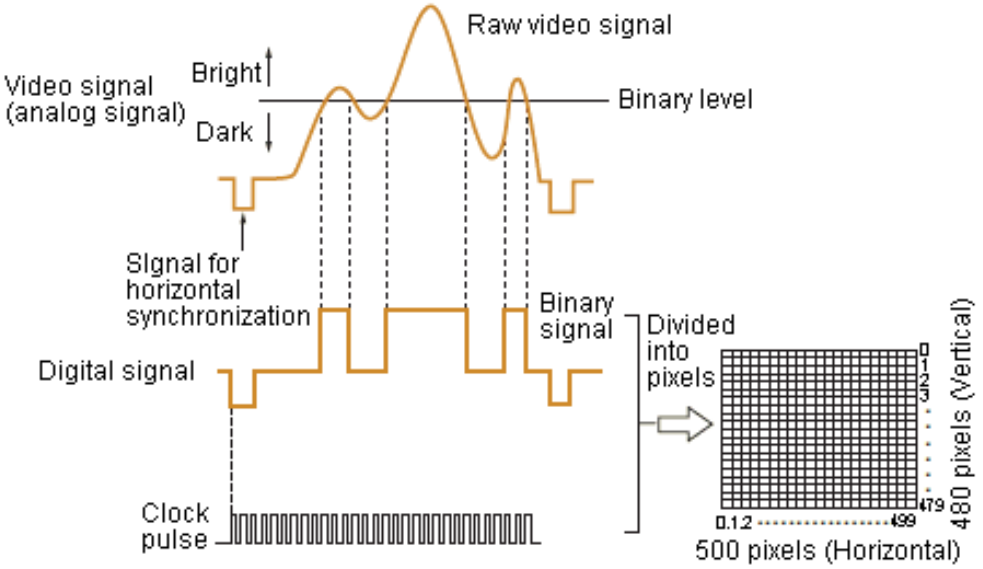
$$\begin{pmatrix} s \bullet x \\ s \bullet y \\ s \end{pmatrix} = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} {}^{r}X \\ {}^{r}Y \\ {}^{r}Z \\ 1 \end{pmatrix}$$

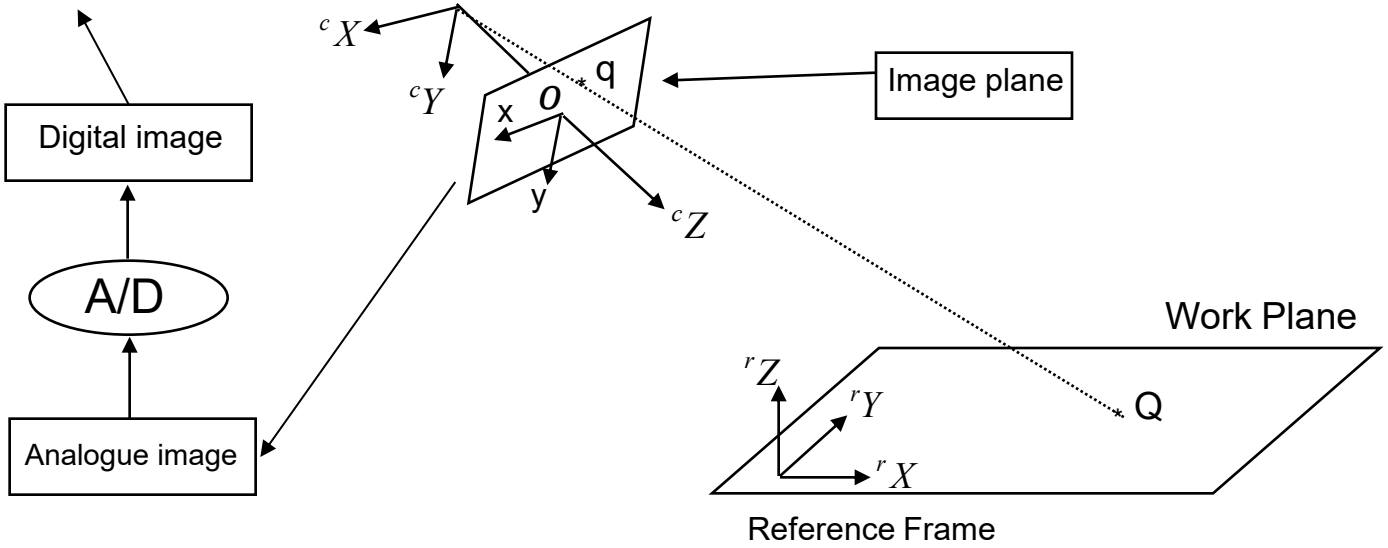


Revision ...

- The analogue images produced at the image plane are digitized into the corresponding digital images:





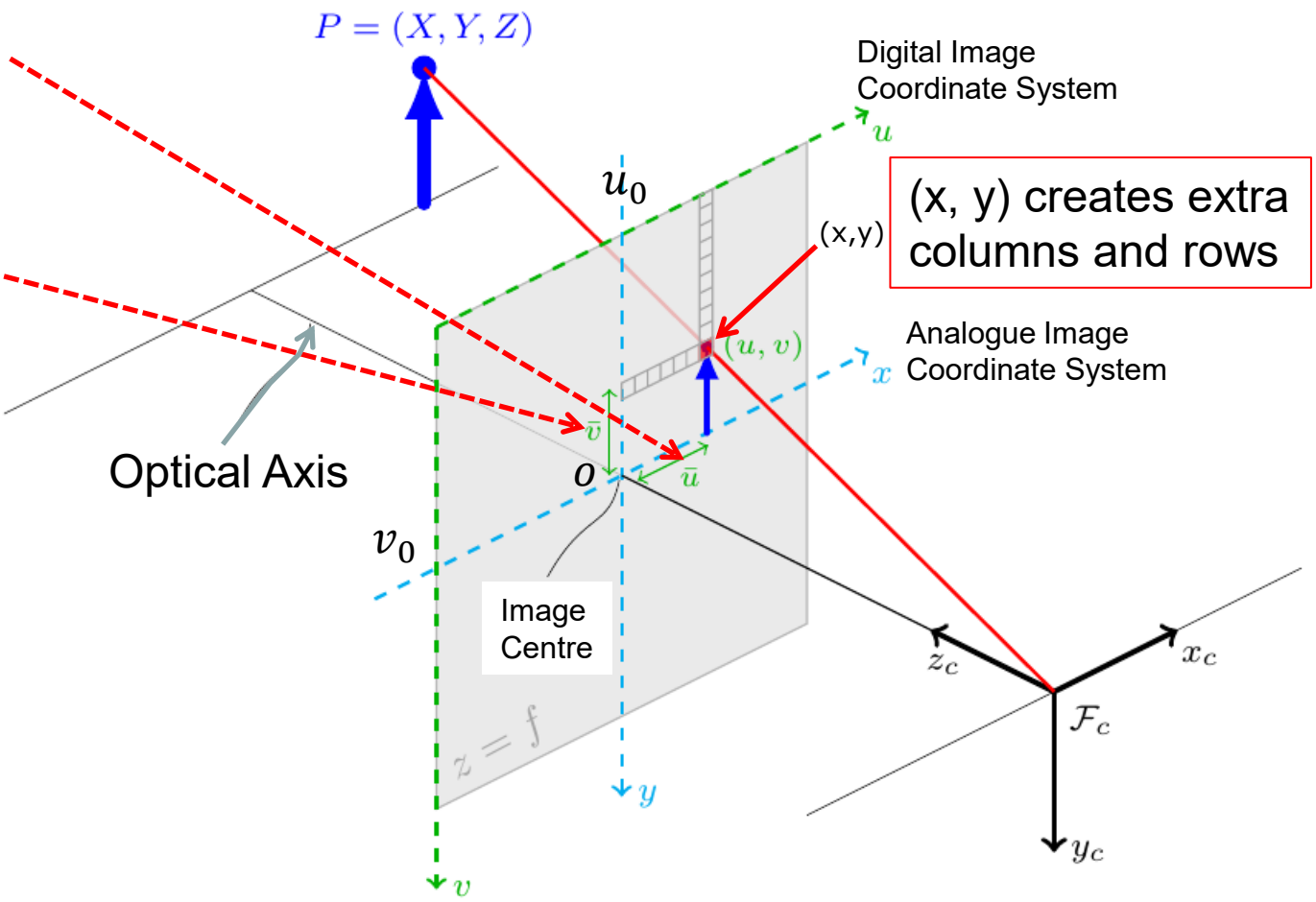


Coordinate Transformation between Analogue Image and Digital Image …

$$u = u_0 + \bar{u} = u_0 + \frac{x}{\Delta u}$$

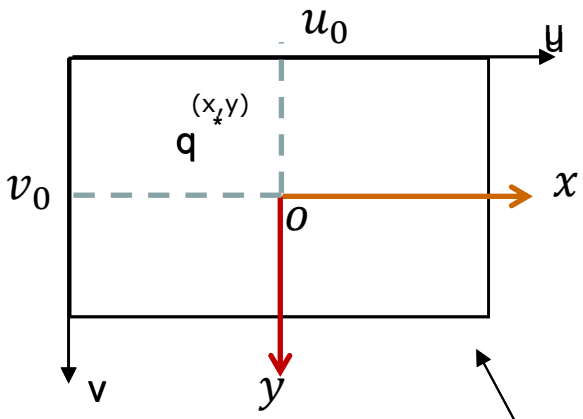
$$v = v_0 + \bar{v} = v_0 + \frac{y}{\Delta v}$$

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{1}{\Delta u} & 0 & u_0 \\ 0 & \frac{1}{\Delta v} & v_0 \\ 0 & 0 & 1 \end{pmatrix} \bullet \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$



Both sides can be multiplied by s

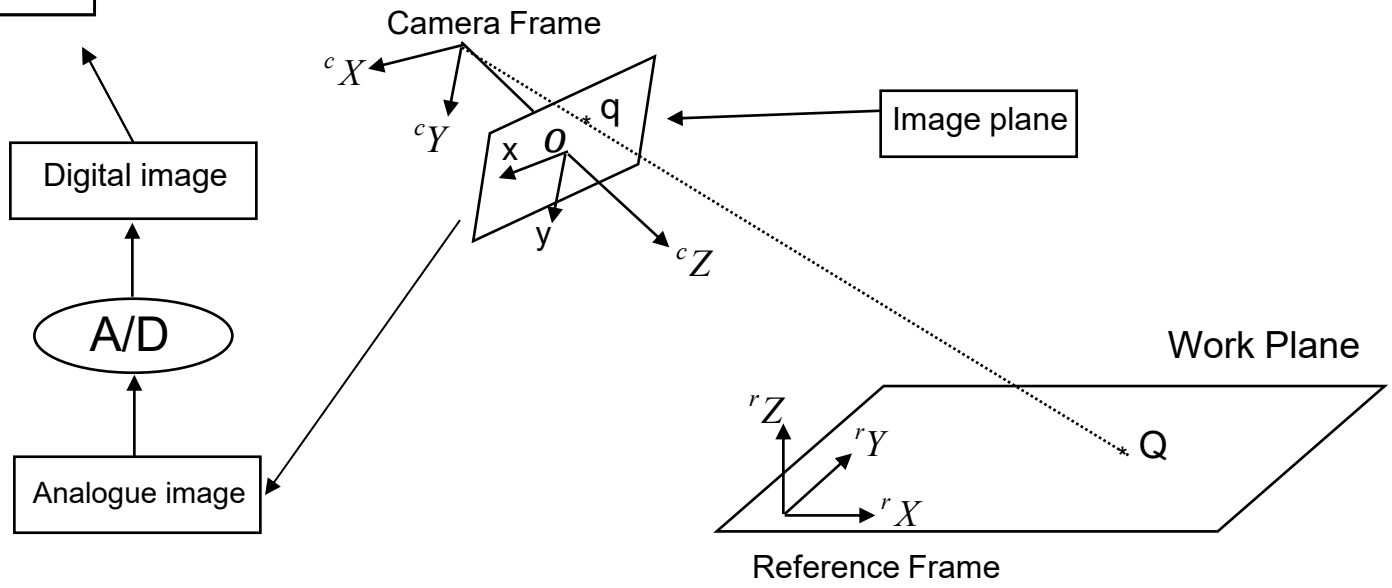
Coordinate Transformation from Analogue Image to Digital Image …



$$u = u_0 + \frac{x}{\Delta u}$$

$$v = v_0 + \frac{y}{\Delta v}$$

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{1}{\Delta u} & 0 & u_0 \\ 0 & \frac{1}{\Delta v} & v_0 \\ 0 & 0 & 1 \end{pmatrix} \bullet \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$



Transformation from Scene to Digital Image …

$$\begin{pmatrix} s \bullet x \\ s \bullet y \\ s \end{pmatrix} = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} {}^{r}X \\ {}^{r}Y \\ {}^{r}Z \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} s \bullet u \\ s \bullet v \\ s \end{pmatrix} = \begin{pmatrix} \frac{1}{\Delta u} & 0 & u_0 \\ 0 & \frac{1}{\Delta v} & v_0 \\ 0 & 0 & 1 \end{pmatrix} \bullet \begin{pmatrix} s \bullet x \\ s \bullet y \\ s \end{pmatrix}$$

$$\begin{pmatrix} s \bullet u \\ s \bullet v \\ s \end{pmatrix} = \begin{pmatrix} \frac{f}{\Delta u} & 0 & u_0 & 0 \\ 0 & \frac{f}{\Delta v} & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} {}^{r}X \\ {}^{r}Y \\ {}^{r}Z \\ 1 \end{pmatrix}$$

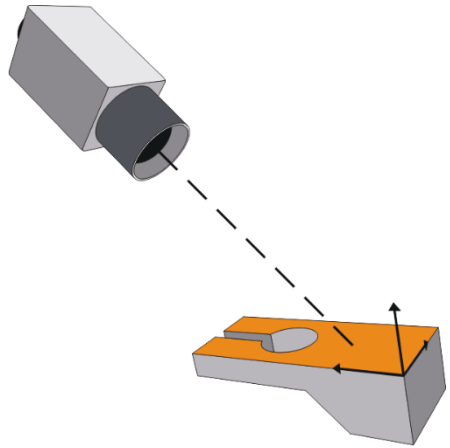
Equation of Camera's Forward Projection …

$$\begin{pmatrix} s \bullet u \\ s \bullet v \\ s \end{pmatrix} = \begin{pmatrix} \frac{f}{\Delta u} & 0 & u_0 & 0 \\ 0 & \frac{f}{\Delta v} & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} {}^{r}X \\ {}^{r}Y \\ {}^{r}Z \\ 1 \end{pmatrix}$$

Camera's Forward Projection Matrix

$$\begin{pmatrix} s \bullet u \\ s \bullet v \\ s \end{pmatrix} = C_{3\times 4} \bullet \begin{pmatrix} {}^{r}X \\ {}^{r}Y \\ {}^{r}Z \\ 1 \end{pmatrix}$$

Equation of Camera's Forward Projection



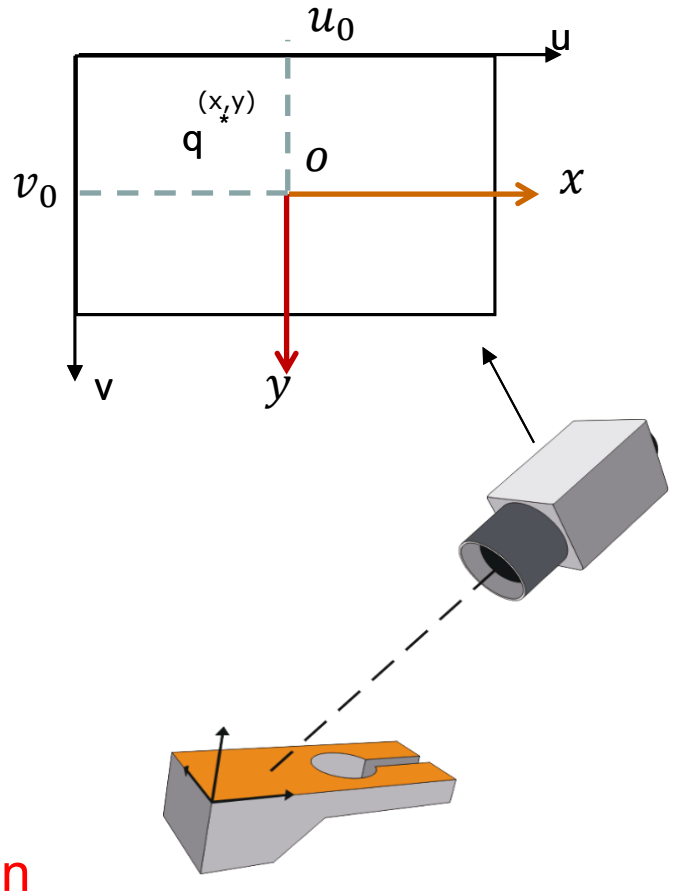
Equation of Monocular Vision's Forward Projection …

- If we let Z coordinate to be zero, then we obtain the equation of monocular vision's forward projection:

$$\begin{pmatrix} s \bullet u \\ s \bullet v \\ s \end{pmatrix} = C_{3\times4} \bullet \begin{pmatrix} {}^{r}X \\ {}^{r}Y \\ 0 \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} s \bullet u \\ s \bullet v \\ s \end{pmatrix} = \mathrm{D}_{3\times3} \bullet \begin{pmatrix} {}^{r}X \\ {}^{r}Y \\ 1 \end{pmatrix}$$

Equation of Monocular Vision's Forward Projection



Equation of Monocular Vision's Inverse Projection …

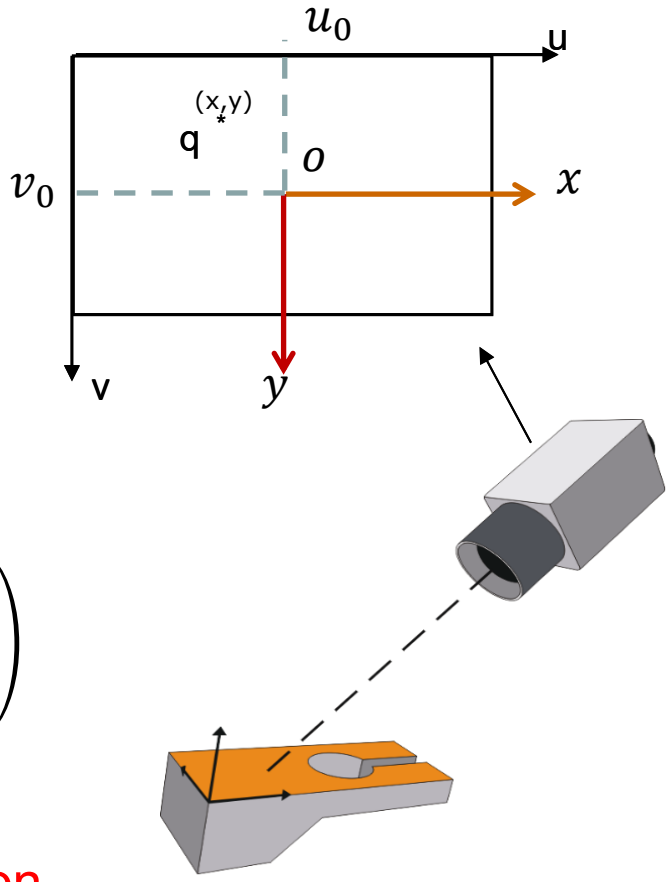
- By inverting matrix D, then we obtain the equation of monocular vision's inverse projection:

$$\begin{pmatrix} s \bullet u \\ s \bullet v \\ s \end{pmatrix} = \mathrm{D}_{3\times3} \bullet \begin{pmatrix} {}^{r}X \\ {}^{r}Y \\ 1 \end{pmatrix}$$

$$M = D^{-1}$$

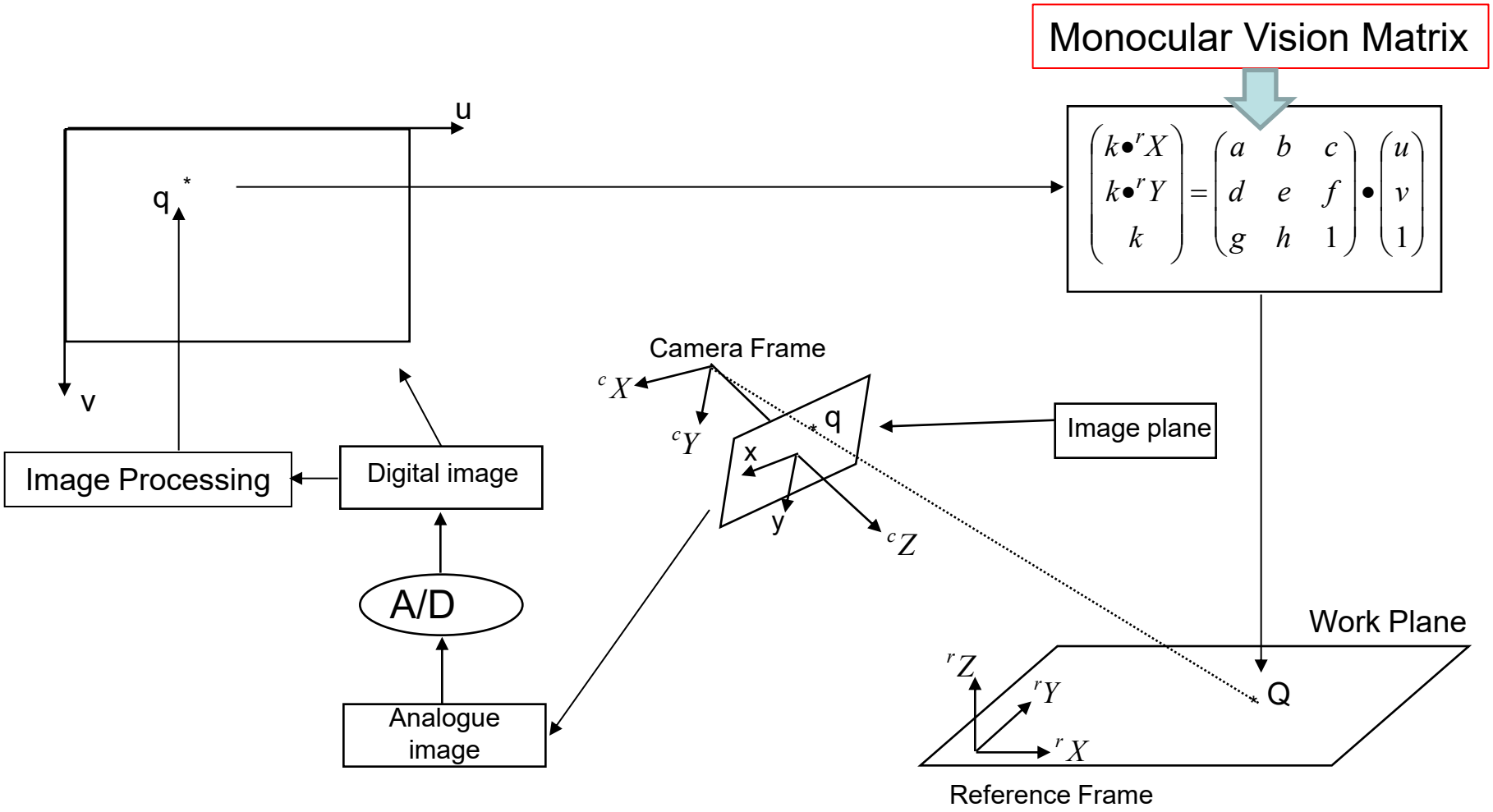
$$\begin{pmatrix} k \bullet {}^{r}X \\ k \bullet {}^{r}Y \\ k \end{pmatrix} = \mathrm{M}_{3\times3} \bullet \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \quad \text{with} \quad M_{3\times3} = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{pmatrix}$$

Equation of Monocular Vision's Inverse Projection



Equation of Monocular Vision

- Finally, we have proven the following equation in matrix form:

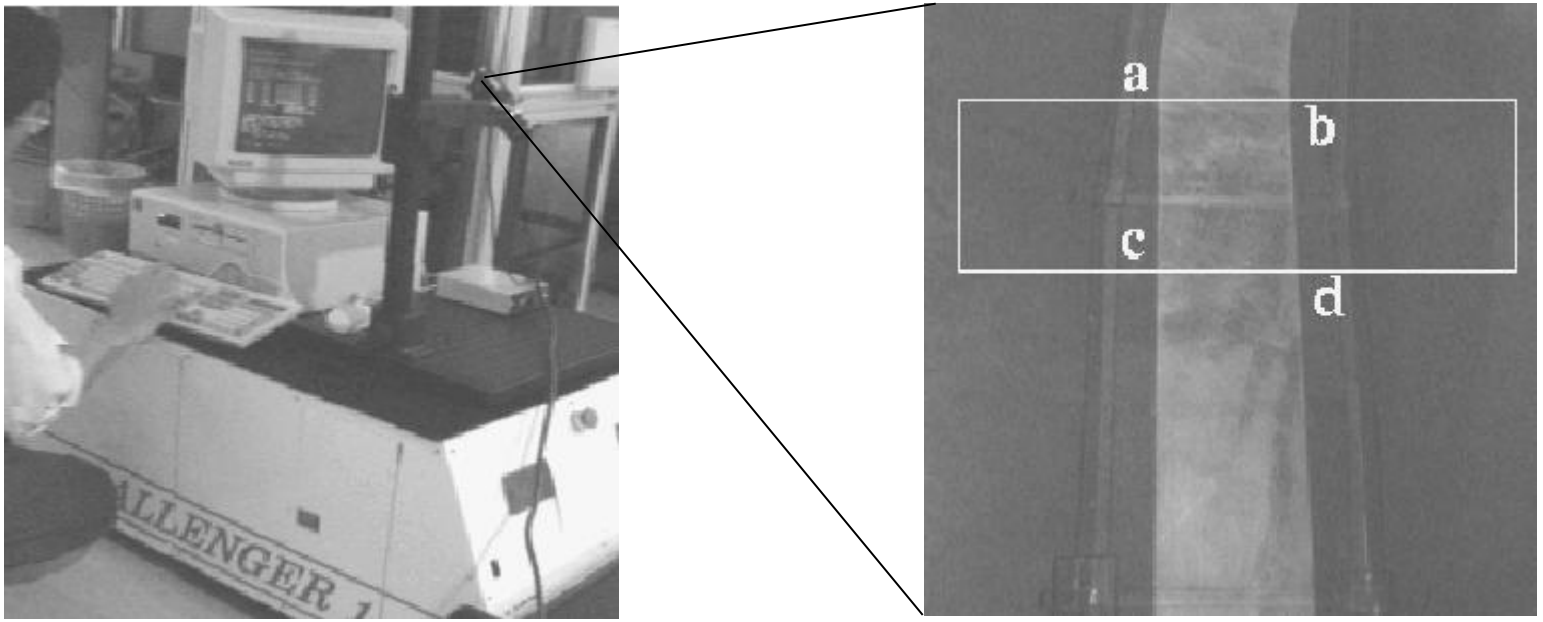


Exercise

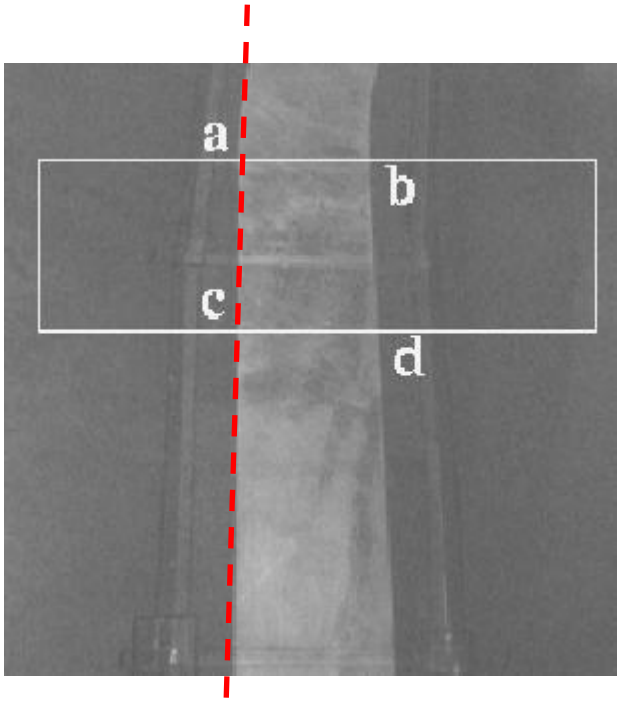
The following photos show:

a) a 2D vision system mounted on a AGV (Automatic Guided Vehicle) and

b) one image seen by the camera. The task for AGV is to follow the land-mark placed on the floor. The floor is assumed to be flat.



Exercise (continued)



- The 2D vision matrix is given as follows:

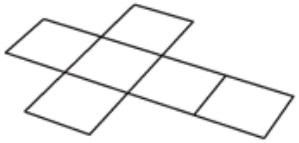
0.164499	0.022269	-27.176669
0.000767	-0.209450	74.306071
0.000018	0.001470	1.0

- If the digital image coordinates (u, v) of points “a” and “c” are a(220, 110) and c(215, 230) respectively, what is the equation of the line which passes through “a” and “c”?

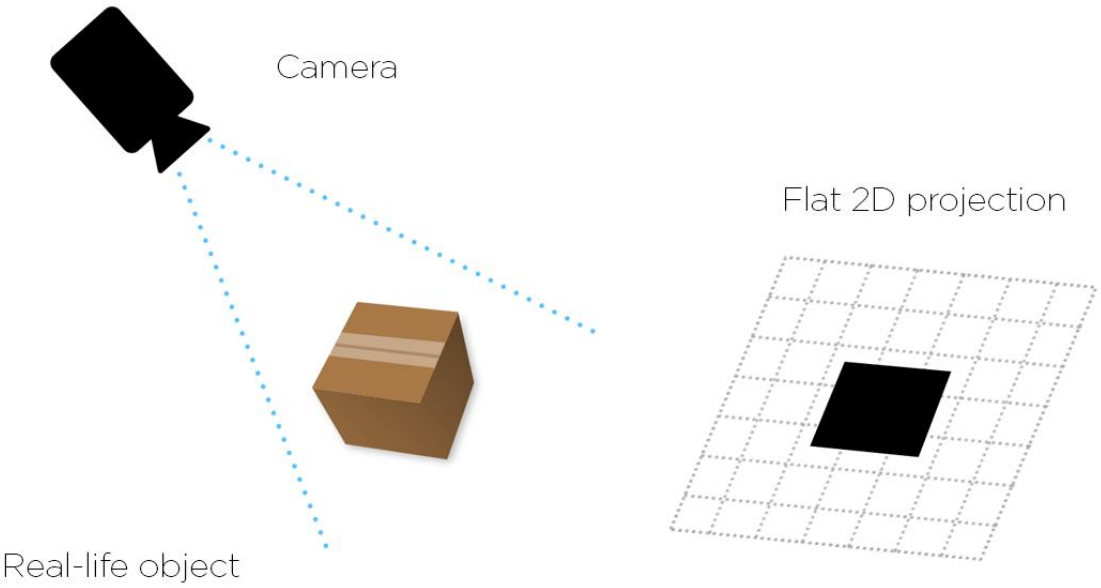
Hint:

Xr	Yr
9.833657	44.12548
9.920107	19.59617

Outline of Lecture 5

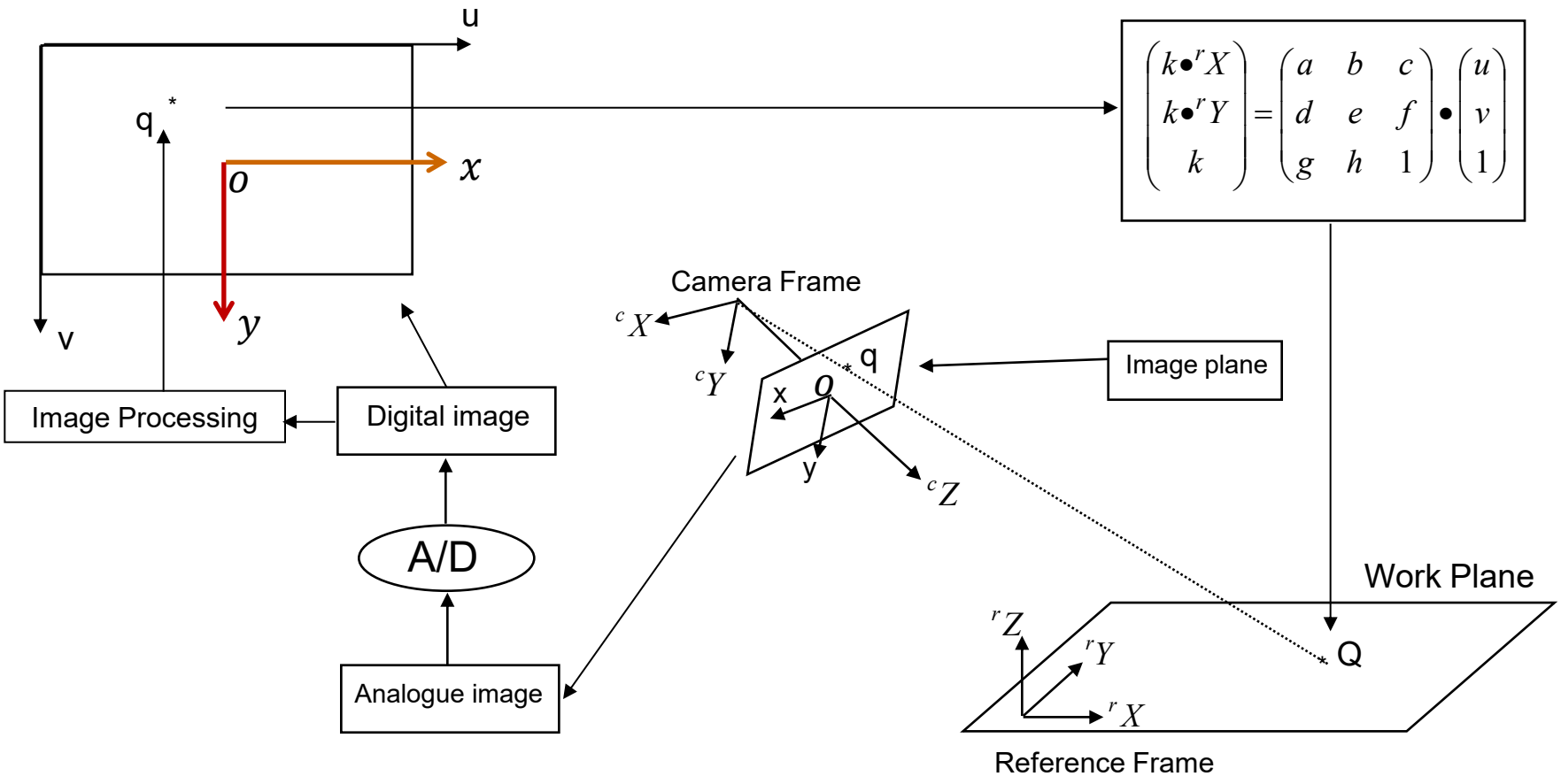


- Concept of Coordinate Transformation
- Transformation from Scene to Camera Space
- Transformation from Camera Space to Image Space
- Equation of Monocular Vision
- Calibration of Monocular Vision



Question

- How to determine the coefficients inside the monocular vision matrix?



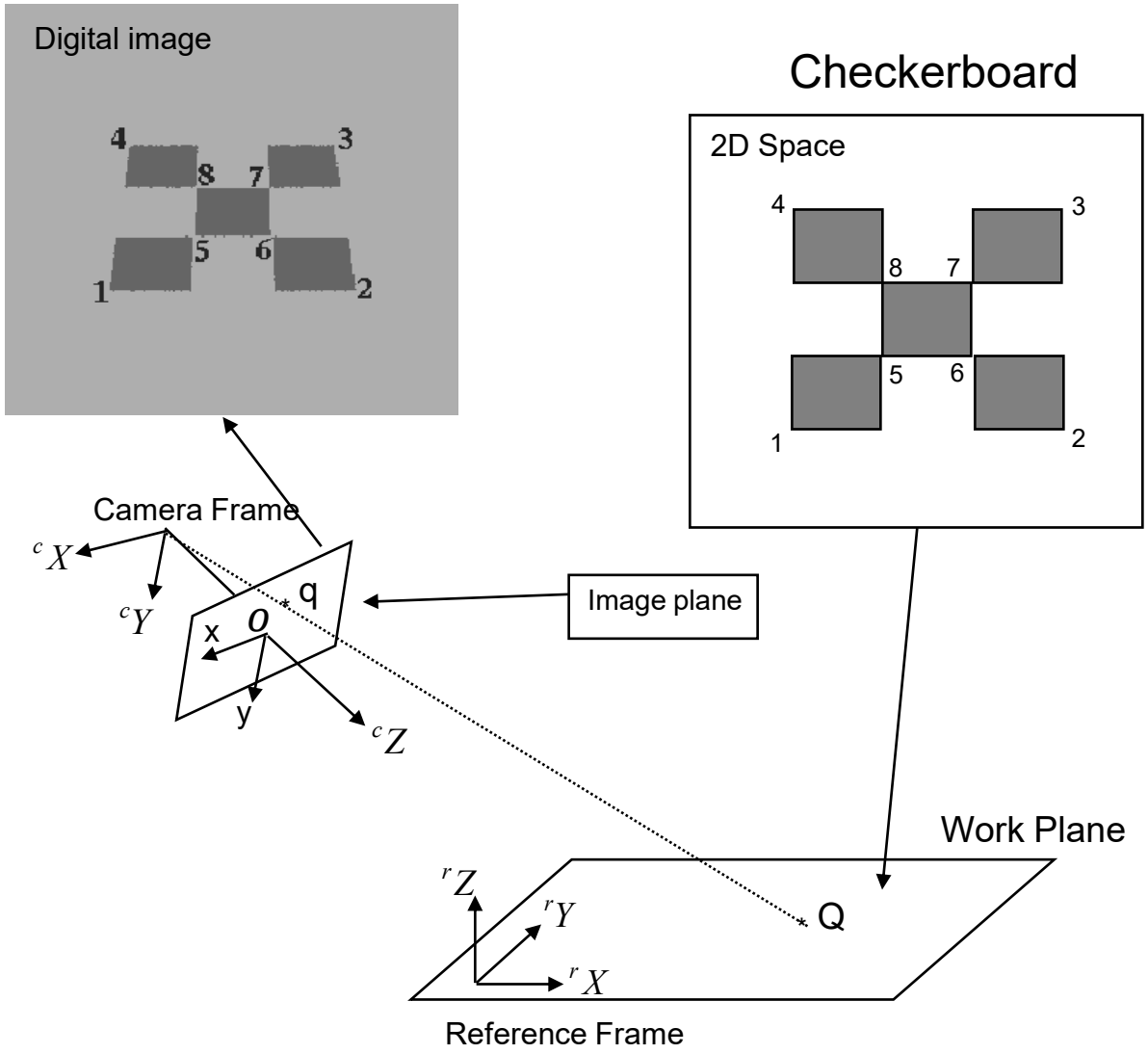
Calibration of Monocular Vision

Principle: (Could we call machine calibration as Machine Learning?)

- A regular patterns (i.e. checkerboard) are painted onto the object plane of a 2D space. The real dimensions of the patterns are known.
- From the digital image, some (at least four) image points are manually detected.
- Once we know the digital image coordinates and the 2D object coordinates of these points, the monocular vision matrix can be computed by a least-square estimation.

Calibration of Monocular Vision

- Setup



Calibration of Monocular Vision

- Procedure

Step 1: 2D object coordinates and digital image coordinates are related to a monocular vision matrix:

$$\begin{pmatrix} k \bullet {}^{r}X \\ k \bullet {}^{r}Y \\ k \end{pmatrix} = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{pmatrix} \bullet \begin{pmatrix} u \\ v \\ 1 \end{pmatrix}$$

Step 2: The matrix equation can be expanded into the following equations:

$$\begin{cases} {}^{r}X = \dfrac{u \bullet a + v \bullet b + c}{u \bullet g + v \bullet h + 1} \\ {}^{r}Y = \dfrac{u \bullet d + v \bullet e + f}{u \bullet g + v \bullet h + 1} \end{cases}$$

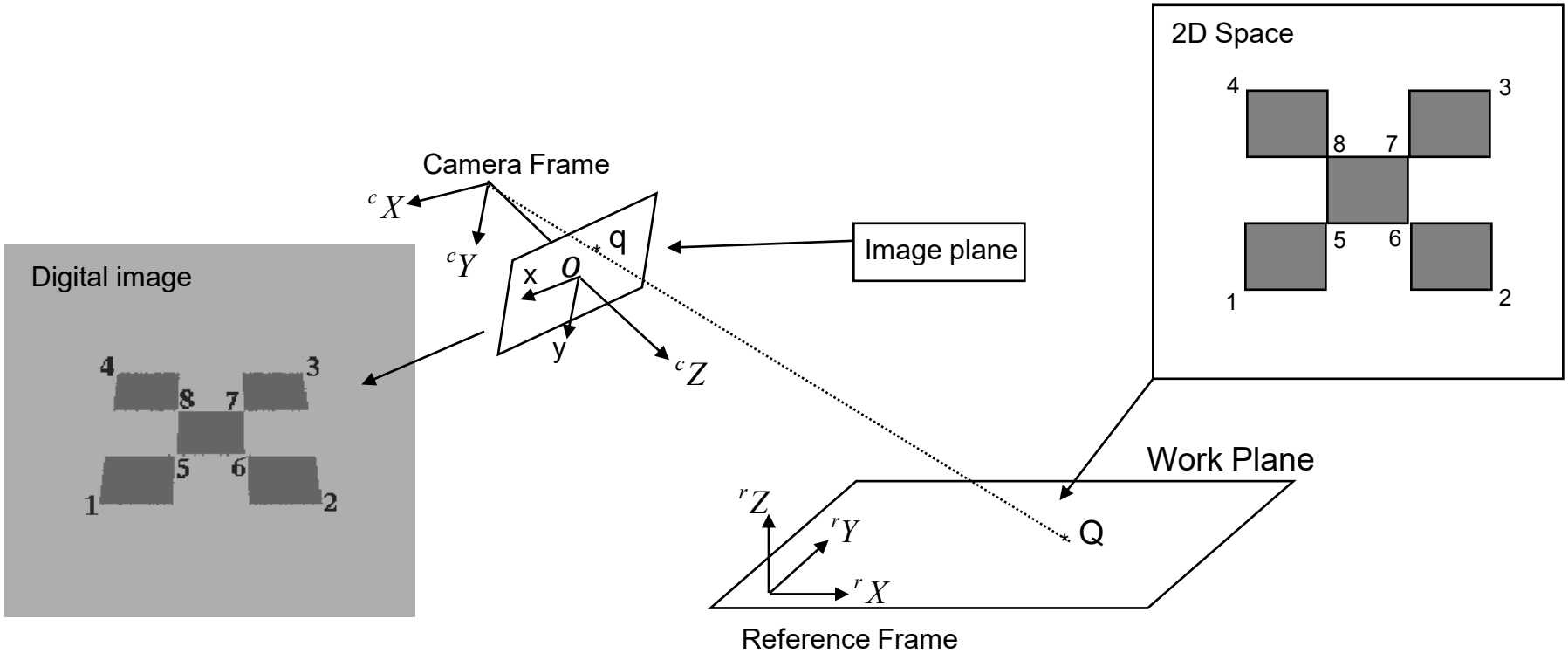
Step 2: The matrix equation can be expanded into the following equations :

$$\begin{cases} {}^{r}X = \dfrac{u \bullet a + v \bullet b + c}{u \bullet g + v \bullet h + 1} \\ {}^{r}Y = \dfrac{u \bullet d + v \bullet e + f}{u \bullet g + v \bullet h + 1} \end{cases}$$

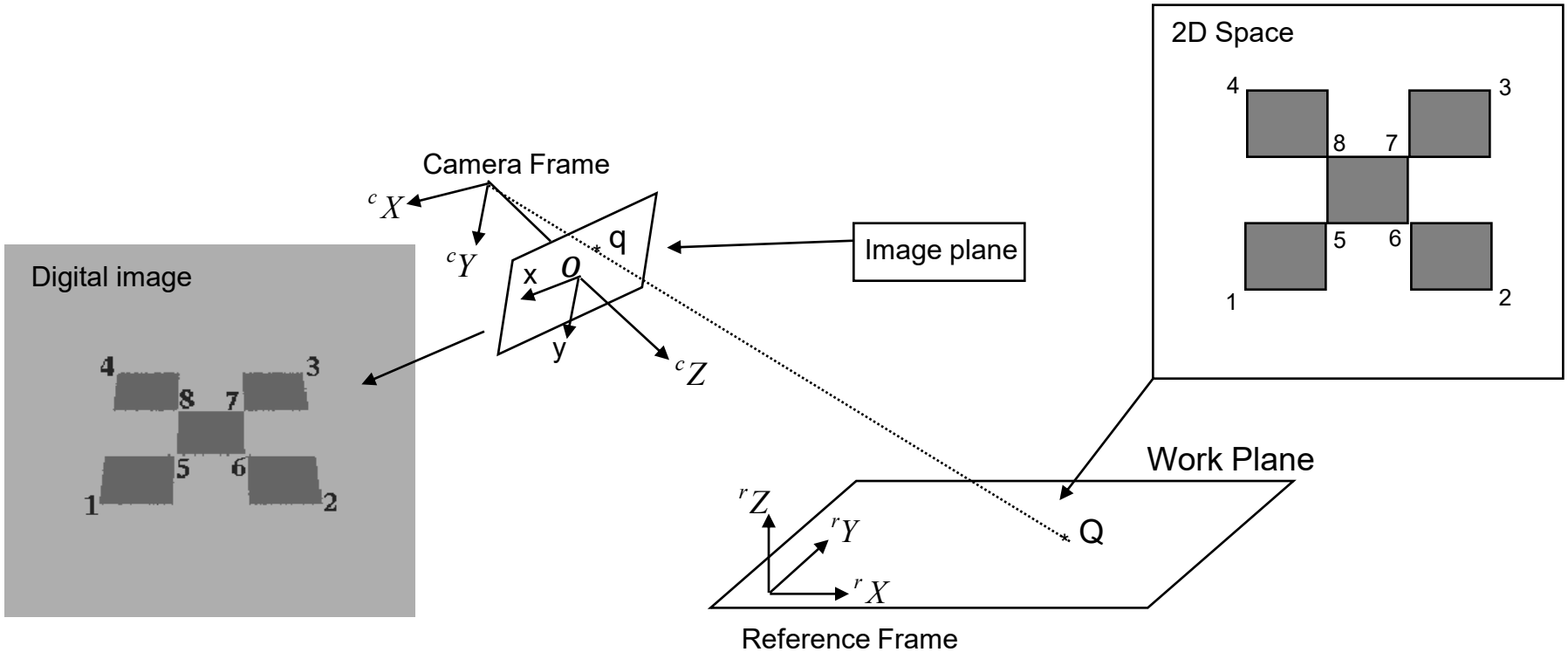
Step 3: If (u,v) and (X,Y) are known, the above two equations can be written in the following form:

$$\begin{cases} u \bullet a + v \bullet b + c - ({}^{r}X \bullet u) \bullet g - ({}^{r}X \bullet v) \bullet h = {}^{r}X. \\ u \bullet d + v \bullet e + f - ({}^{r}Y \bullet u) \bullet g - ({}^{r}Y \bullet v) \bullet h = {}^{r}Y. \end{cases}$$

Step 4: Given one pair of {(u.v), (X,Y)}, we can have two equations that are linear with respect to the unknown (a,b,c,d,e,f,g,).



Step 5: There are eight unknowns to be estimated. We need to have at least eight equations. Therefore, four pairs of {(u,v), (X,Y)} are sufficient enough for the estimation of the eight unknowns (a,b,c,d,e,f,g,h).



Step 6: If we have “n” pairs of {(ui, vi), (Xi, Yi)} (i=1, 2, …, n), we can establish “2n” linear equations:

$$\begin{cases} u_i \bullet a + v_i \bullet b + c - ({}^{r}X_i \bullet u_i) \bullet g - ({}^{r}X_i \bullet v_i) \bullet h = {}^{r}X_i. \\ u_i \bullet d + v_i \bullet e + f - ({}^{r}Y_i \bullet u_i) \bullet g - ({}^{r}Y_i \bullet v_i) \bullet h = {}^{r}Y_i. \end{cases}$$
$$i = 1,2,\ldots,n$$

$$A \bullet Z = B$$

With:

$$Z = \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \end{bmatrix}$$

$$A = \begin{bmatrix} u_1 & v_1 & 1 & 0 & 0 & 0 & -{}^{r}X_1 \bullet u_1 & -{}^{r}X_1 \bullet v_1 \\ 0 & 0 & 0 & u_1 & v_1 & 1 & -{}^{r}Y_1 \bullet u_1 & -{}^{r}Y_1 \bullet v_1 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ u_n & v_n & 1 & 0 & 0 & 0 & -{}^{r}X_n \bullet u_n & -{}^{r}X_n \bullet v_n \\ 0 & 0 & 0 & u_n & v_n & 1 & -{}^{r}Y_n \bullet u_n & -{}^{r}Y_n \bullet v_n \end{bmatrix}$$

$$B = \begin{bmatrix} {}^{r}X_1 \\ {}^{r}Y_1 \\ \ldots \\ {}^{r}X_n \\ {}^{r}Y_n \end{bmatrix}$$

Step 7: The solution with the least squared error is:

$$Z = (A^t \bullet A)^{-1} \bullet (A^t \bullet B)$$

With:

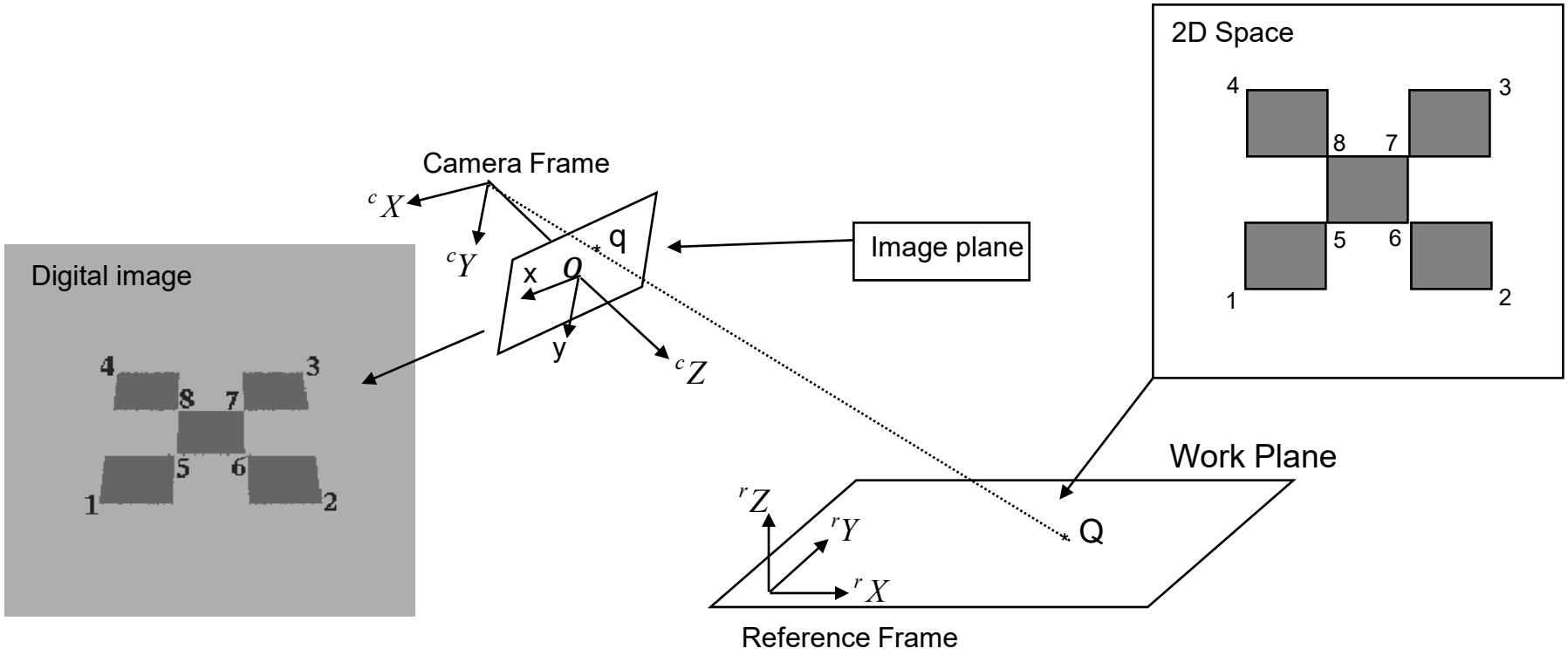
$$Z = \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \end{bmatrix}$$

$$A = \begin{bmatrix} u_1 & v_1 & 1 & 0 & 0 & 0 & -{}^rX_1 \bullet u_1 & -{}^rX_1 \bullet v_1 \\ 0 & 0 & 0 & u_1 & v_1 & 1 & -{}^rY_1 \bullet u_1 & -{}^rY_1 \bullet v_1 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ u_n & v_n & 1 & 0 & 0 & 0 & -{}^rX_n \bullet u_n & -{}^rX_n \bullet v_n \\ 0 & 0 & 0 & u_n & v_n & 1 & -{}^rY_n \bullet u_n & -{}^rY_n \bullet v_n \end{bmatrix}$$

$$B = \begin{bmatrix} {}^rX_1 \\ {}^rY_1 \\ \dots \\ {}^rX_n \\ {}^rY_n \end{bmatrix}$$

Experimental Results

- The sizes of the black square in a 2D space are 10.0x10.0 (cm). A CCD camera is fixed at a location above the 2D plane.



Experimental Results (continued)

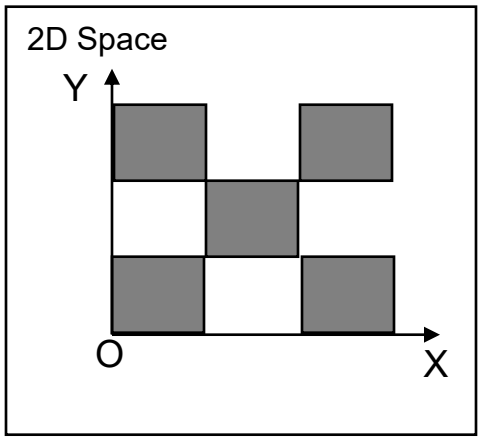
- From the digital image captured by the camera, we select eight image points. The digital image coordinates (u, v) of these eight points are:

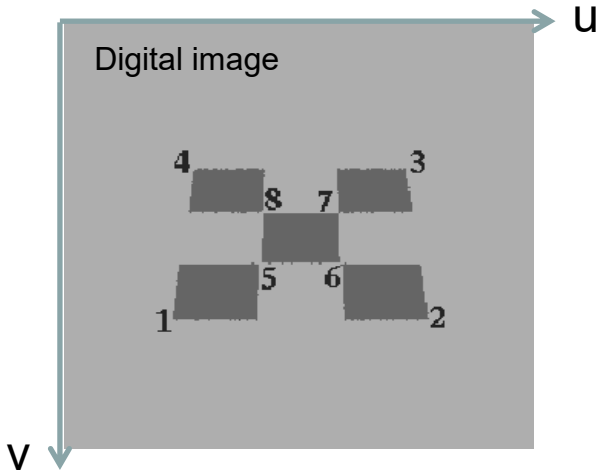
Point 1: (117, 355) Point 2: (396, 356)

Point 3: (372, 175) Point 4: (142, 175)

Point 5: (213, 288) Point 6: (301, 288)

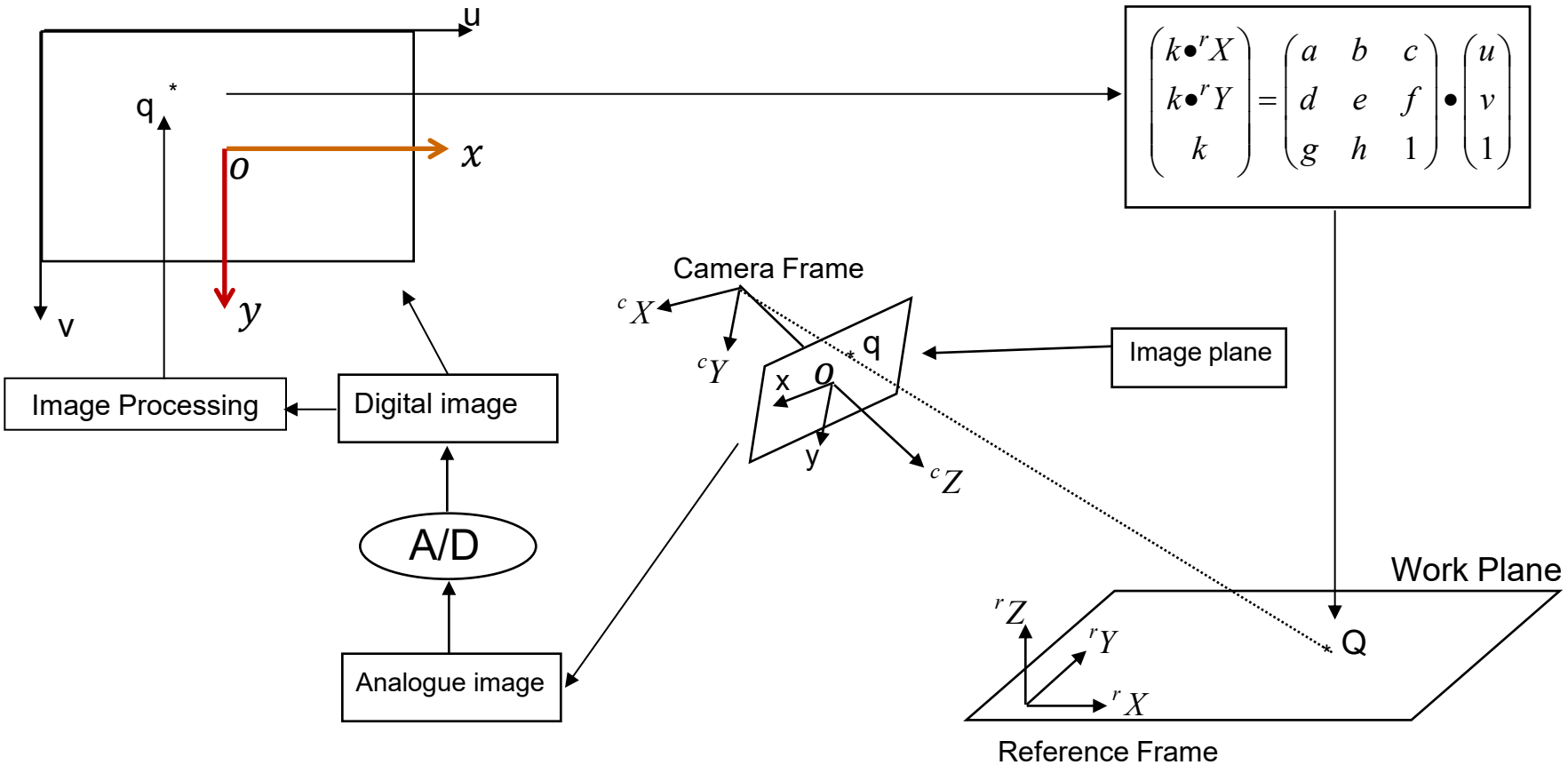
Point 7: (297, 228) Point 8: (215, 227)





Experimental Results (continued)

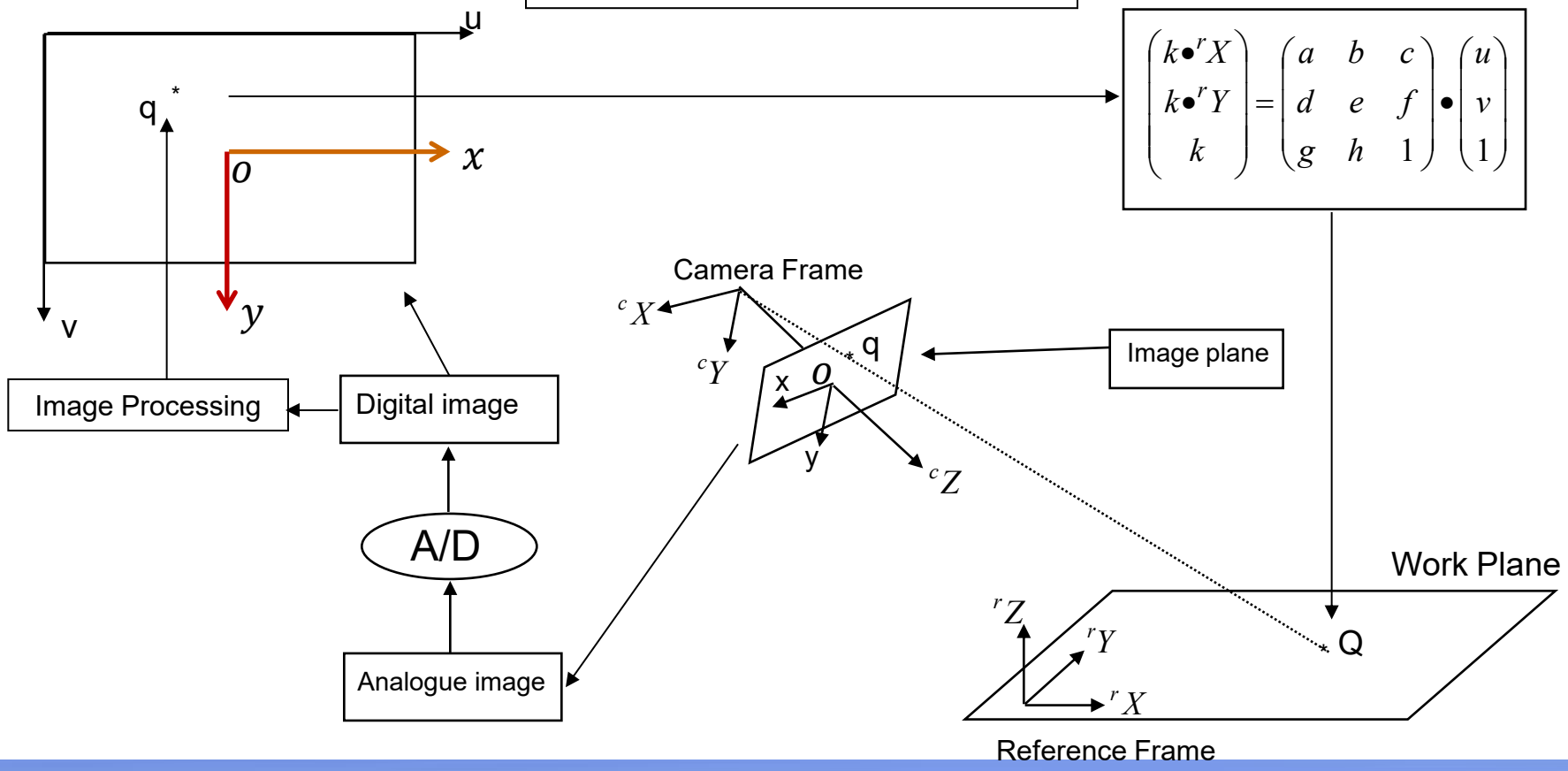
- What are the coefficients inside the monocular vision matrix?



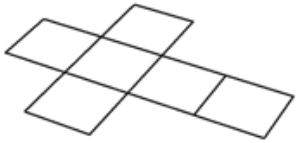
Experimental Results (continued)

- Answer:

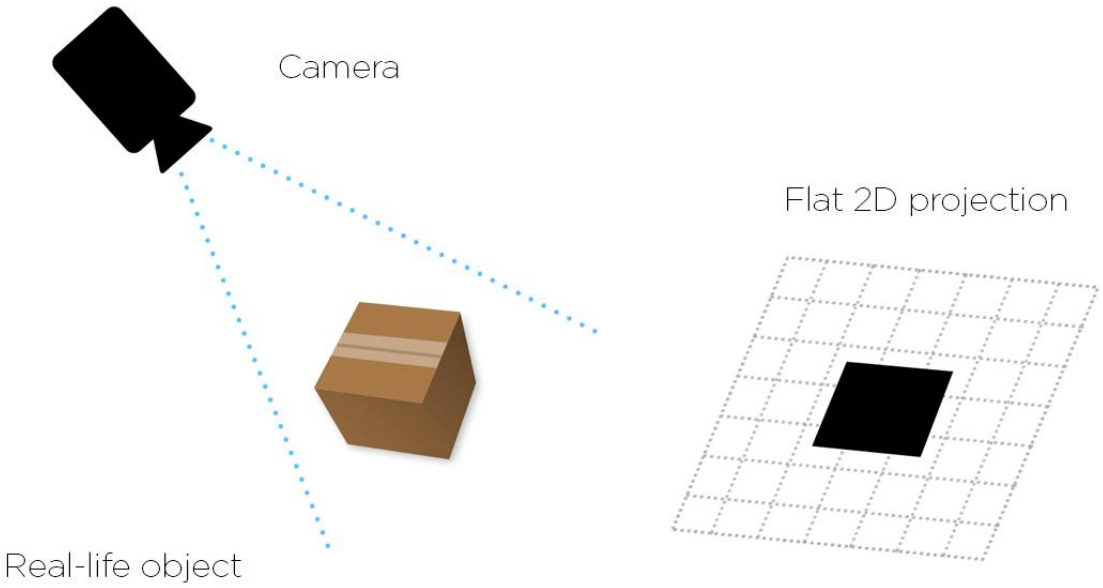
0.164499	0.022269	-27.176669
0.000767	-0.209450	74.306071
0.000018	0.001470	1.0



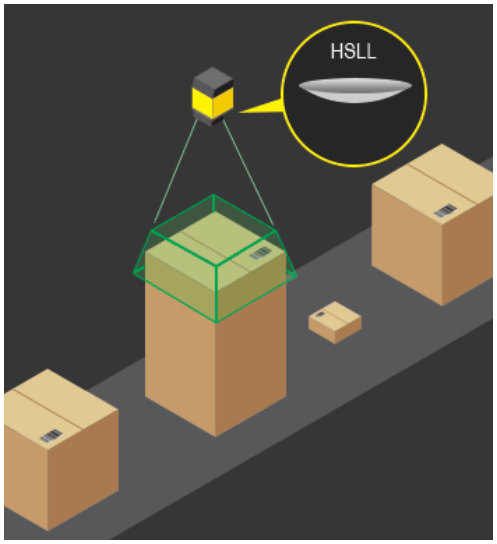
Summary of Lecture 5



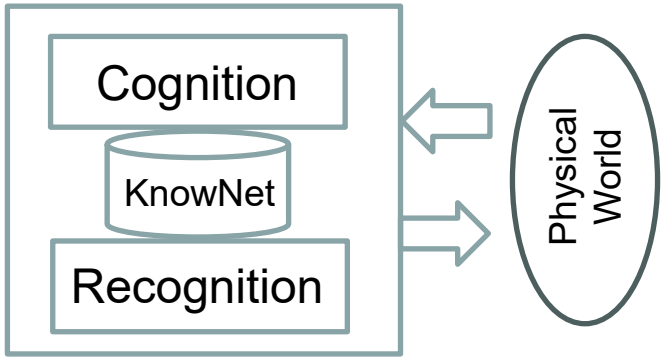
- Concept of Coordinate Transformation
- Transformation from Scene to Camera Space
- Transformation from Camera Space to Image Space
- Equation of Monocular Vision
- Calibration of Monocular Vision



Outline of Module 4



- Lecture 1:
  - Geometry-Driven Computation
- Lecture 2:
  - Fuzziness-Driven Computation
- Lecture 3:
  - Cognition-Driven Computation
- Lecture 4:
  - Recognition-Driven Computation
- Lecture 5:
  - Computation Using Monocular Vision
- Lecture 6:
  - Computation Using Binocular Vision



NANYANG TECHNOLOGICAL UNIVERSITY

School of Mechanical & Aerospace Engineering

Design, Machine, Control, Intelligence

Lecture 6 of Module 4

AI 3.0

MA4829 Machine Intelligence

Computation Using Binocular Vision

XIE Ming, PhD (France)

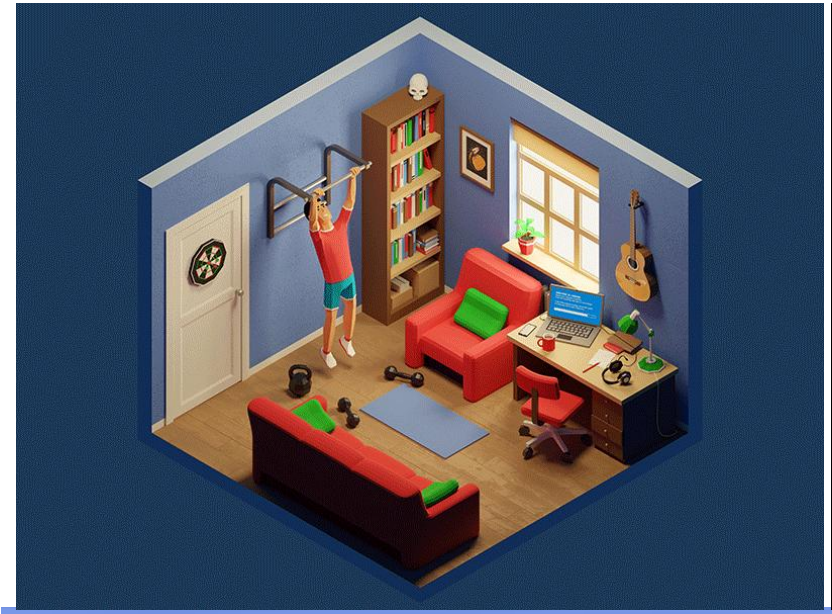
http://personal.ntu.edu.sg/mmxie

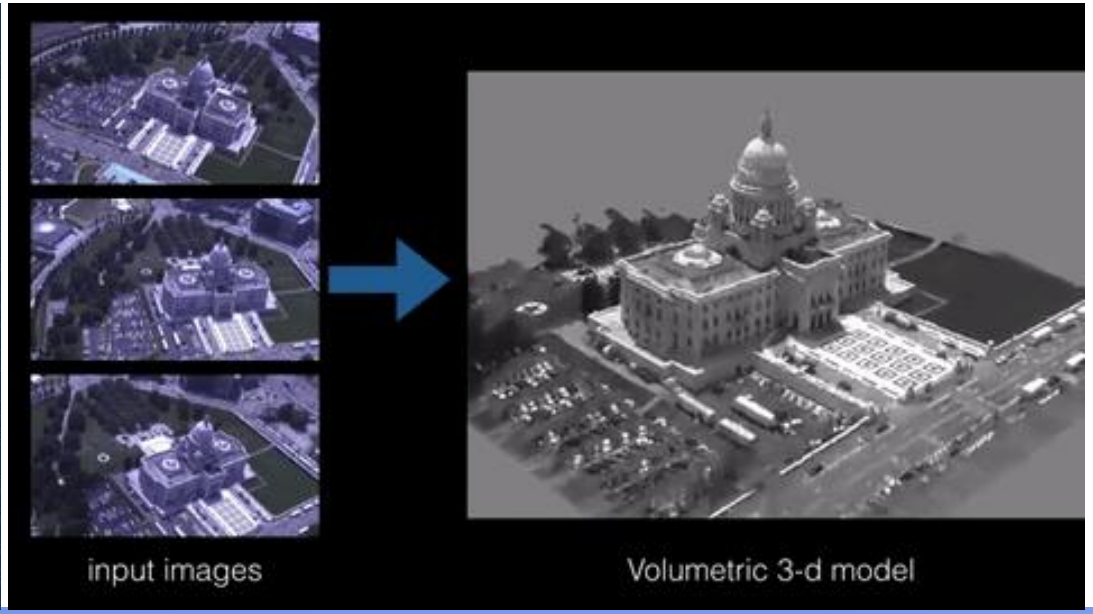
"We are living inside an ocean of signals"

What is a system with intelligence inside?

Outline of Lecture 6

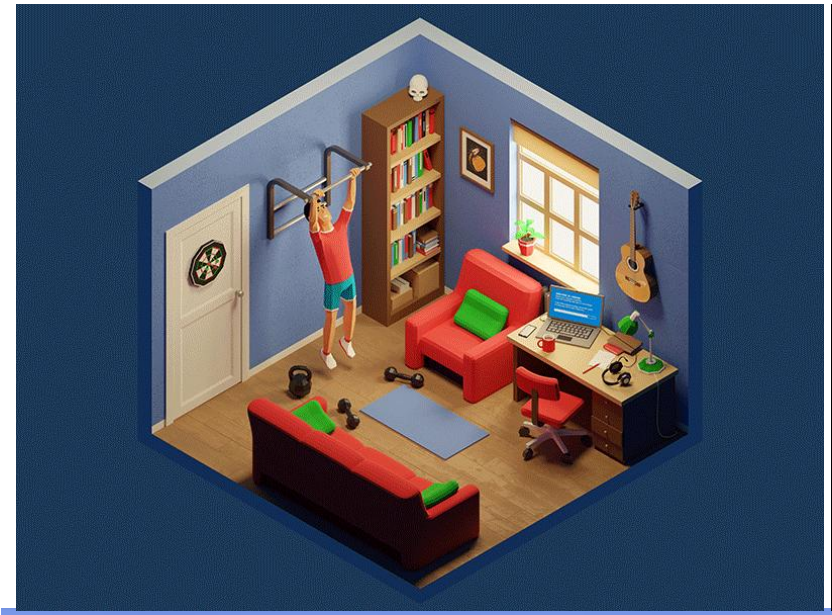
- Concept of Coordinate Transformation
- Transformation from Scene to Camera Space
- Transformation from Camera Space to Image Space
- Equation of Binocular Vision
- Calibration of Binocular Vision

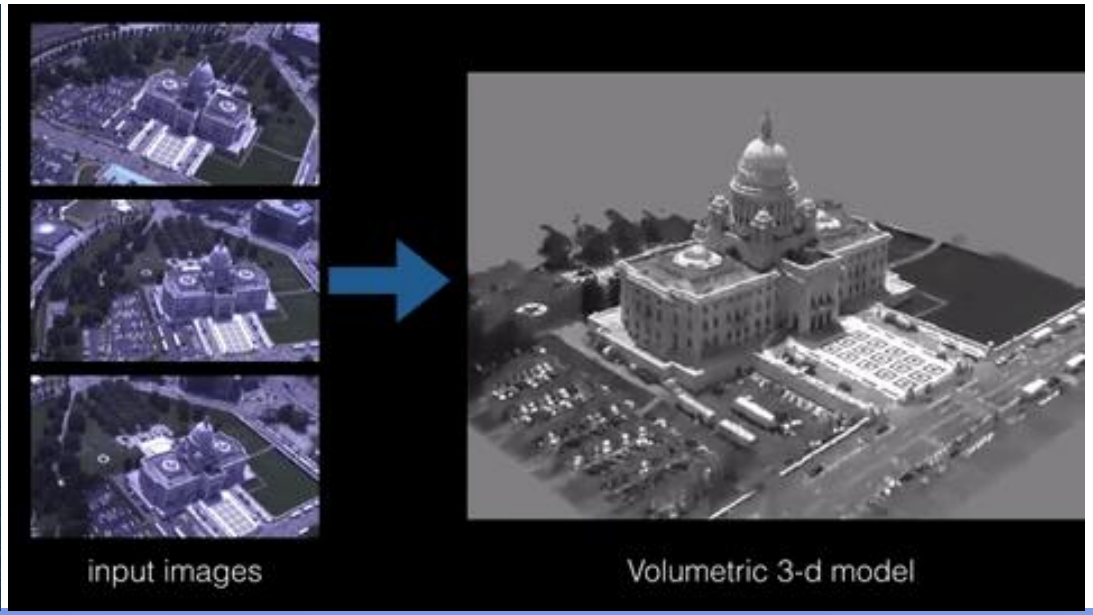




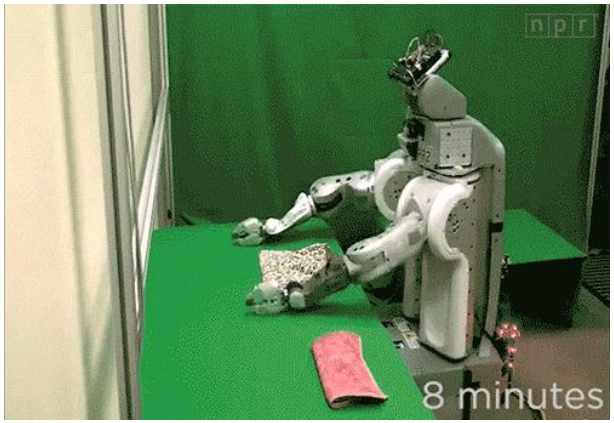
Outline of Lecture 6

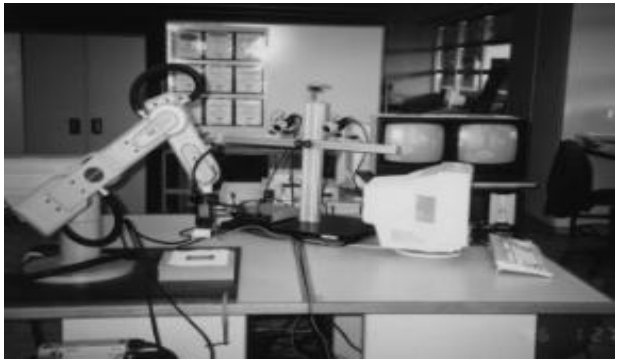
- Concept of Coordinate Transformation
- Transformation from Scene to Camera Space
- Transformation from Camera Space to Image Space
- Equation of Binocular Vision
- Calibration of Binocular Vision





Vision-Guided Manipulation ...

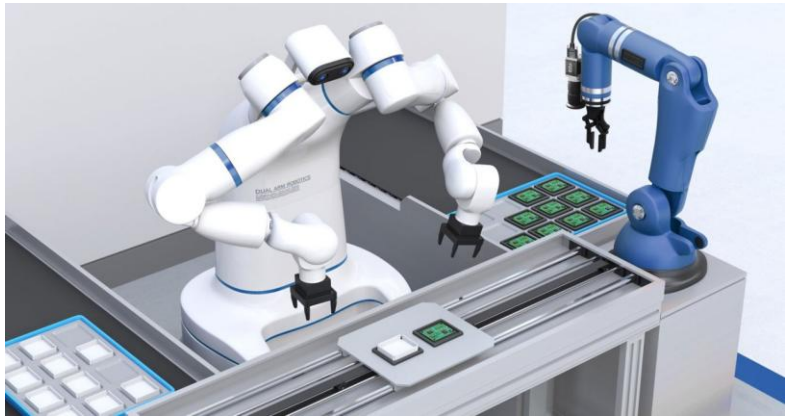


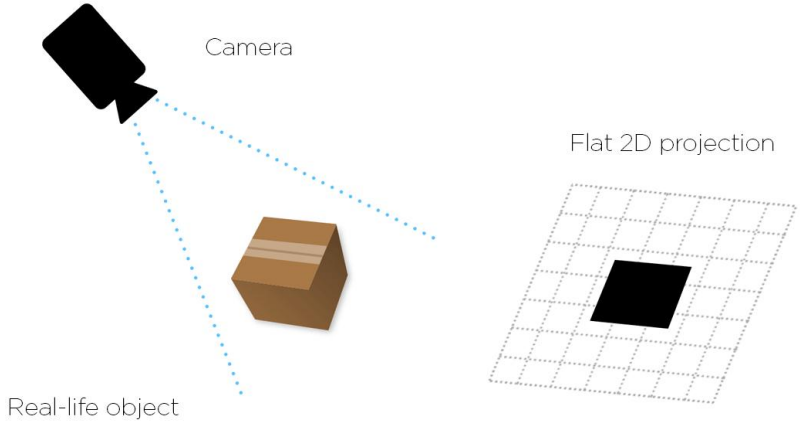


Vision-Guided Mobile Manipulation …

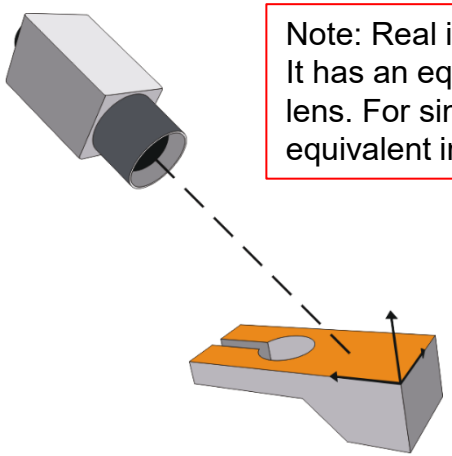
A vision system itself is a mechanical system …

- A mechanical system consists of many rigid elements, devices or components.
- Hence, it is an important topic to study the spatial relationship among rigid elements, devices and components.
- As a result, each element, device or component will be assigned a local coordinate system.

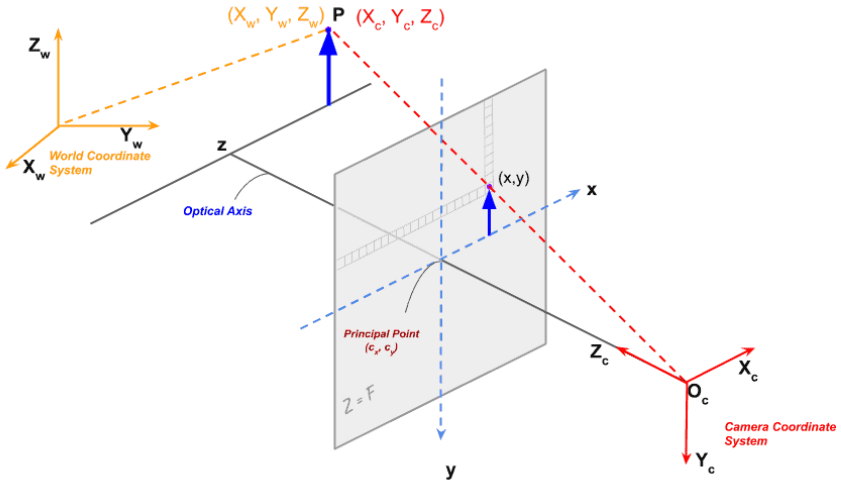


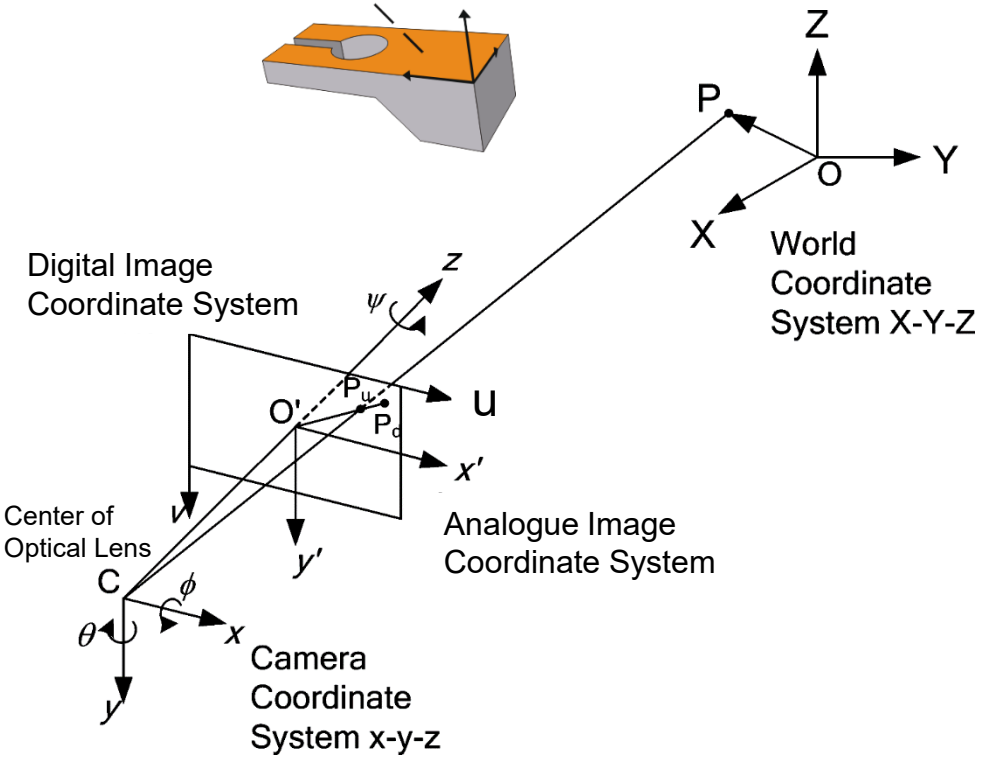


Example of Coordinate Systems Assigned to a Camera and its Images Inside …



Note: Real image is inside the camera. It has an equivalent image in front of the lens. For simplicity, we use a real image's equivalent image.





What is the scenario of coordinate transformations?

Basics of Homogeneous Transformation

- Pose = Position + Orientation

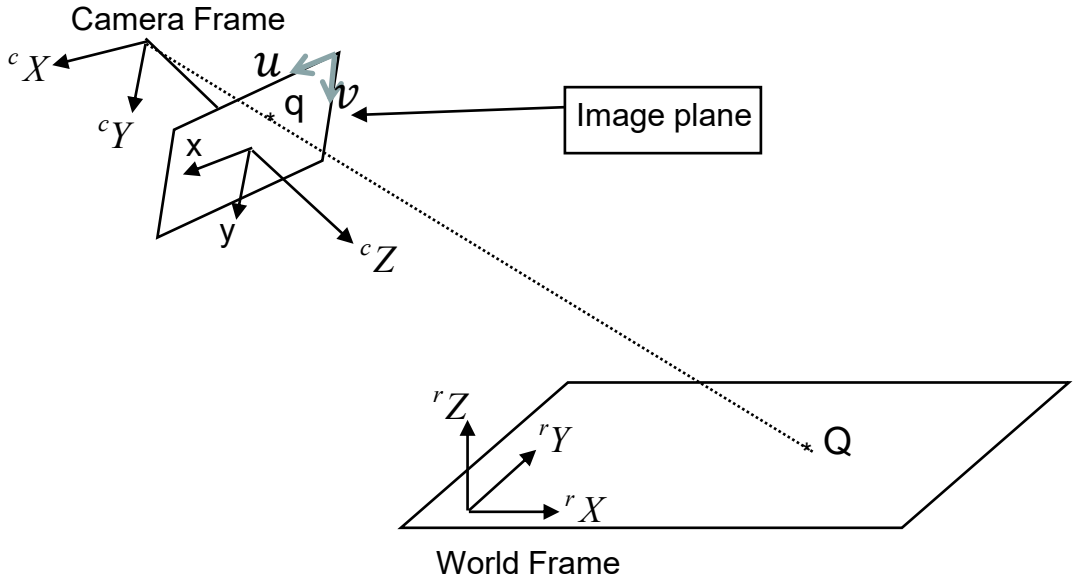
$$H_{camera} = \begin{bmatrix} R_{camera} & T_{camera} \\ 0 \quad 0 \quad 0 & 1 \end{bmatrix}$$

This is the pose of the camera frame with respect to the world frame.

$$H_{world} = \begin{bmatrix} R_{camera}^{-1} & -R_{camera}^{-1} \times T_{camera} \\ 0 \quad 0 \quad 0 & 1 \end{bmatrix}$$

(This is the pose of the world frame with respect to the camera frame)

(Monocular Vision)



$$H_{camera} \times H_{world} = I_{4\times4}$$

$$H_{camera} = H_{world}^{-1}$$

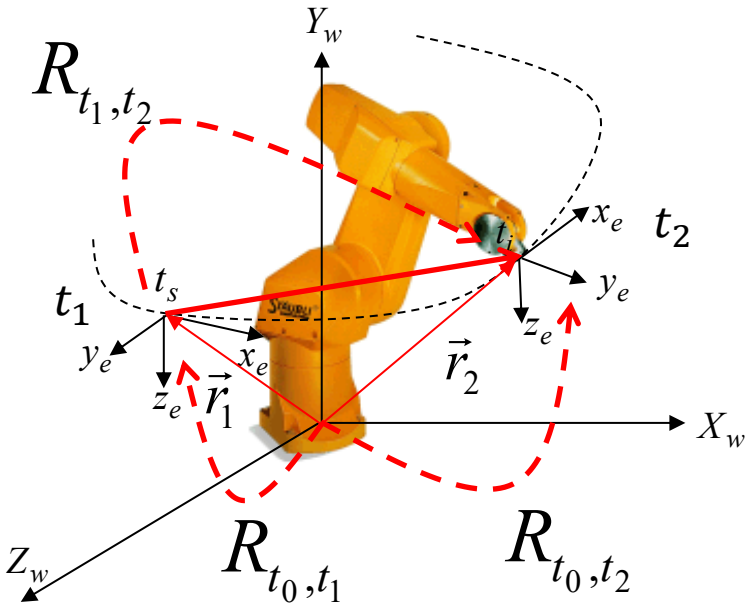
Basics of Homogeneous Transformation

This is the pose of the tooltip frame with respect to the world frame

$$H_{tooltip} = \begin{bmatrix} R_{tooltip} & T_{tooltip} \\ 0 \quad 0 \quad 0 & 1 \end{bmatrix}$$

This is the pose of the world frame with respect to the tooltip frame

$$H_{world} = \begin{bmatrix} R_{tooltip}^{-1} & -R_{tooltip}^{-1} \times T_{tooltip} \\ 0 \quad 0 \quad 0 & 1 \end{bmatrix}$$



(Robot Arm)

$$H_{tooltip} \times H_{world} = I_{4\times4}$$

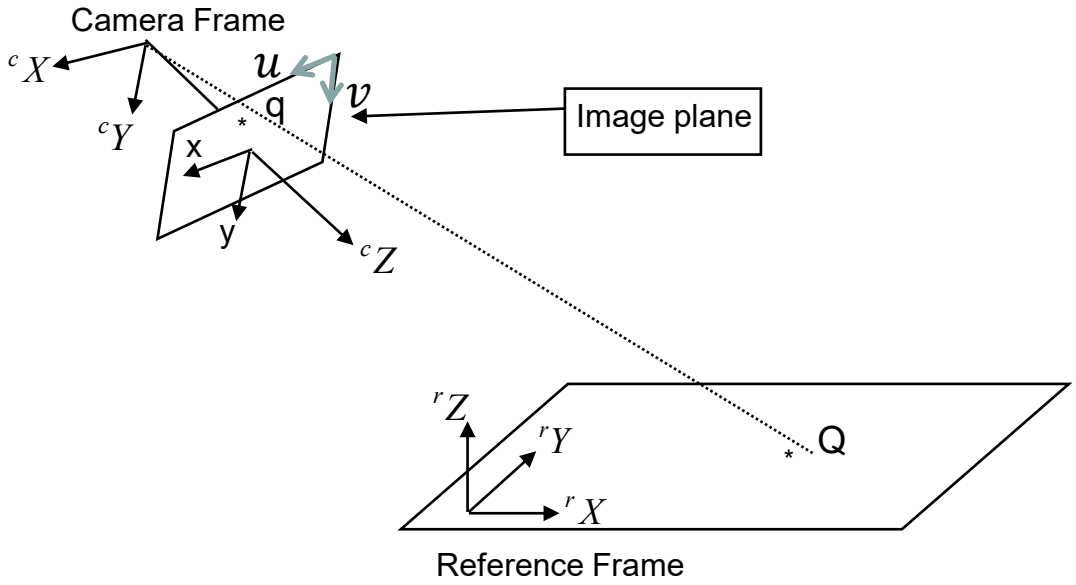
$$H_{tooltip} = H_{world}^{-1}$$

Transformation of Coordinates in 3D Space

- The coordinates in the reference frame can be transformed into the coordinates in the camera frame.

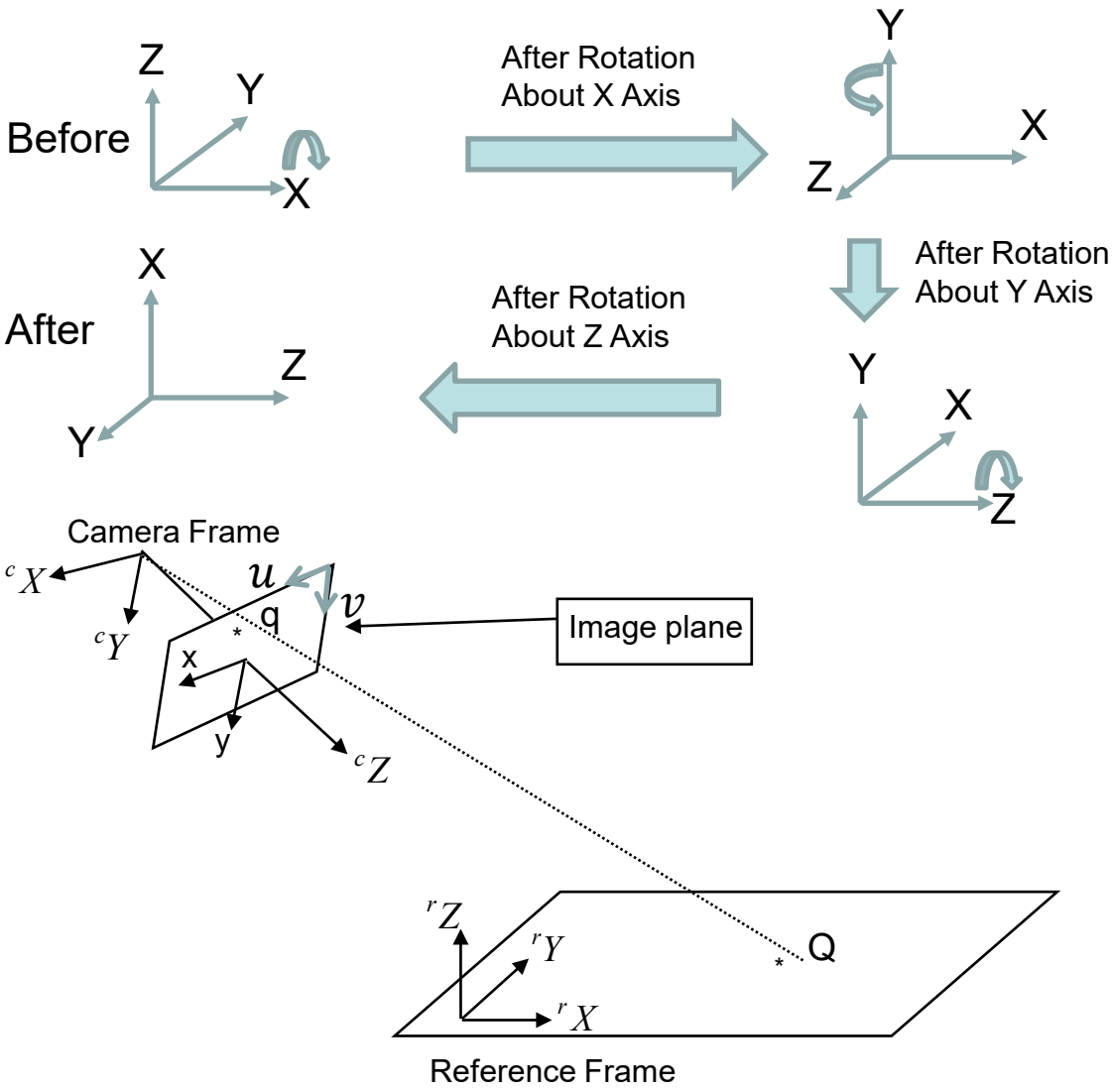
This is the pose of the reference frame with respect to the camera frame

$$ {}^{c}H_{r} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} $$



$$ \begin{bmatrix} {}^{c}X \\ {}^{c}Y \\ {}^{c}Z \\ 1 \end{bmatrix} = {}^{c}H_{r} \bullet \begin{bmatrix} {}^{r}X \\ {}^{r}Y \\ {}^{r}Z \\ 1 \end{bmatrix} $$

Practices with Rotational Transformation




```
1
2 -   ax = 90*pi/180;
3 -   rx = [1   0          0 ;
4           0  cos(ax)   -sin(ax);
5           0  sin(ax)    cos(ax)];
6
7 -   ay = 90*pi/180;
8 -   ry = [cos(ay)   0    sin(ay);
9             0       1    0 ;
10           -sin(ay) 0    cos(ay)];
11
12 -  az = 90*pi/180;
13 -  rz = [cos(az)   -sin(az)   0;
14          sin(az)    cos(az)   0;
15          0          0         1];
16
17 -  r = rx*ry*rz
18
19
```

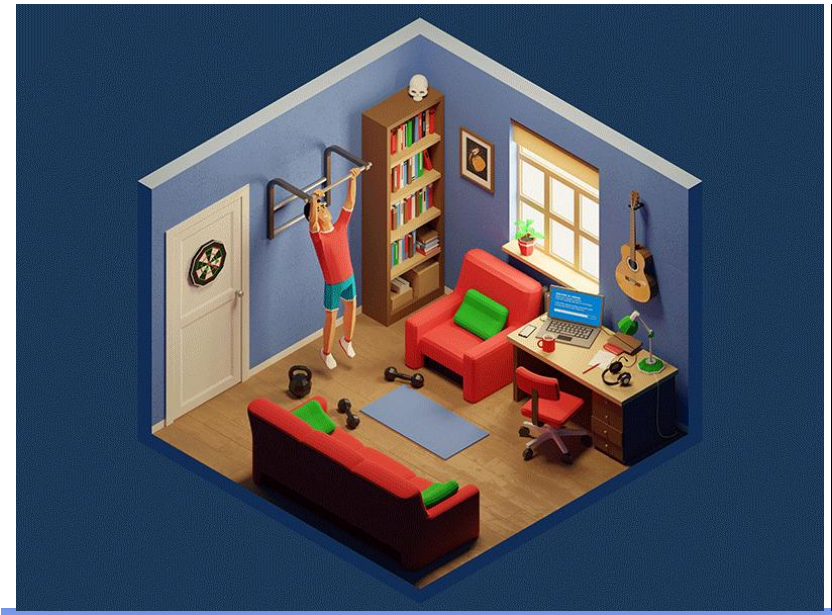
Command Window

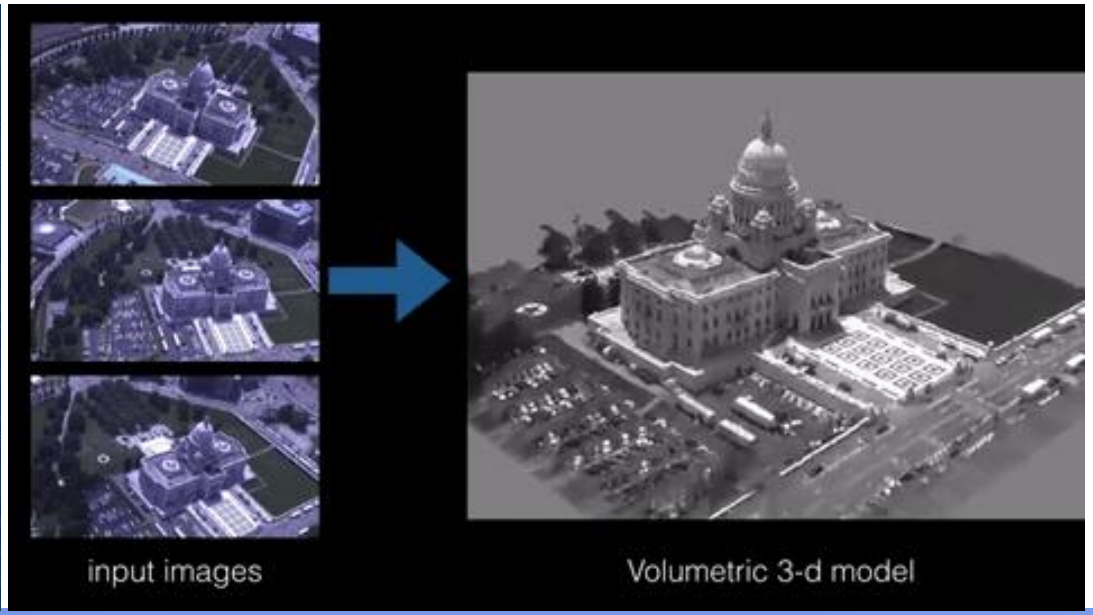
New to MATLAB? See resources for Getting Started.

```
r =

    0.0000   -0.0000    1.0000
    0.0000   -1.0000   -0.0000
    1.0000    0.0000    0.0000
```

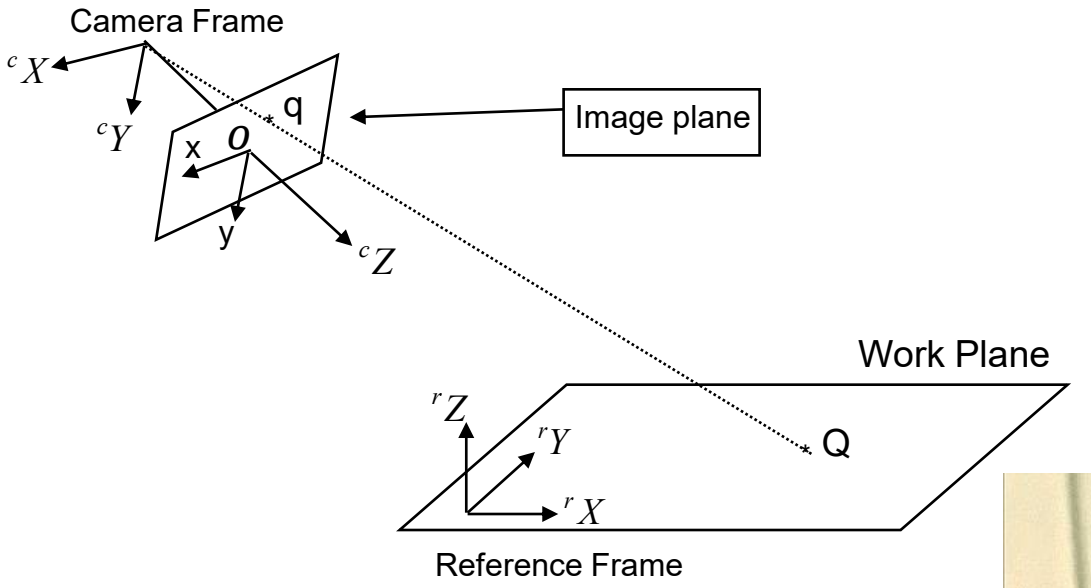
Outline of Lecture 6

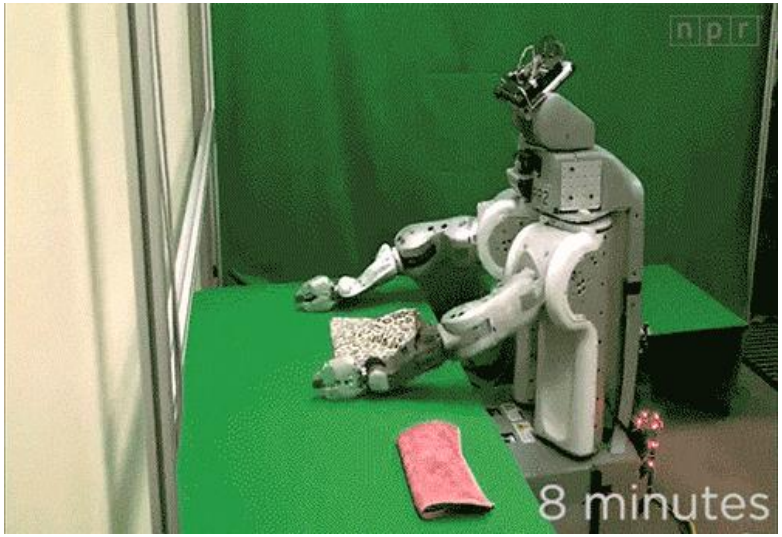
- Concept of Coordinate Transformation
- Transformation from Scene to Camera Space
- Transformation from Camera Space to Image Space
- Equation of Binocular Vision
- Calibration of Binocular Vision





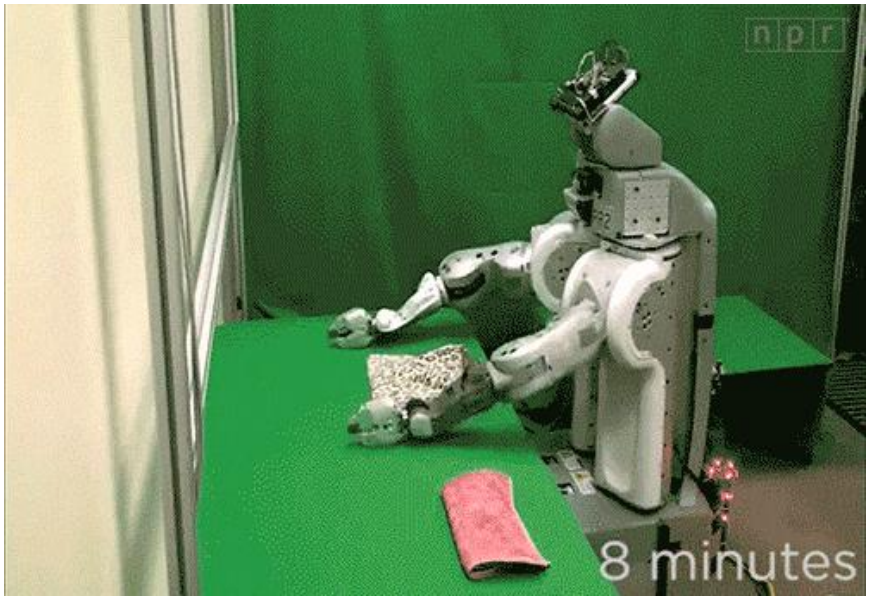
Setup of Coordinate Systems …

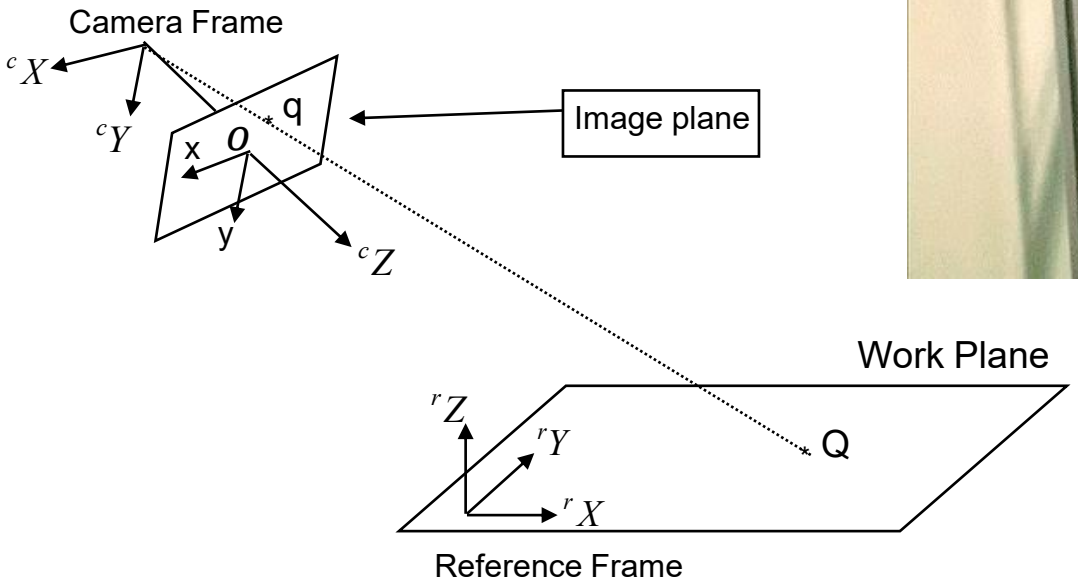




Representation of Scene with Respect to Camera's Coordinate System …

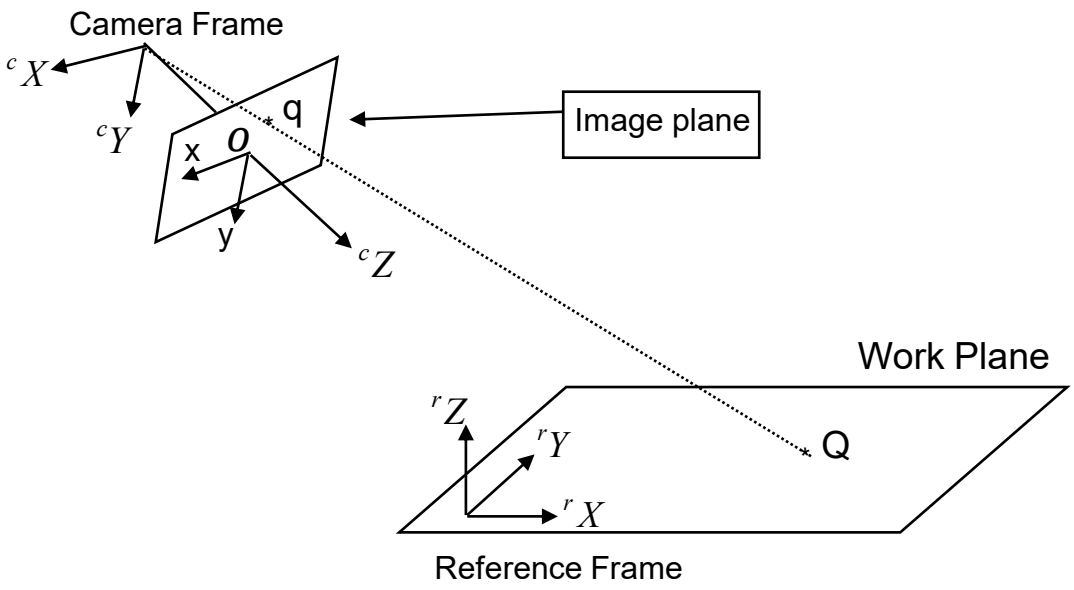
1. Position
2. Orientation





$$ {}^{c}H_{r} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Transformation from Scene to Camera Space …

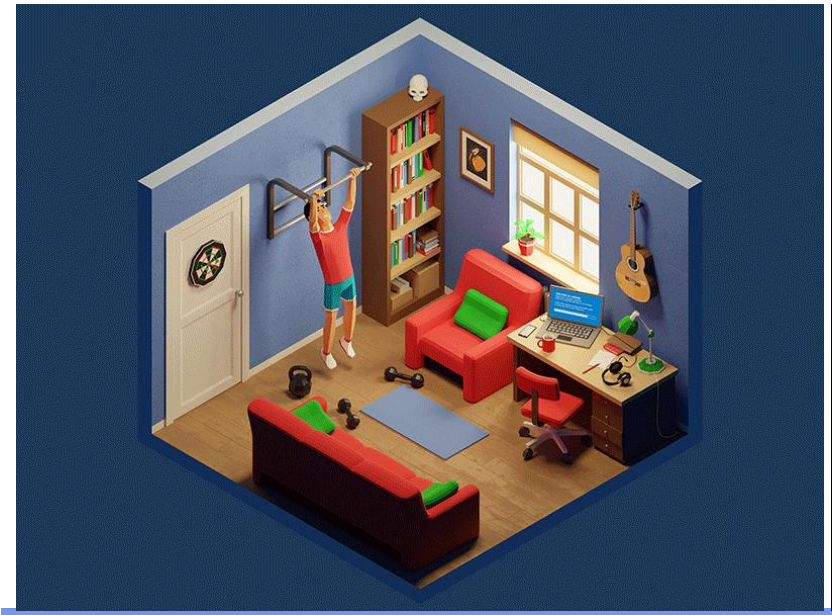


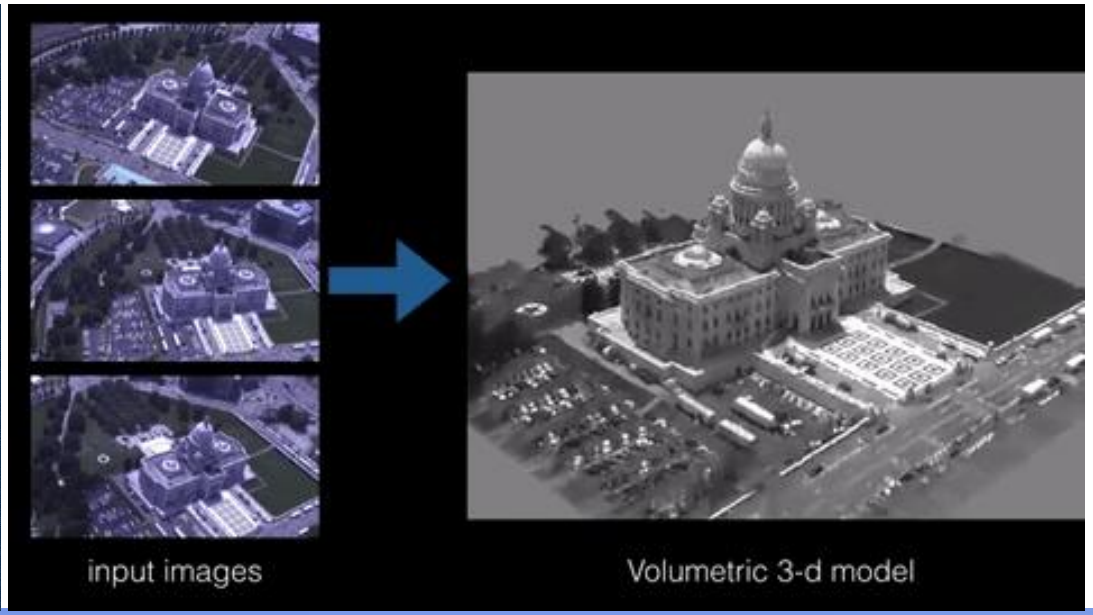
$$^cH_r = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} ^cX \\ ^cY \\ ^cZ \\ 1 \end{bmatrix} = {}^cH_r \bullet \begin{bmatrix} ^rX \\ ^rY \\ ^rZ \\ 1 \end{bmatrix}$$

Outline of Lecture 6

- Concept of Coordinate Transformation
- Transformation from Scene to Camera Space
- Transformation from Camera Space to Image Space
- Equation of Binocular Vision
- Calibration of Binocular Vision





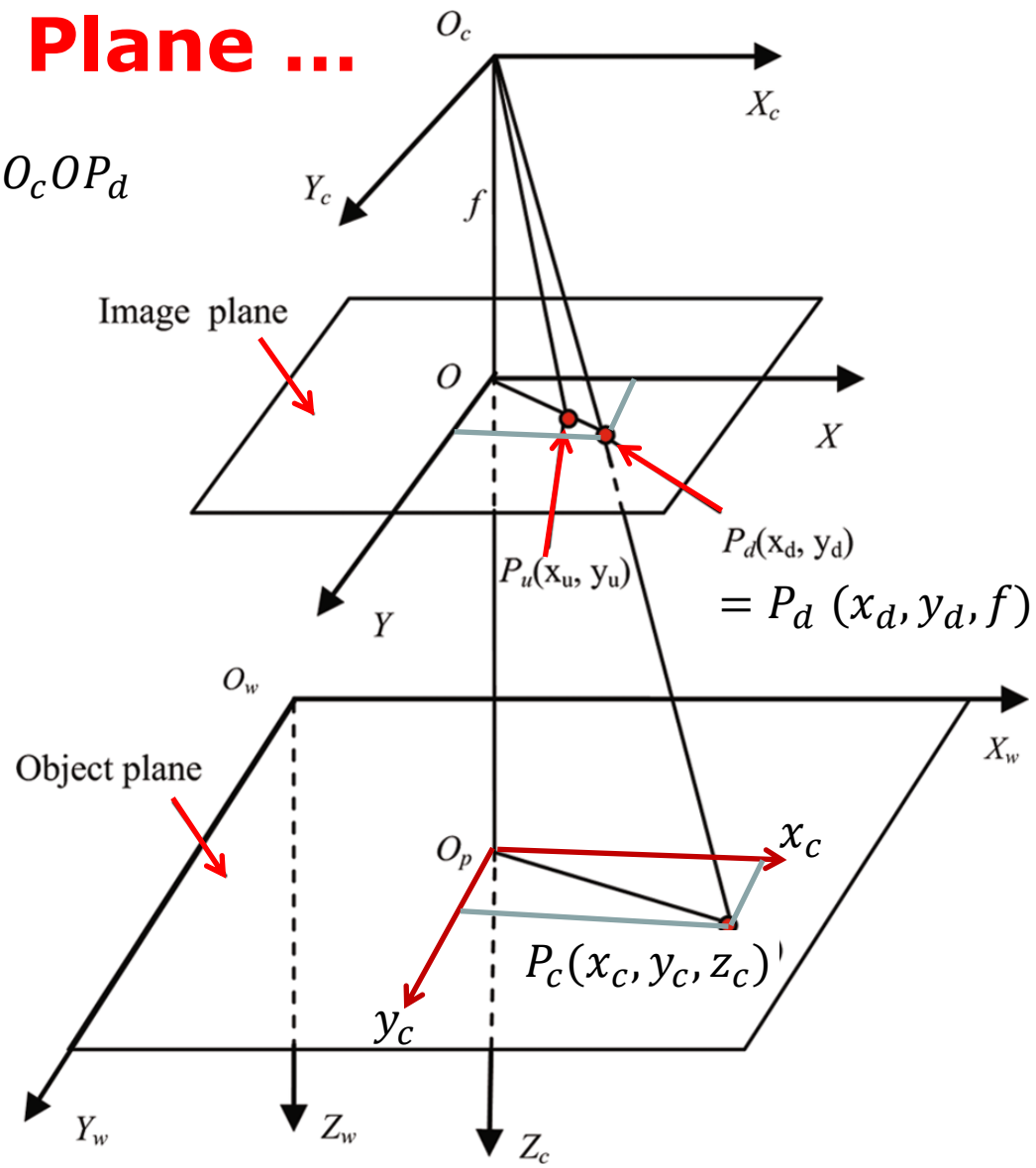
Projection to Image Plane …

Triangle $O_c O_p P_c$ is similar to Triangle $O_c O P_d$

$$\frac{O_p O_c}{O O_c} = \frac{P_c O_p}{P_d O} = \frac{z_c}{f} = \frac{x_c}{x_d} = \frac{y_c}{y_d}$$

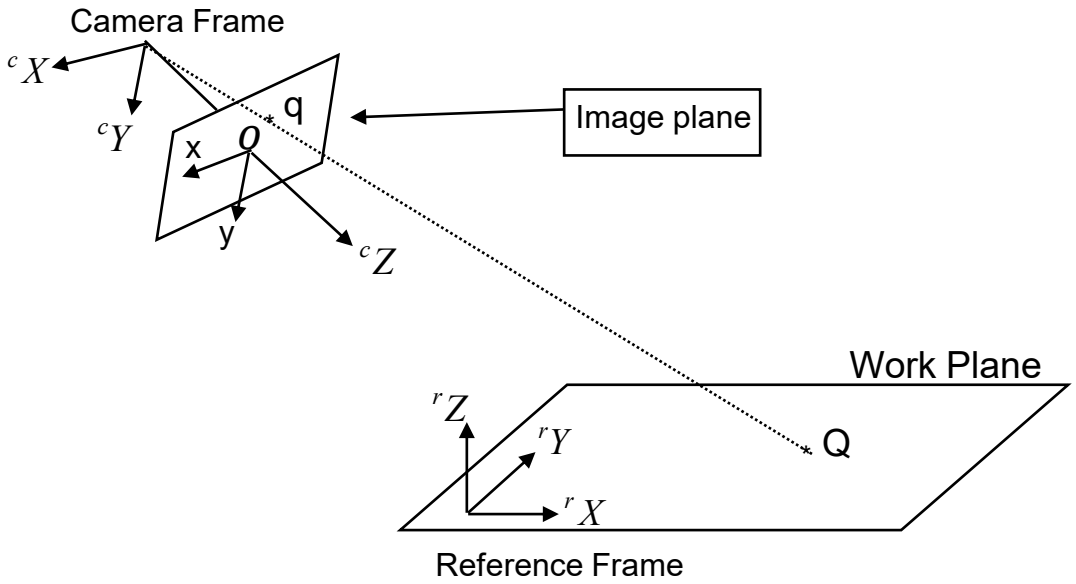
$$\frac{z_c}{f} = \frac{x_c}{x_d} \Rightarrow x_d = f\frac{x_c}{z_c}$$

$$\frac{z_c}{f} = \frac{y_c}{y_d} \Rightarrow y_d = f\frac{y_c}{z_c}$$



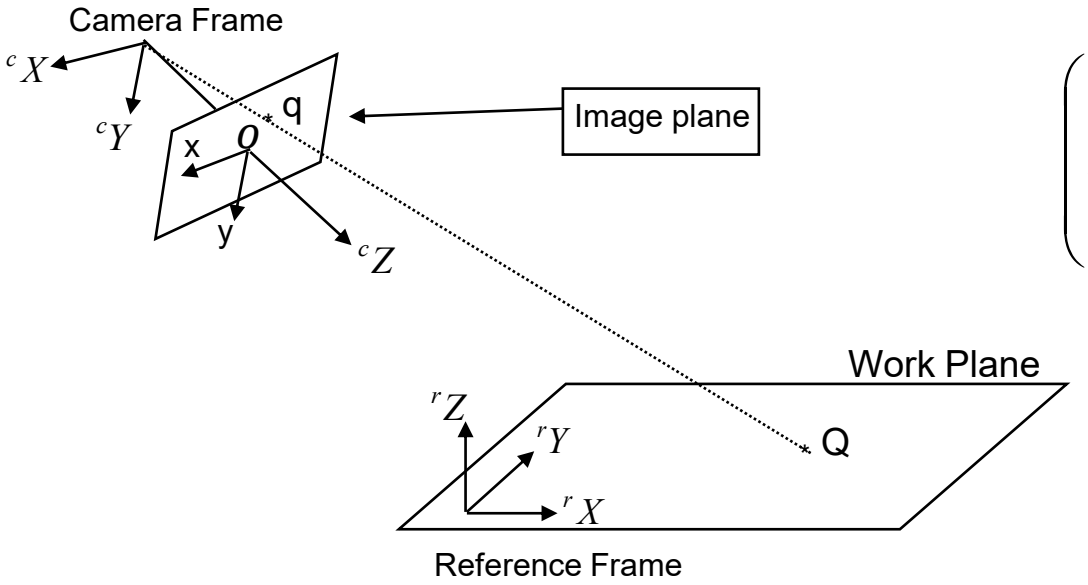
Projection to Image Plane …

$$x = f \bullet \frac{{}^{c}X}{{}^{c}Z} \qquad \text{and} \qquad y = f \bullet \frac{{}^{c}Y}{{}^{c}Z}$$



Matrix Equation of Projection ...

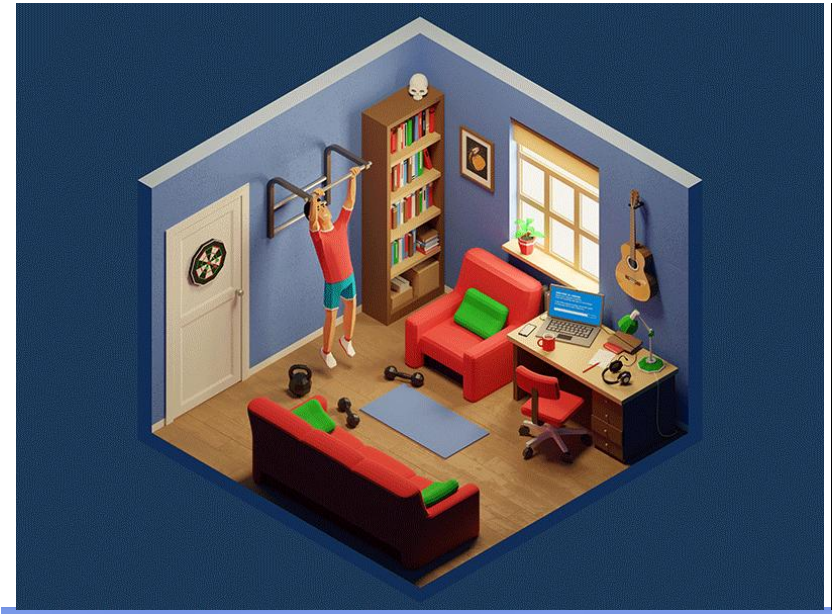
$$x = f \bullet \frac{{}^{c}X}{{}^{c}Z} \quad \text{and} \quad y = f \bullet \frac{{}^{c}Y}{{}^{c}Z}$$

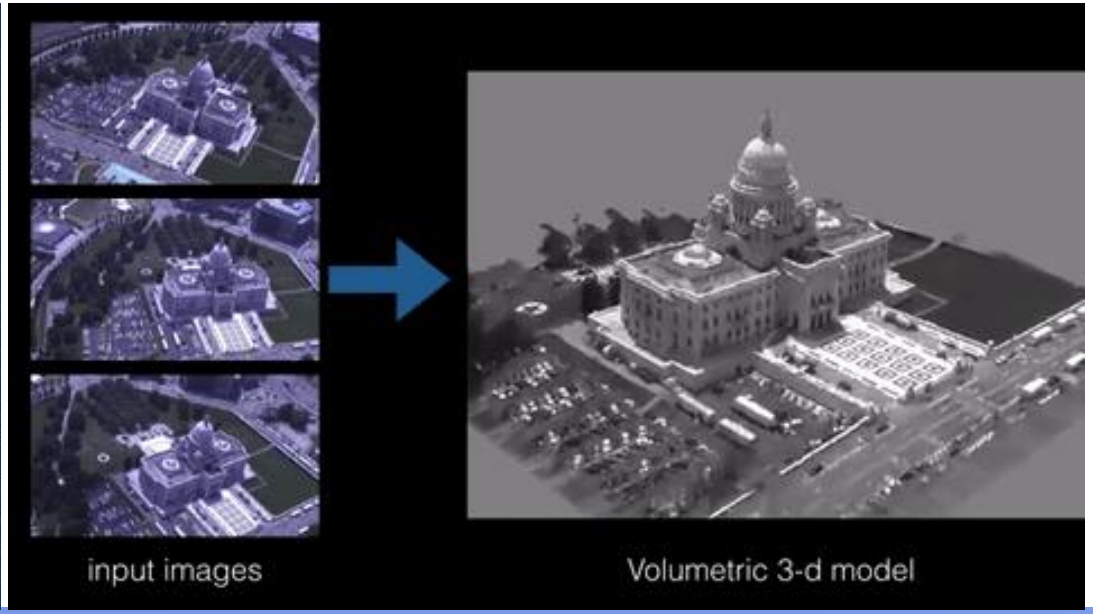


$$\begin{pmatrix} s \bullet x \\ s \bullet y \\ s \end{pmatrix} = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \bullet \begin{pmatrix} {}^{c}X \\ {}^{c}Y \\ {}^{c}Z \\ 1 \end{pmatrix}$$

Outline of Lecture 6

- Concept of Coordinate Transformation
- Transformation from Scene to Camera Space
- Transformation from Camera Space to Image Space
- Equation of Binocular Vision
- Calibration of Binocular Vision





Why do we have binocular vision?

- A binocular vision is a pair of two collaborative monocular visions, each of which could work independently. The collaboration between two monocular visions poses the difficult challenge of achieving stereo matching. - Xie Ming, 2004

Why to have two dependent eyes?

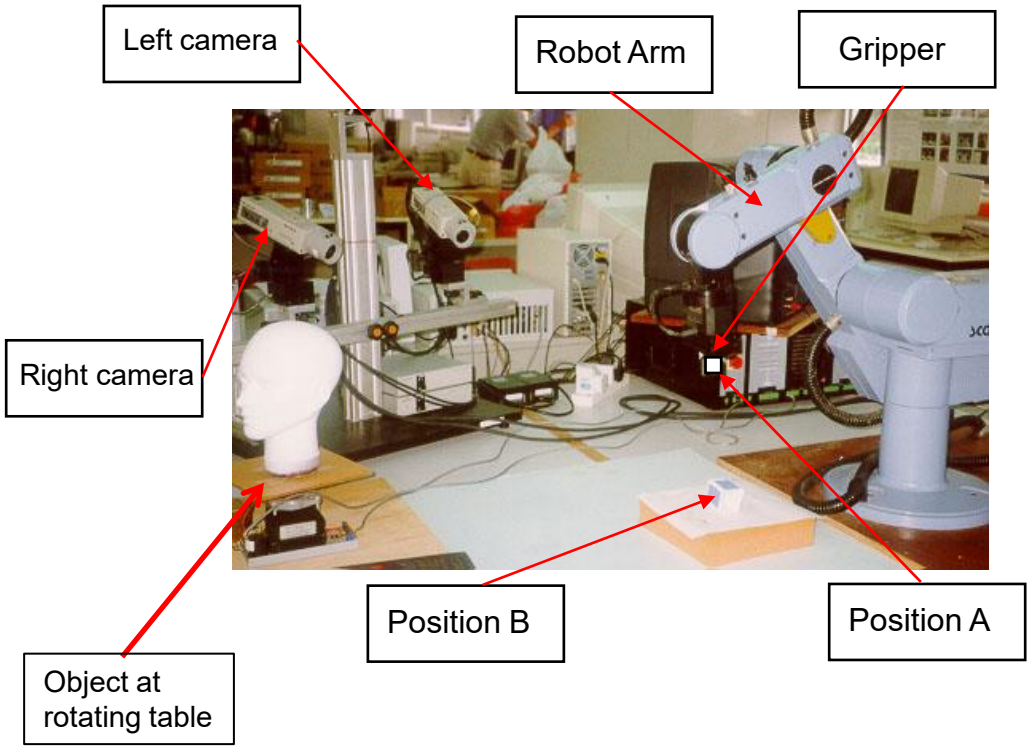
Why to have two dependent eyes?

Why to have two independent eyes?

Why to have two independent eyes?

Setup of Binocular Vision

- The Details of Hardware Support



You could set up binocular vision system at home with two laptops

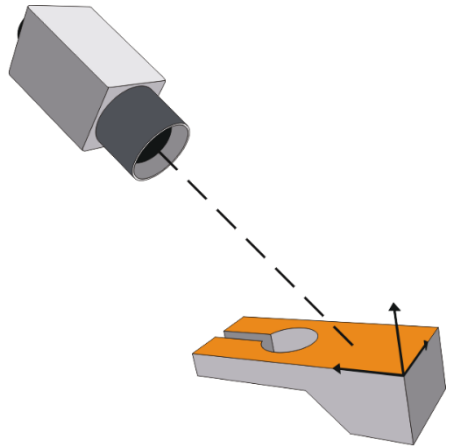
Equation of Camera's Forward Projection …

$$\begin{pmatrix} s \bullet u \\ s \bullet v \\ s \end{pmatrix} = \begin{pmatrix} \frac{f}{\Delta u} & 0 & u_0 & 0 \\ 0 & \frac{f}{\Delta v} & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} {}^{r}X \\ {}^{r}Y \\ {}^{r}Z \\ 1 \end{pmatrix}$$

Camera's Forward Projection Matrix

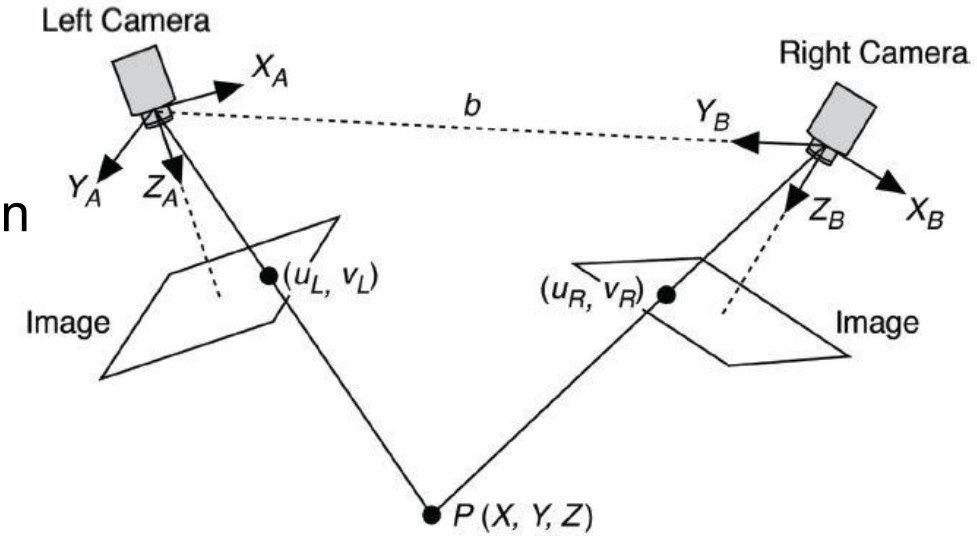
$$\begin{pmatrix} s \bullet u \\ s \bullet v \\ s \end{pmatrix} = C_{3\times 4} \bullet \begin{pmatrix} {}^{r}X \\ {}^{r}Y \\ {}^{r}Z \\ 1 \end{pmatrix}$$

Equation of Camera's Forward Projection



Forward Reconstruction Method in Binocular Vision

- Working Principle:
  - The geometry of physical entities in a 3D space depends on the coordinates of points.
  - The measurement of coordinates of points in a 3D space can be achieved with stereovision or binocular vision system.
  - A binocular vision can see a point P and will output two image points.
  - The coordinates of the two images allow a machine to uniquely determine the coordinates of point P.



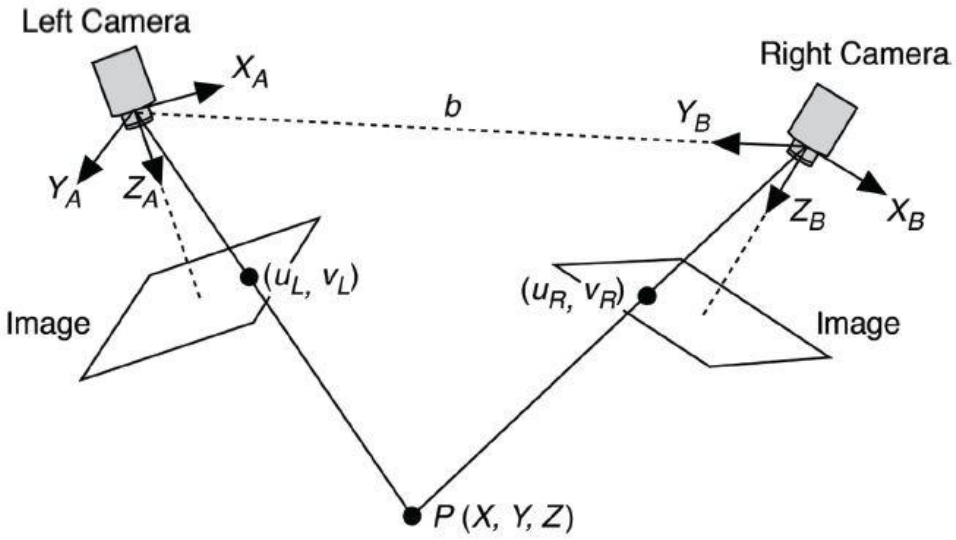
Search Space For Stereo Matching

$$[u_l \quad v_l \quad 1]\begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix}\begin{bmatrix} u_r \\ v_r \\ 1 \end{bmatrix} = 0$$

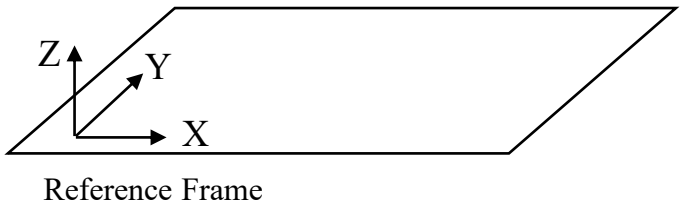
Transformation from Scene to Left Camera …

- The equation of left camera's forward projection could be written into a system of equations:

$$\begin{pmatrix} s_l \bullet u_l \\ s_l \bullet v_l \\ s_l \end{pmatrix} = \begin{bmatrix} c_{l1} & c_{l2} & c_{l3} & c_{l4} \\ c_{l5} & c_{l6} & c_{l7} & c_{l8} \\ c_{l9} & c_{l10} & c_{l11} & 1 \end{bmatrix} \bullet \begin{pmatrix} {}^rX \\ {}^rY \\ {}^rZ \\ 1 \end{pmatrix}$$



$$\begin{cases} s_l \bullet u_l = c_{l1} \bullet {}^rX + c_{l2} \bullet {}^rY + c_{l3} \bullet {}^rZ + c_{l4} \\ s_l \bullet v_l = c_{l5} \bullet {}^rX + c_{l6} \bullet {}^rY + c_{l7} \bullet {}^rZ + c_{l8} \\ s_l = c_{l9} \bullet {}^rX + c_{l10} \bullet {}^rY + c_{l11} \bullet {}^rZ + 1 \end{cases}$$

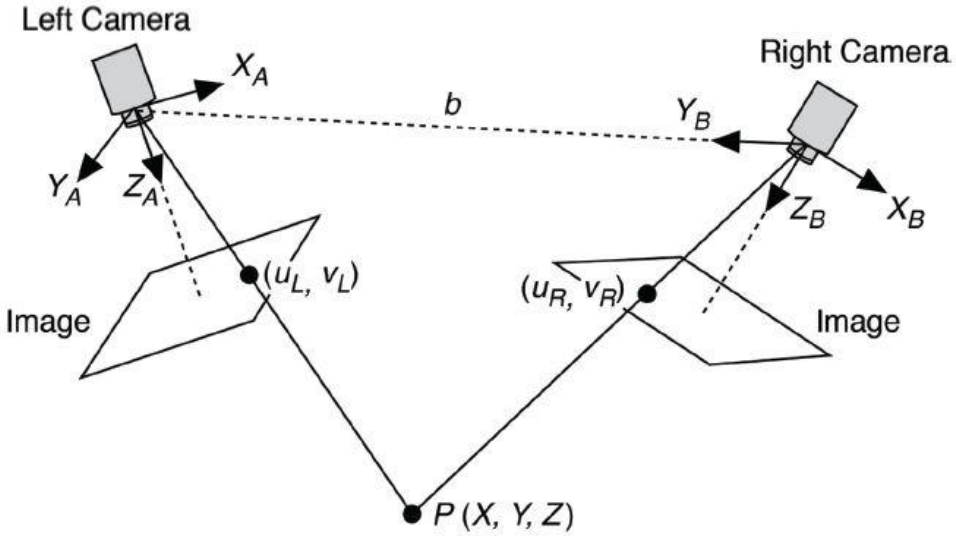


$$\begin{cases} (c_{l1} - c_{l9} \bullet u_l) \bullet {}^rX + (c_{l2} - c_{l10} \bullet u_l) \bullet {}^rY + (c_{l3} - c_{l11} \bullet u_l) \bullet {}^rZ + c_{l4} = u_l \\ (c_{l5} - c_{l9} \bullet v_l) \bullet {}^rX + (c_{l6} - c_{l10} \bullet v_l) \bullet {}^rY + (c_{l7} - c_{l11} \bullet v_l) \bullet {}^rZ + c_{l8} = v_l \end{cases}$$

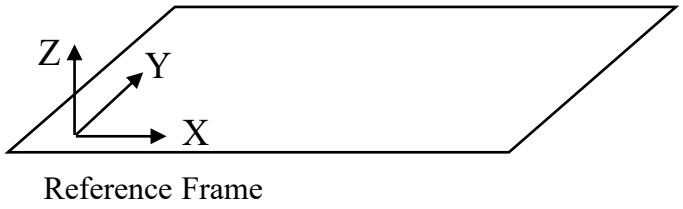
Transformation from Scene to Right Camera …

- The equation of right camera's forward projection could be written into a system of equations:

$$\begin{pmatrix} s_r \bullet u_r \\ s_r \bullet v_r \\ s_r \end{pmatrix} = \begin{bmatrix} c_{r1} & c_{r2} & c_{r3} & c_{r4} \\ c_{r5} & c_{r6} & c_{r7} & c_{r8} \\ c_{r9} & c_{r10} & c_{r11} & 1 \end{bmatrix} \bullet \begin{pmatrix} {}^rX \\ {}^rY \\ {}^rZ \\ 1 \end{pmatrix}$$



$$\begin{cases} s_r \bullet u_r = c_{r1} \bullet {}^rX + c_{r2} \bullet {}^rY + c_{r3} \bullet {}^rZ + c_{r4} \\ s_r \bullet v_r = c_{r5} \bullet {}^rX + c_{r6} \bullet {}^rY + c_{r7} \bullet {}^rZ + c_{r8} \\ s_r = c_{r9} \bullet {}^rX + c_{r10} \bullet {}^rY + c_{r11} \bullet {}^rZ + 1 \end{cases}$$



$$\begin{cases} (c_{r1} - c_{r9} \bullet u_r) \bullet {}^rX + (c_{r2} - c_{r10} \bullet u_r) \bullet {}^rY + (c_{r3} - c_{r11} \bullet u_r) \bullet {}^rZ + c_{r4} = u_r \\ (c_{r5} - c_{r9} \bullet v_r) \bullet {}^rX + (c_{r6} - c_{r10} \bullet v_r) \bullet {}^rY + (c_{r7} - c_{r11} \bullet v_r) \bullet {}^rZ + c_{r8} = v_r \end{cases}$$

A System of Equations in Binocular Vision

- The equations from both cameras could be combined into the following system of equations:

$$\begin{cases} (c_{l1} - c_{l9} \bullet u_l) \bullet {}^rX + (c_{l2} - c_{l10} \bullet u_l) \bullet {}^rY + (c_{l3} - c_{l11} \bullet u_l) \bullet {}^rZ + c_{l4} = u_l \\ (c_{l5} - c_{l9} \bullet v_l) \bullet {}^rX + (c_{l6} - c_{l10} \bullet v_l) \bullet {}^rY + (c_{l7} - c_{l11} \bullet v_l) \bullet {}^rZ + c_{l8} = v_l \\ (c_{r1} - c_{r9} \bullet u_r) \bullet {}^rX + (c_{r2} - c_{r10} \bullet u_r) \bullet {}^rY + (c_{r3} - c_{r11} \bullet u_r) \bullet {}^rZ + c_{r4} = u_r \\ (c_{r5} - c_{r9} \bullet v_r) \bullet {}^rX + (c_{r6} - c_{r10} \bullet v_r) \bullet {}^rY + (c_{r7} - c_{r11} \bullet v_r) \bullet {}^rZ + c_{r8} = v_r \end{cases}$$

We define:

$$A = \begin{pmatrix} (c_{l1} - c_{l9} \bullet u_l) & (c_{l2} - c_{l10} \bullet u_l) & (c_{l3} - c_{l11} \bullet u_l) \\ (c_{l5} - c_{l9} \bullet v_l) & (c_{l6} - c_{l10} \bullet v_l) & (c_{l7} - c_{l11} \bullet v_l) \\ (c_{r1} - c_{r9} \bullet u_r) & (c_{r2} - c_{r10} \bullet u_r) & (c_{r3} - c_{r11} \bullet u_r) \\ (c_{r5} - c_{r9} \bullet v_r) & (c_{r6} - c_{r10} \bullet v_r) & (c_{r7} - c_{r11} \bullet v_r) \end{pmatrix} \qquad B = \begin{pmatrix} u_l - c_{l4} \\ v_l - c_{l8} \\ u_r - c_{r4} \\ v_r - c_{r8} \end{pmatrix}$$

Solution of Forward Reconstruction

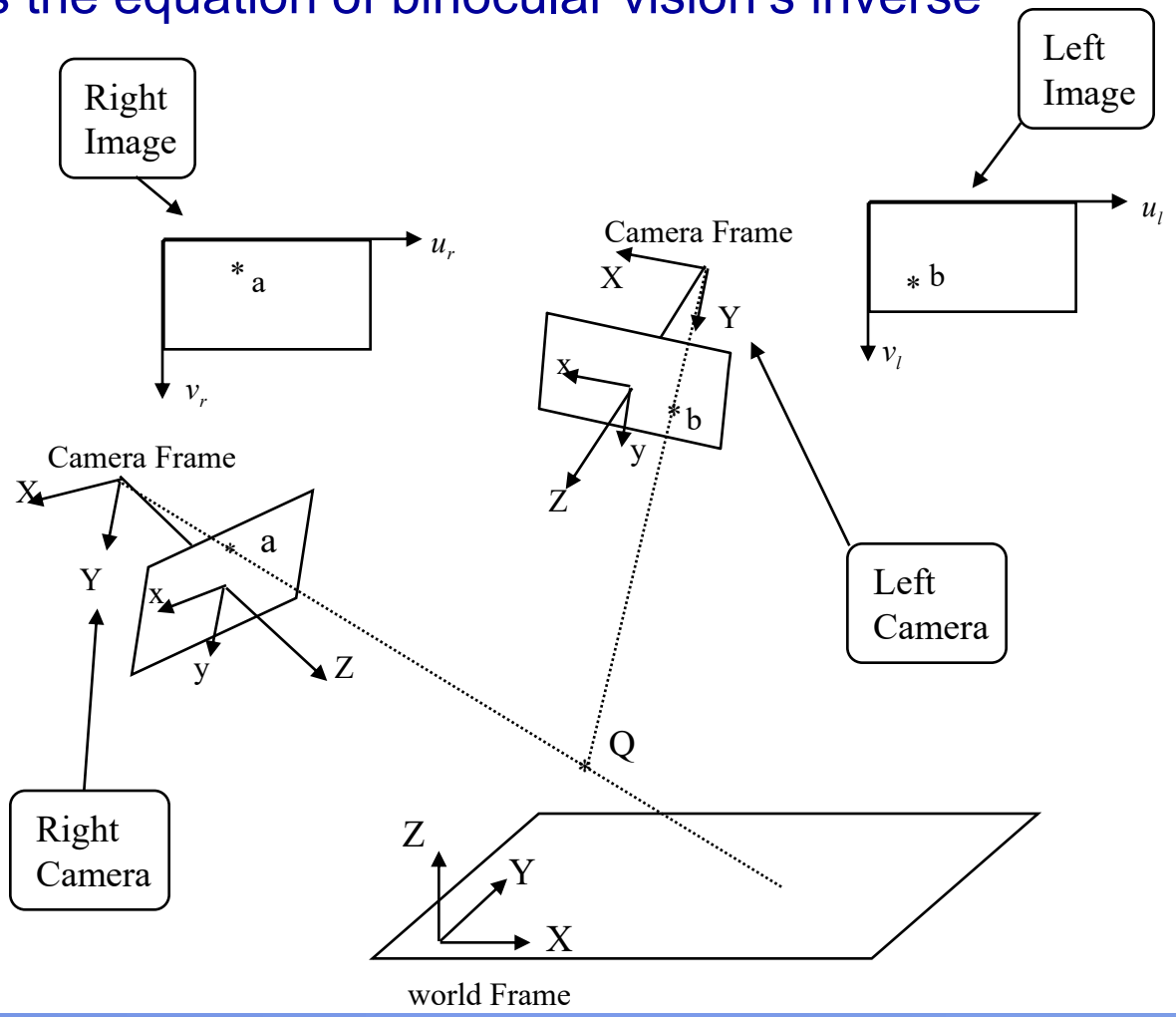
- The previous system of equations could be written into a matrix form. The pseudo-inverse yields the equation of binocular vision's inverse projection:

$$A \bullet \begin{pmatrix} {}^{r}X \\ {}^{r}Y \\ {}^{r}Z \end{pmatrix} = B$$

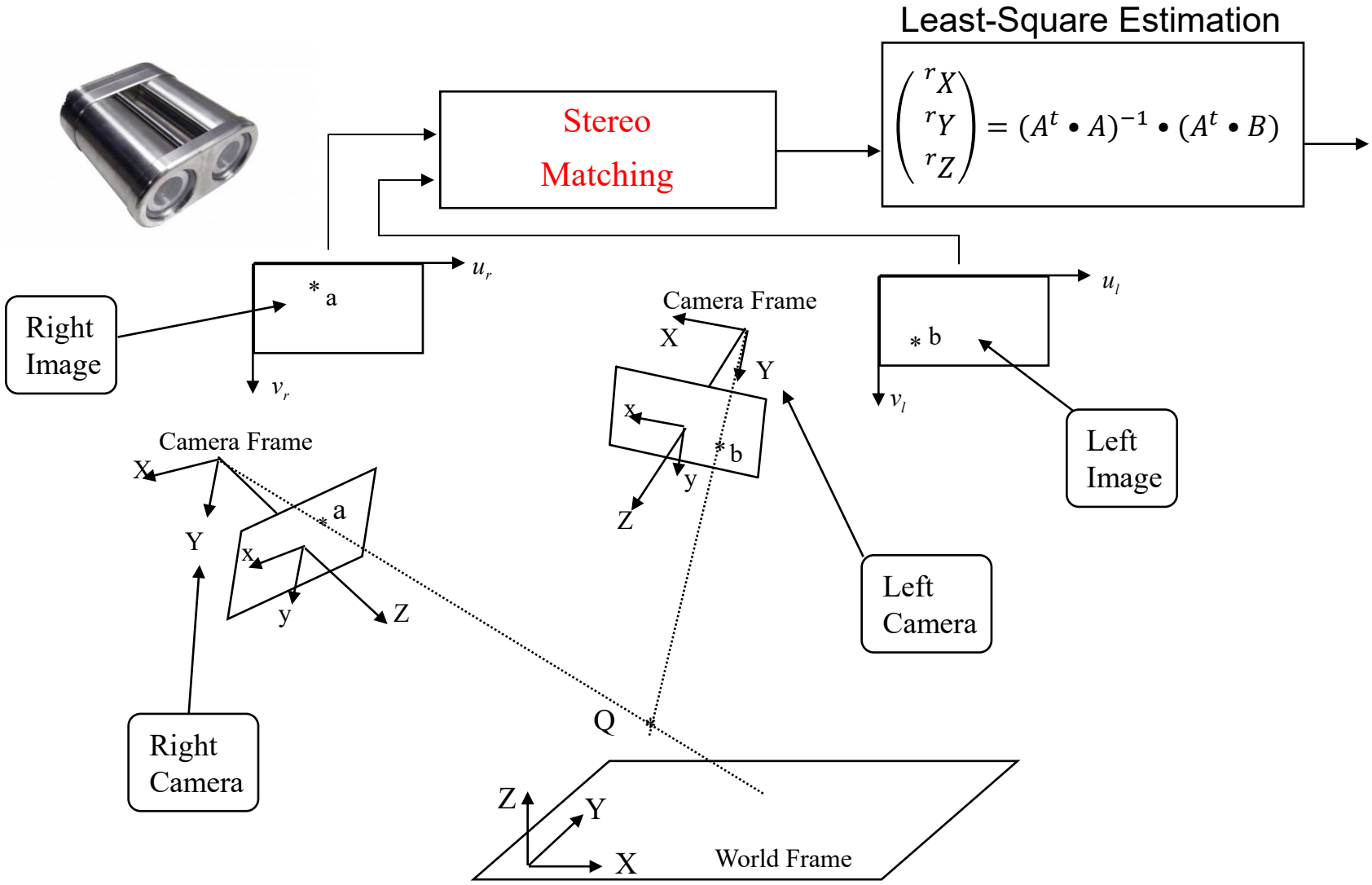
$$(A^t \bullet A) \bullet \begin{pmatrix} {}^{r}X \\ {}^{r}Y \\ {}^{r}Z \end{pmatrix} = A^t \bullet B$$

$$\begin{pmatrix} {}^{r}X \\ {}^{r}Y \\ {}^{r}Z \end{pmatrix} = (A^t \bullet A)^{-1} \bullet (A^t \bullet B)$$

Least-Square Estimation

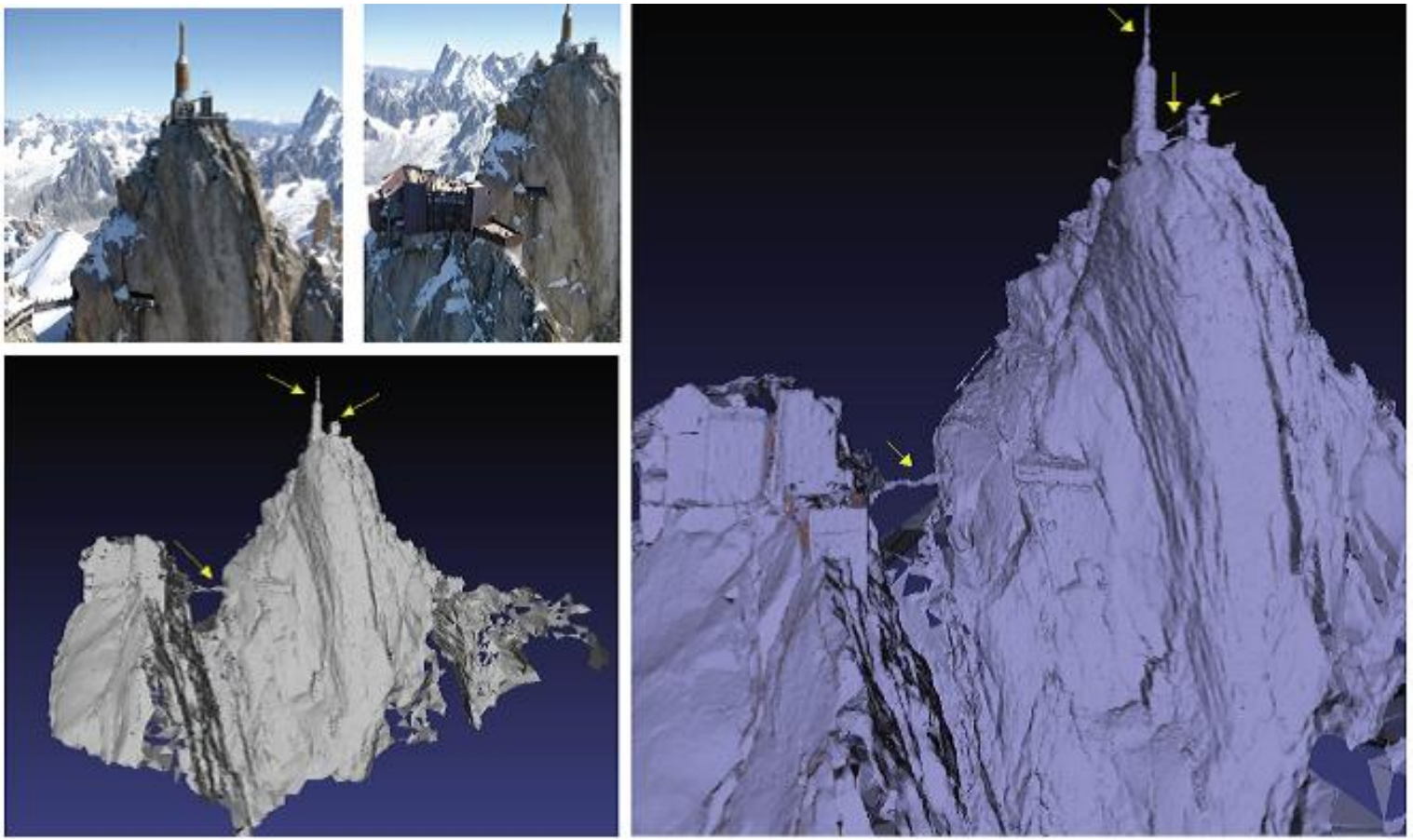


Summary of Forward Reconstruction Method



Results with Pixels in General

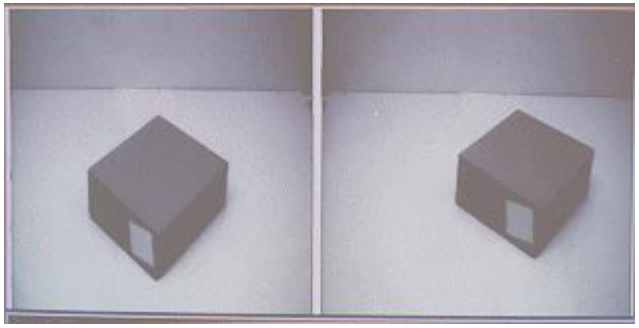
- Experimental Results



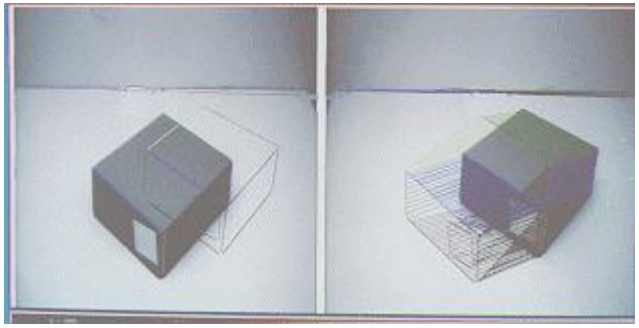
Results with Line Segments

- More Experimental Results

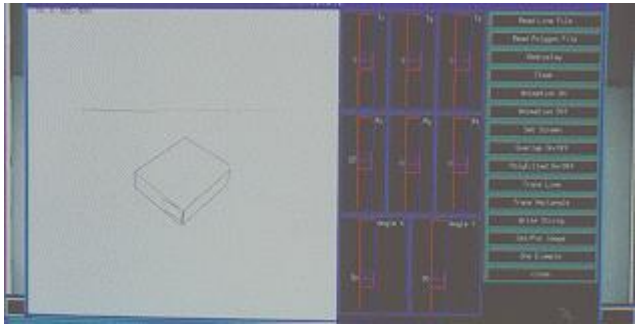
One pair of stereo images



Result of stereo matching



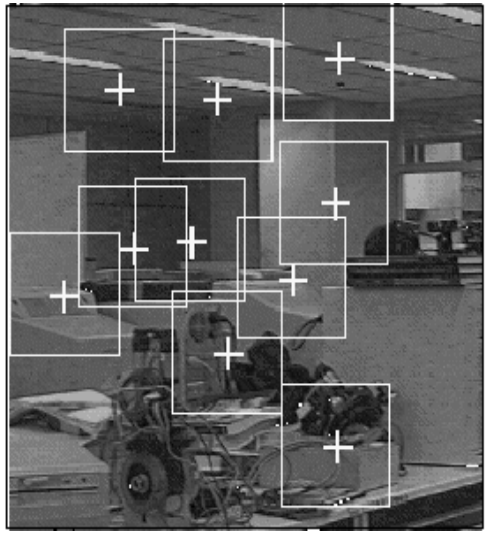
3D view of reconstructed line segments

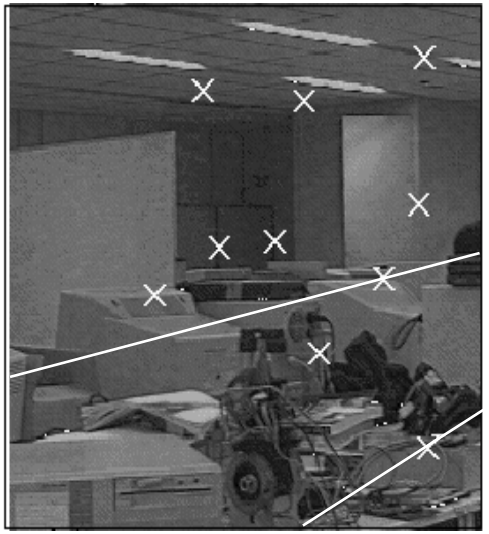


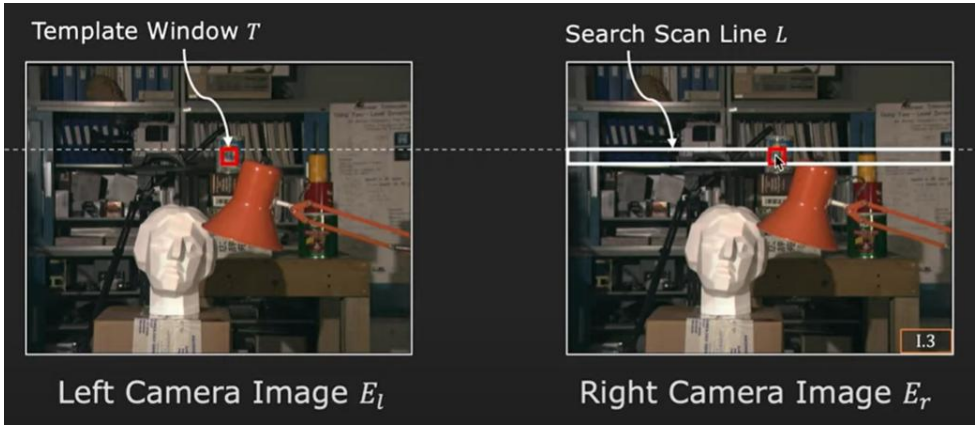
Open Issue of Stereo Matching

- How to find the match inside the right camera, given a pixel inside the left camera?

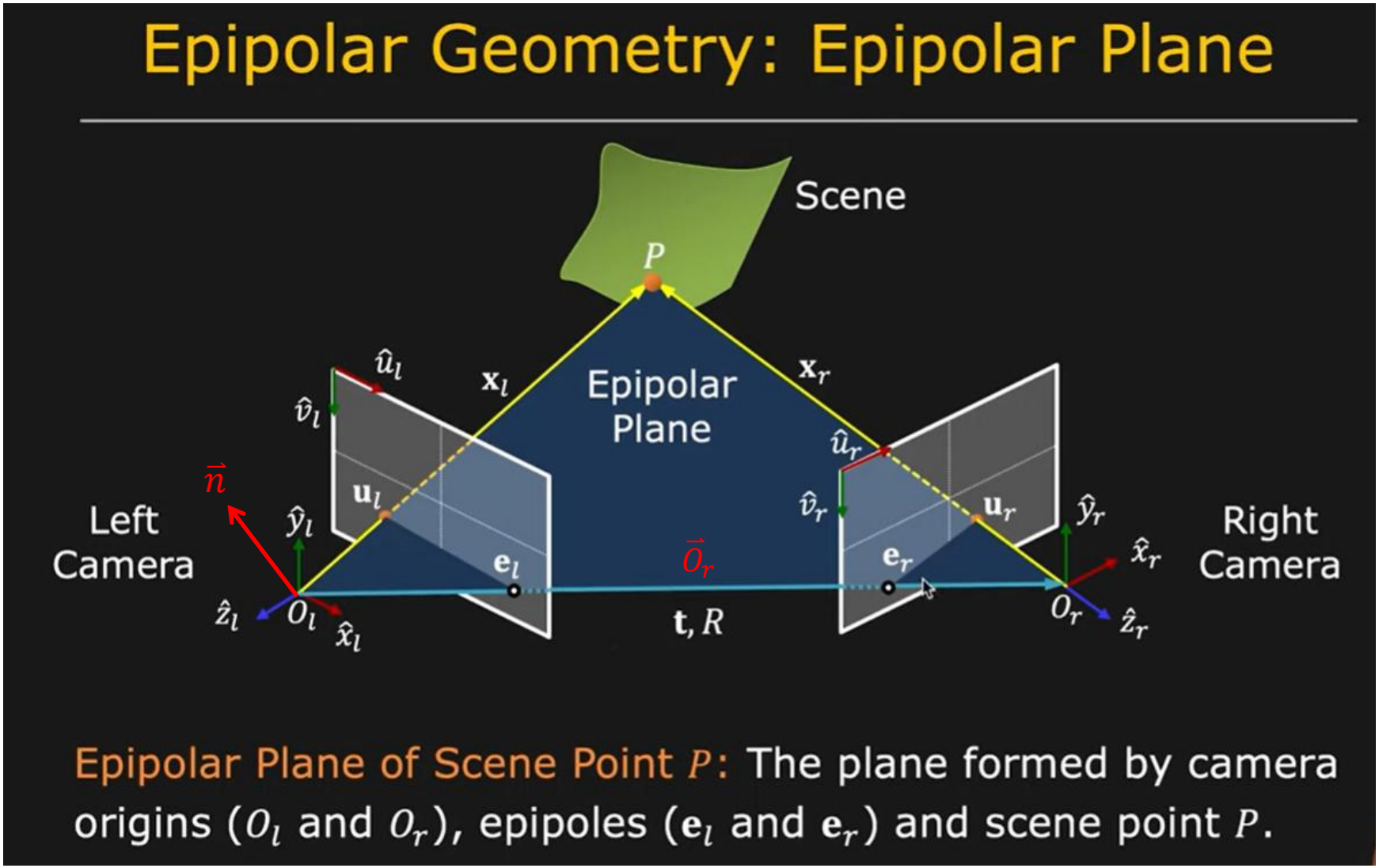
- Definition of Epipolar Line: Given a point in the left image, there is a straight line in the right image, which contains all the possible points to be matched with the given point in the left image. Such a line is called as an Epipolar line.







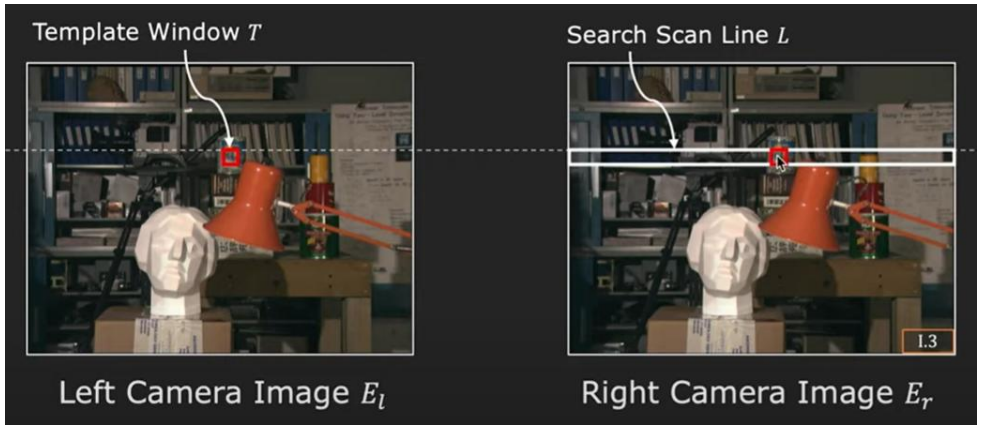
About Epipolar Plane and Epipoles …

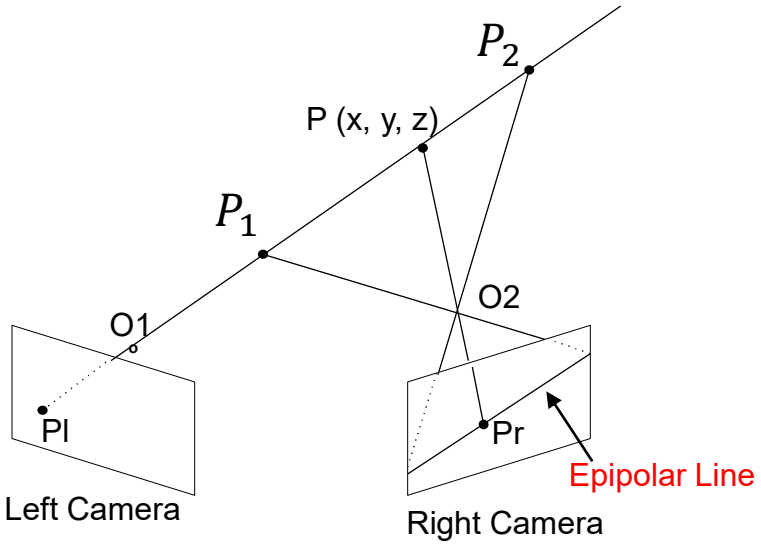


More About Equation of Epipolar Line:

- Choose any pair of two possible value of Z coordinate: (Z1, Z2), which correspond to two 3D points (P1, P2).
- Compute the 3D coordinates of (P1, P2) by using the knowledge about the image coordinates in left camera.
- Project these two locations (P1, P2) onto the image plane of right camera.
- Compute the equation of epipolar line on image plane of right camera.

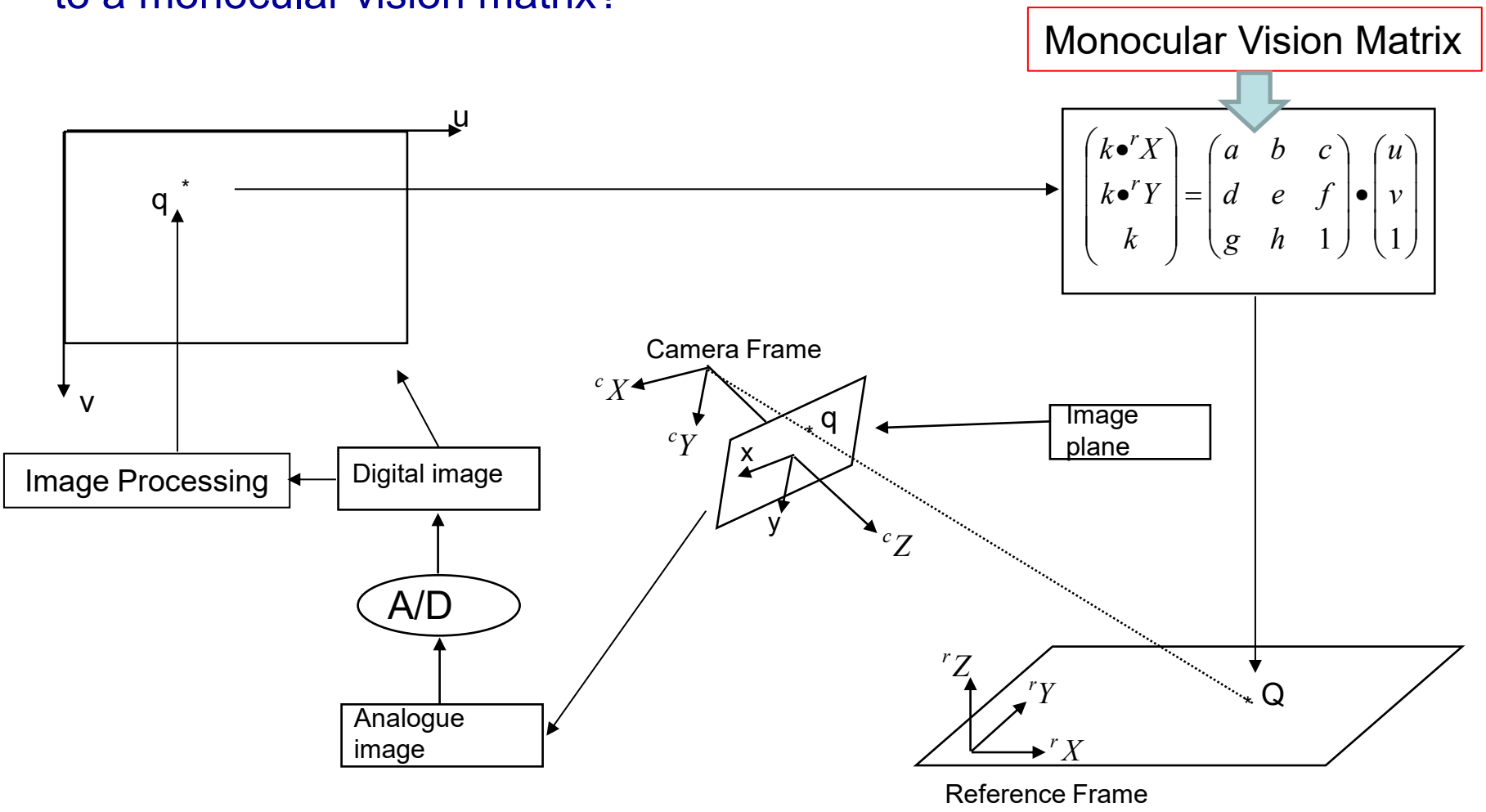
$$[u_l \quad v_l \quad 1]\begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix}\begin{bmatrix} u_r \\ v_r \\ 1 \end{bmatrix} = 0$$





New Method: Projective Solution in Binocular Vision

- Question: Could we obtain a binocular vision matrix which is similar to a monocular vision matrix?



New Method: Projective Solution in Binocular Vision

- The answer is Yes!

Stereo Matching

Binocular Vision Matrix

$$\begin{bmatrix} k\,{}^{r}X \\ k\,{}^{r}Y \\ k\,{}^{r}Z \\ k \end{bmatrix} = B_{4\times5} \begin{bmatrix} u_l \\ v_l \\ u_r \\ v_r \\ 1 \end{bmatrix}$$

Right Image

Left Image

Camera Frame

Right Camera

Left Camera

World Frame

Q

Global Impact

Very Important Discovery

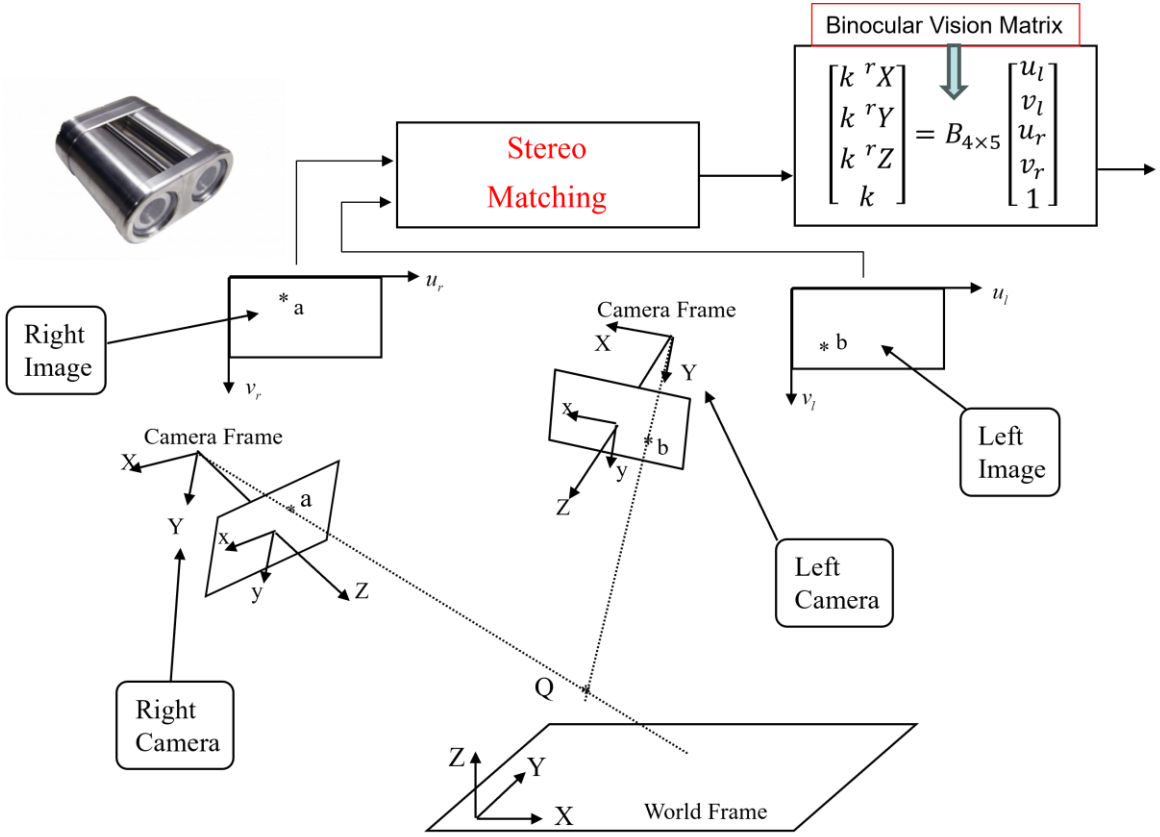
Xie M., Fang Yuhui and Lai Tingfeng, 2025, New Solution to 3D Projection in Human-like Binocular Vision, International Journal of Humanoid Robotics.

Nobel Prize?

New Method: Projective Solution in Binocular Vision

- Reference:

Xie M., **Fang Yuhui and **Lai Tingfeng, 2025, New Solution to 3D Projection in Human-like Binocular Vision, International Journal of Humanoid Robotics,

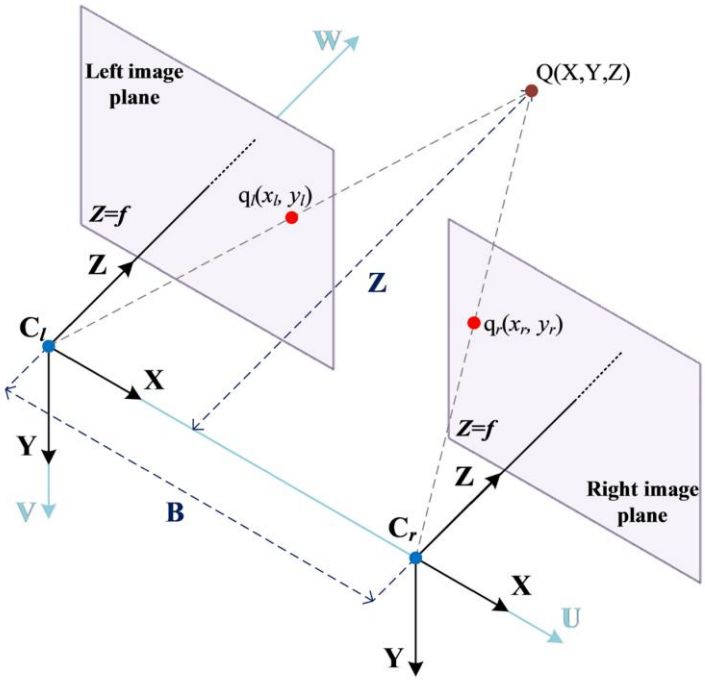


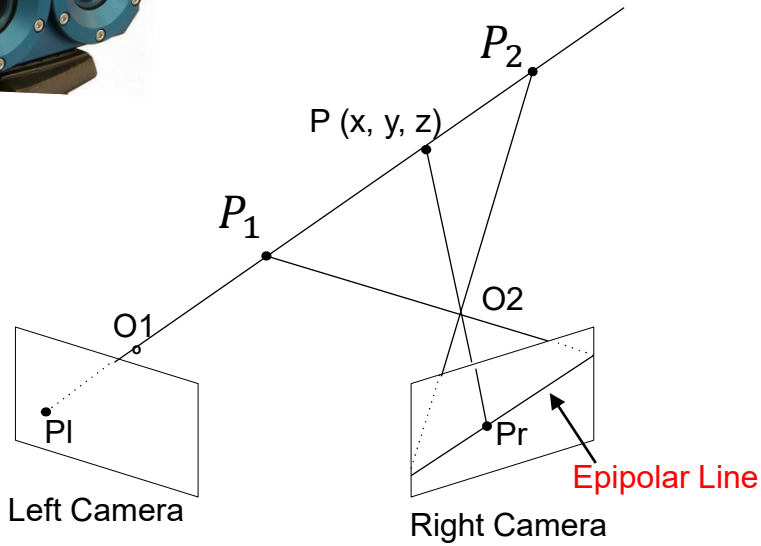
What are the advantages of using the projective solution in binocular vision?

- Accurate
- Differentiable
- Able to Directly Compute Displacements

Xie M. (2003). *Fundamentals of Robotics: Linking Perception to Action*, *World Scientific Publishing Co.*

Global Impact





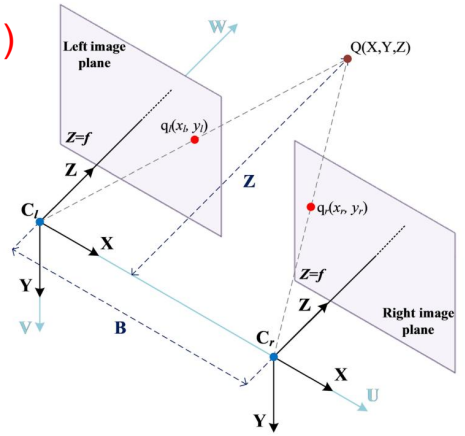
Other Method (1)

Inverse Reconstruction Method

Global Impact

(Award Winning Research)

Zhang Y. and **Xie M.**, 1998, New Principle for Passive 3D Scanner, Third World Automation Congress (ISOMA-043), United States, TSI Press, pp. 1-6.



3D Discrete Scene :

$$\{(X_i, Y_i, Z_i), i = 0,1,2,\ldots\}$$

with :

$$Z_i = Z_{\min} + i \bullet \Delta Z$$

Meaningful Image Data

Left Image :

$$\{(u_{lj}, v_{lj}), j = 0,1,2,\ldots\}$$

Right Image:

$$\{(u_{ri}, v_{ri}), i = 0,1,2,\ldots\}$$

Stereo Matching

Left Camera :

$$\begin{pmatrix} \rho \bullet u_{lj} \\ \rho \bullet v_{lj} \\ \rho \end{pmatrix} = M_l \bullet \begin{pmatrix} X_i \\ Y_i \\ Z_{\min} + i \bullet \Delta Z \\ 1 \end{pmatrix}$$

(i=0,2,3,…)

$$(X_i, Y_i, Z_i)$$

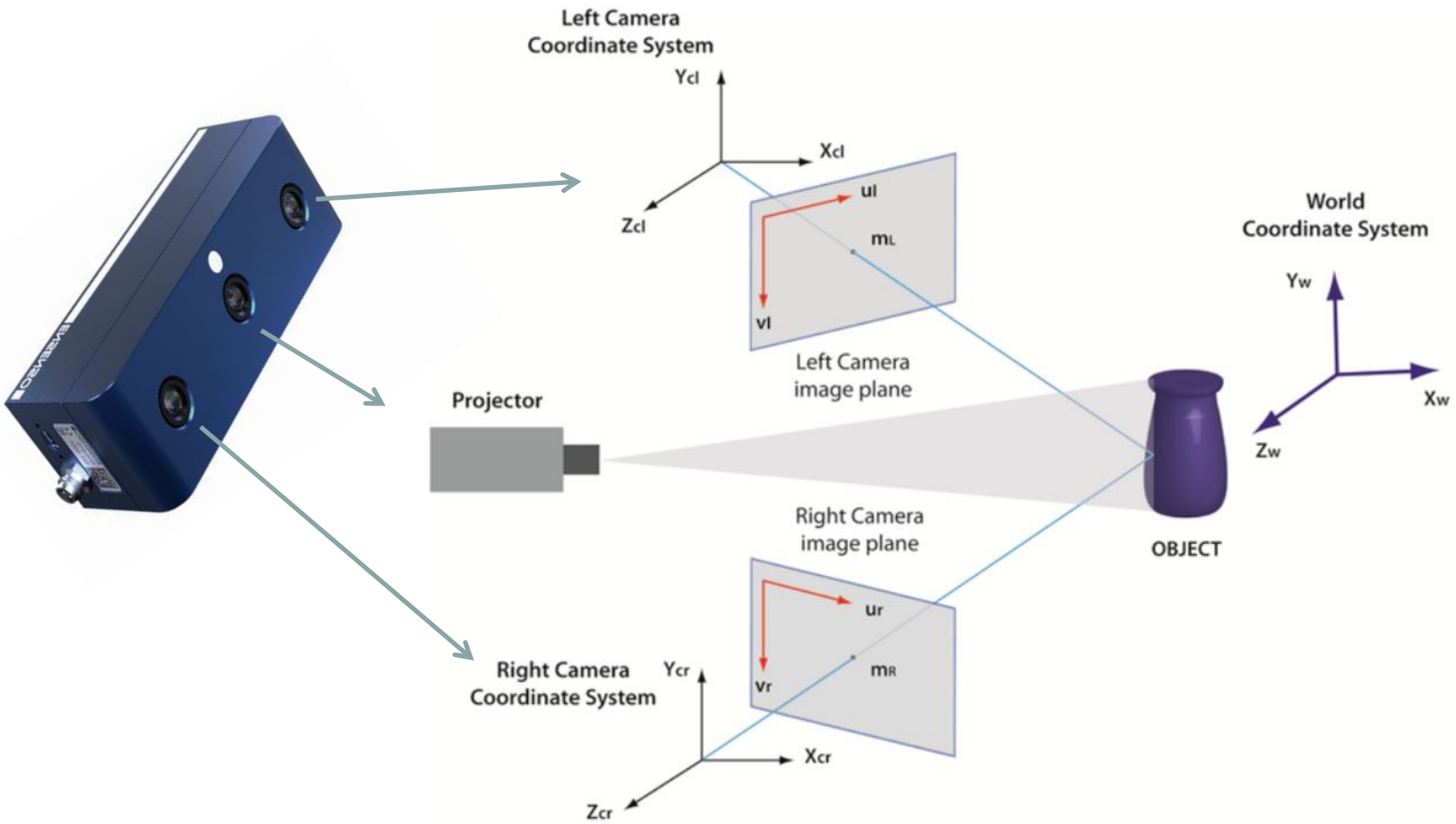
Right Camera:

$$\begin{pmatrix} s \bullet u_{ri} \\ s \bullet v_{ri} \\ s \end{pmatrix} = C_r \bullet \begin{pmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{pmatrix}$$

(i=0,2,3,…)

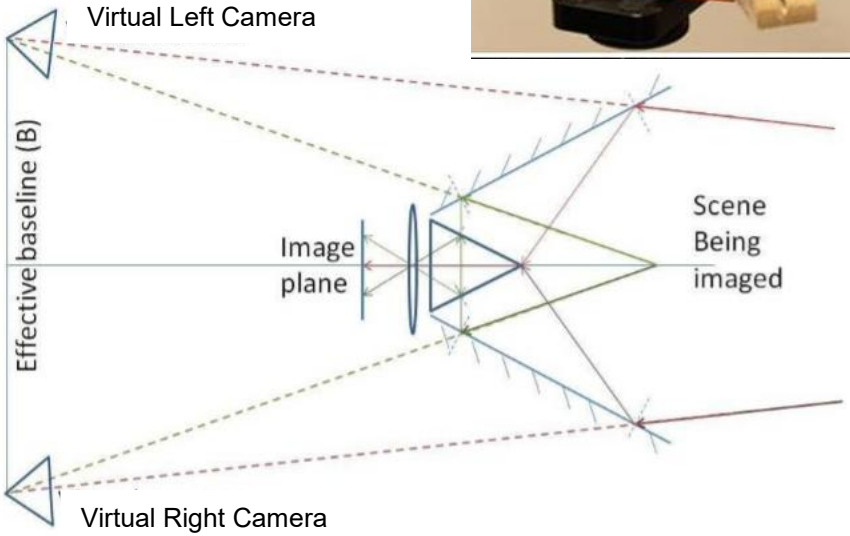
Output

Other Method (2) …



Other Method (3) …

(a)

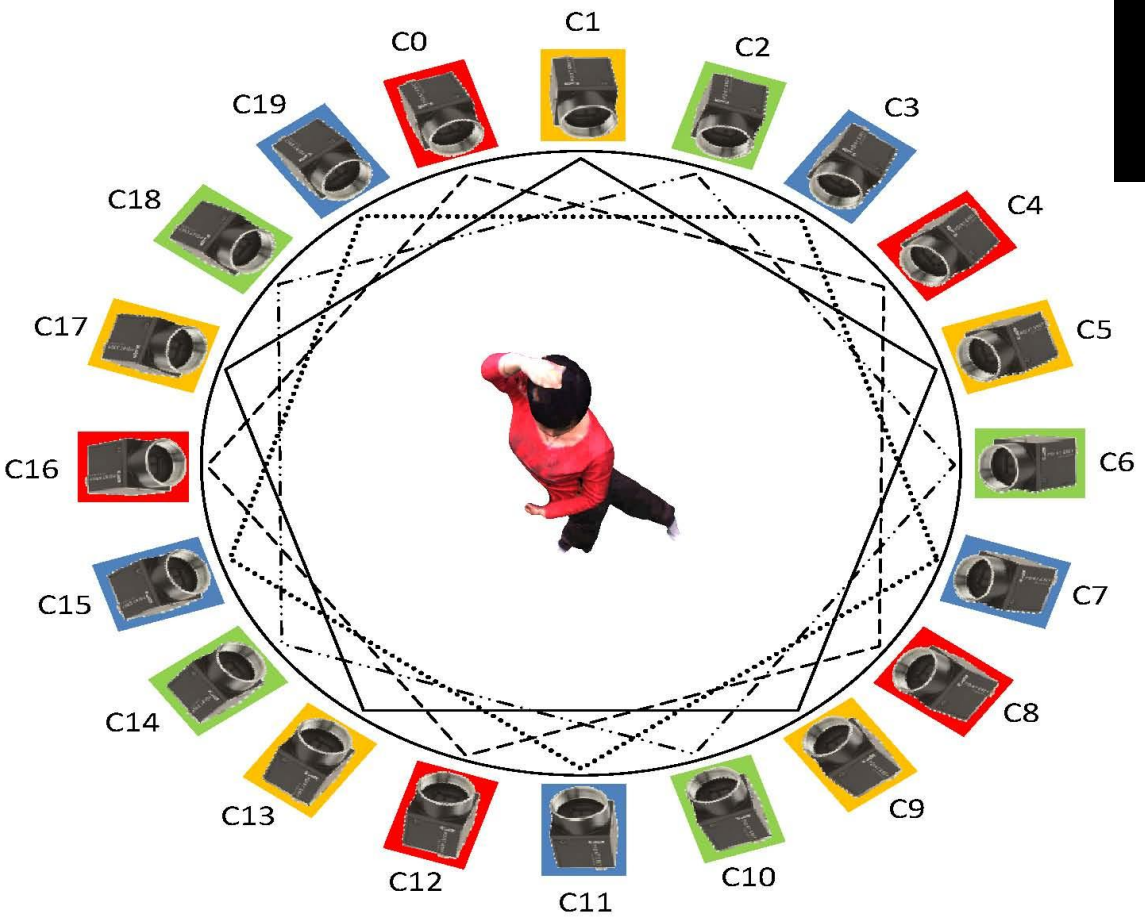


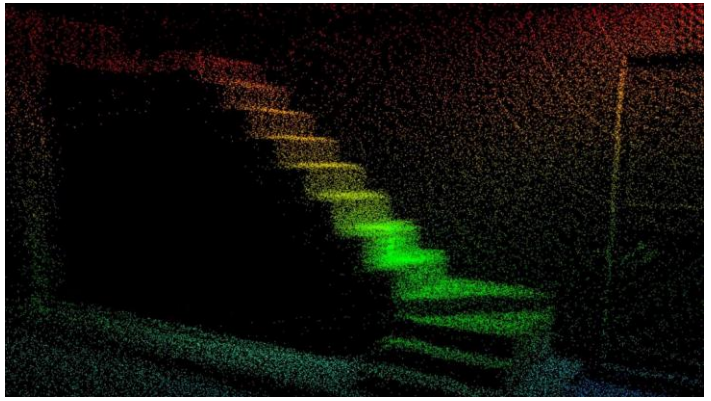
(b)

Raw image captured by the setup. The stereo images are extracted as two halves of the raw image.

Ray diagram illustrating the formation of the two images, the virtual cameras and baseline.

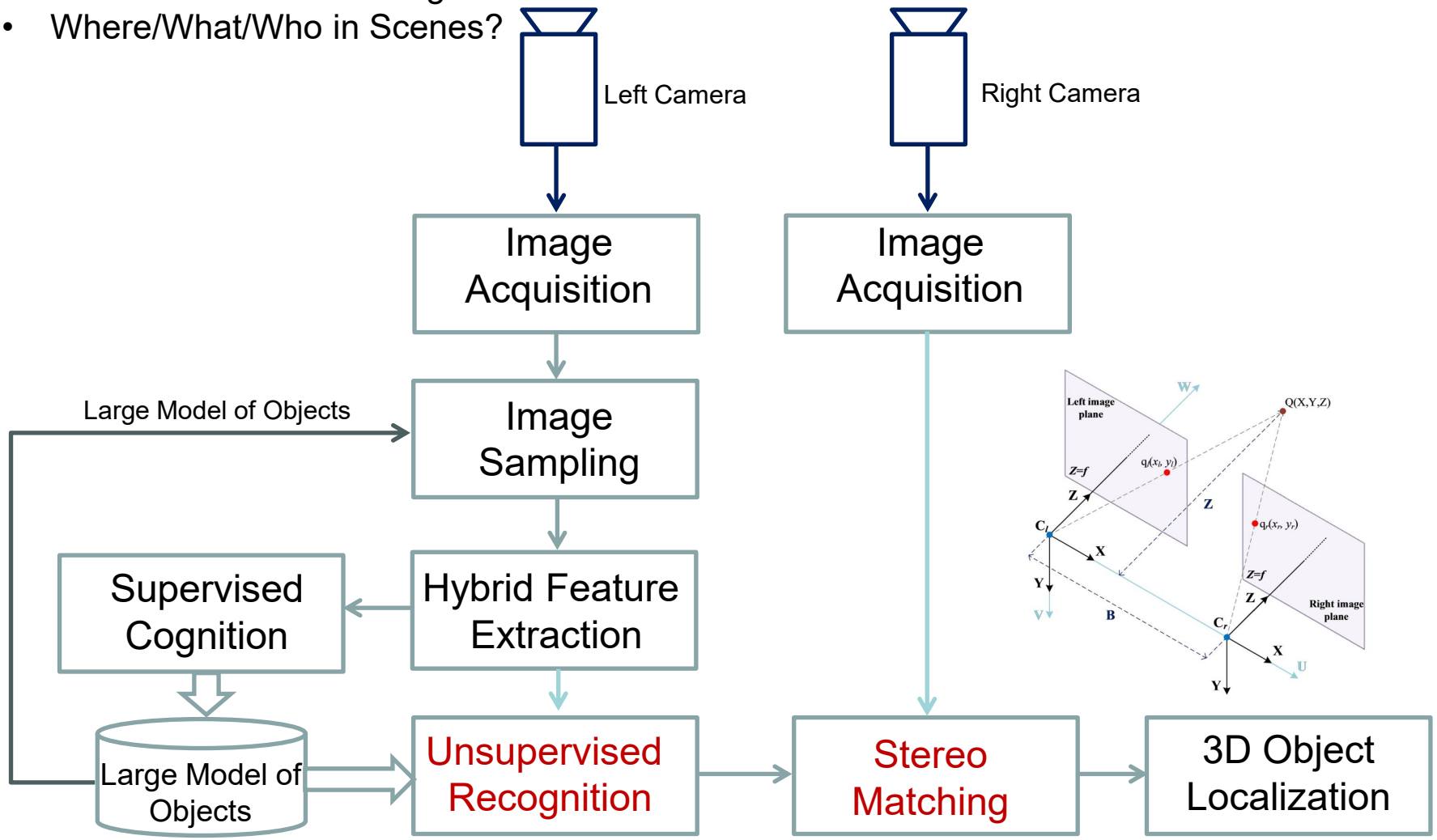
More Methods …





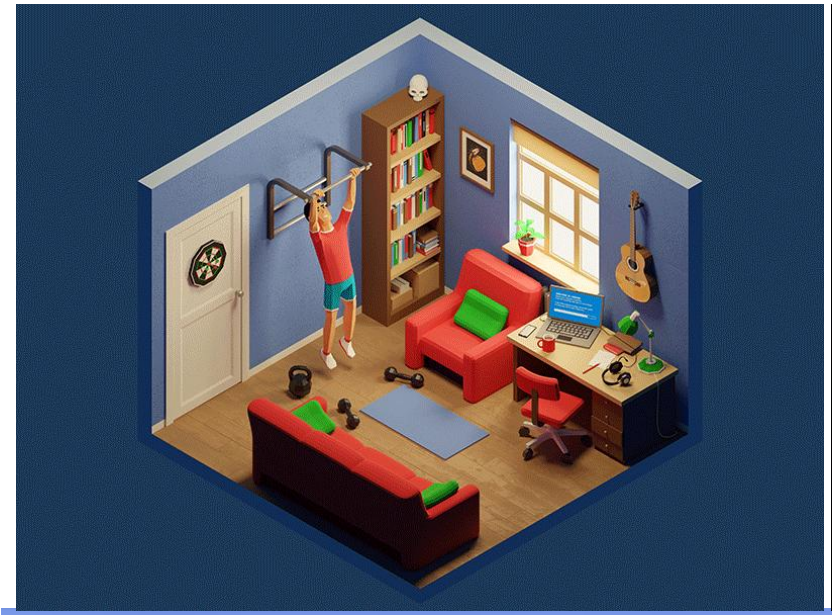
Future Research:
Solution Toward Achieving Human-like Binocular Vision

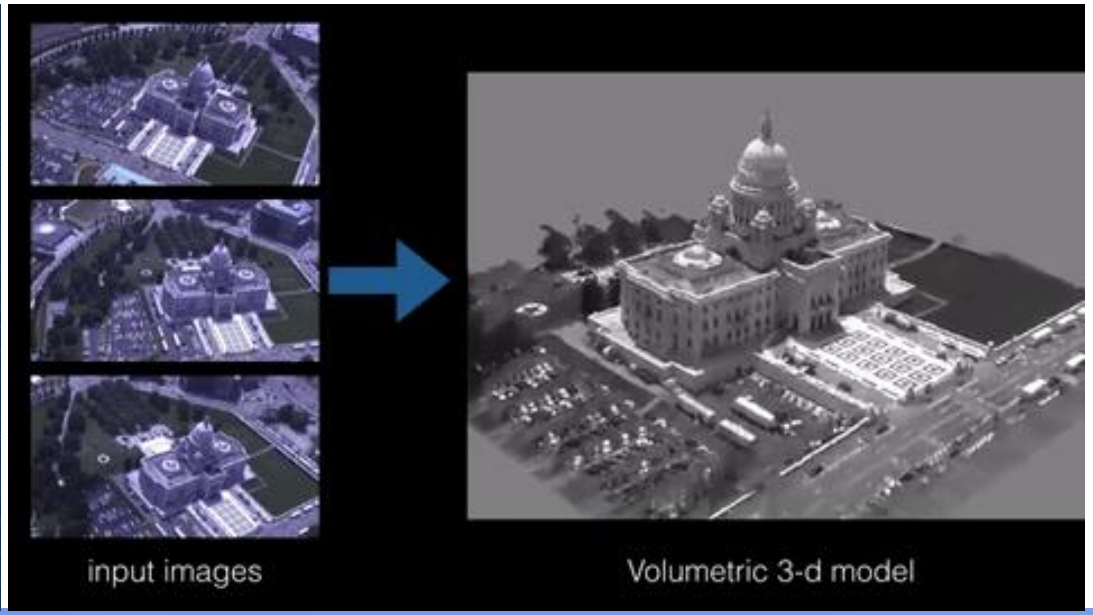
- Where/What/Who in Images?
- Where/What/Who in Scenes?



Outline of Lecture 6

- Concept of Coordinate Transformation
- Transformation from Scene to Camera Space
- Transformation from Camera Space to Image Space
- Equation of Binocular Vision
- Calibration of Binocular Vision

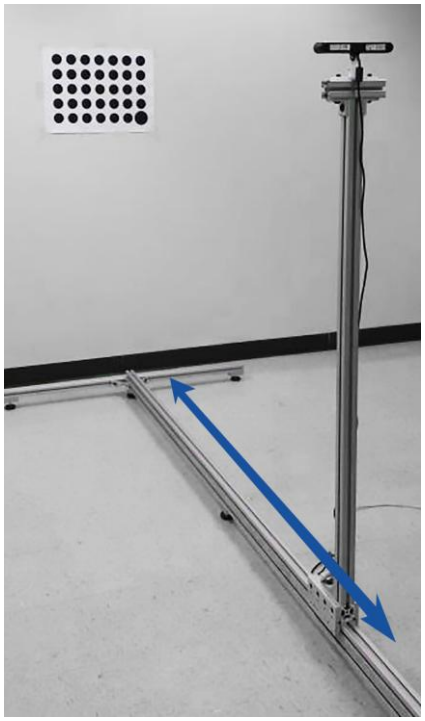




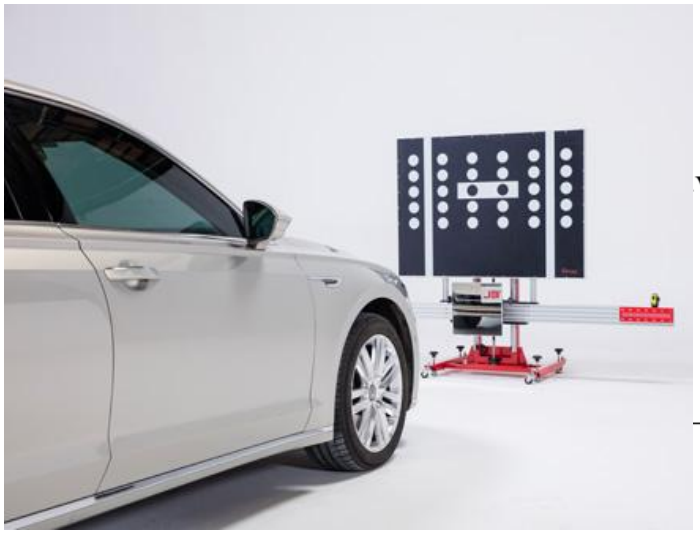
How to calibrate individual cameras in binocular vision?

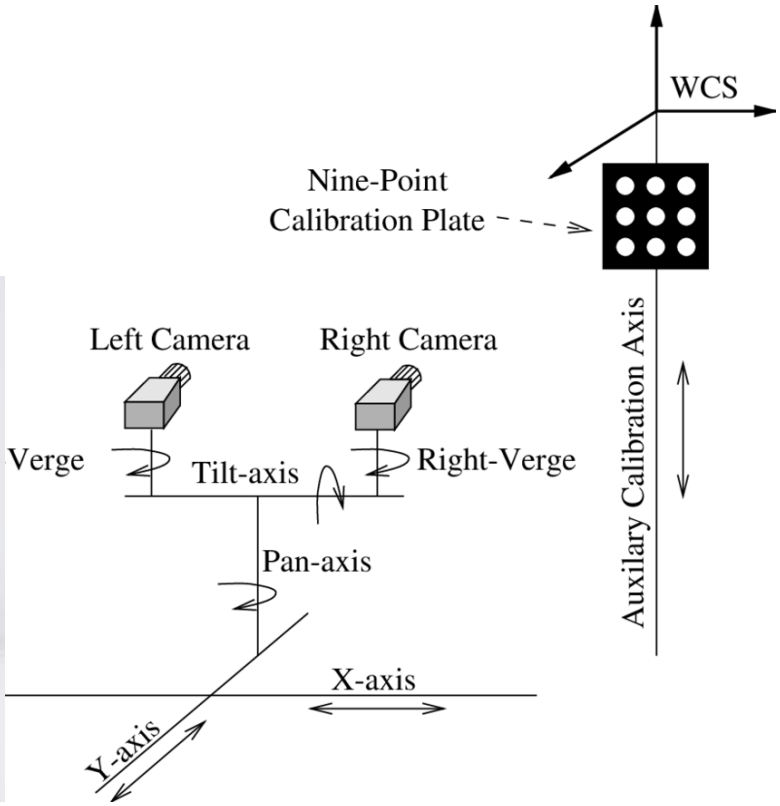
- The Details of Individual Camera Calibration:
  - There are 11 coefficients inside each M matrix.
  - One pair of {(u,v), (X,Y,Z)} provides 2 equations for each M matrix.
  - Six (6) pairs of {(u,v), (X,Y,Z)} are sufficient enough.

$$\begin{pmatrix} s \bullet u_l \\ s \bullet v_l \\ s \end{pmatrix} = \begin{bmatrix} c_{l1} & c_{l2} & c_{l3} & c_{l4} \\ c_{l5} & c_{l6} & c_{l7} & c_{l8} \\ c_{l9} & c_{l10} & c_{l11} & 1 \end{bmatrix} \bullet \begin{pmatrix} {}^rX \\ {}^rY \\ {}^rZ \\ 1 \end{pmatrix}$$



$$\begin{pmatrix} s \bullet u_r \\ s \bullet v_r \\ s \end{pmatrix} = \begin{bmatrix} c_{r1} & c_{r2} & c_{r3} & c_{r4} \\ c_{r5} & c_{r6} & c_{r7} & c_{r8} \\ c_{r9} & c_{r10} & c_{r11} & 1 \end{bmatrix} \bullet \begin{pmatrix} {}^rX \\ {}^rY \\ {}^rZ \\ 1 \end{pmatrix}$$



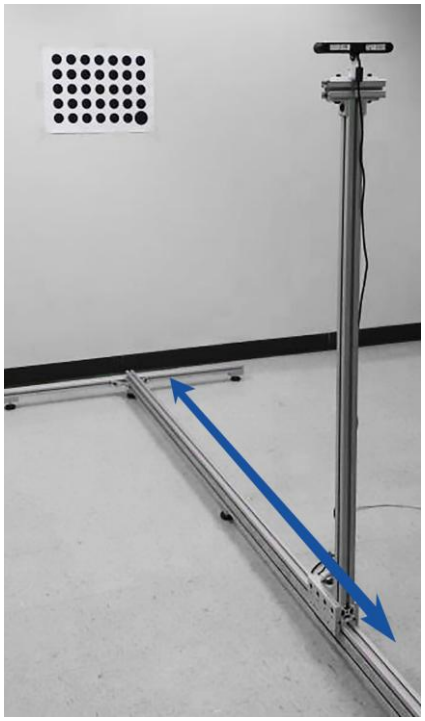


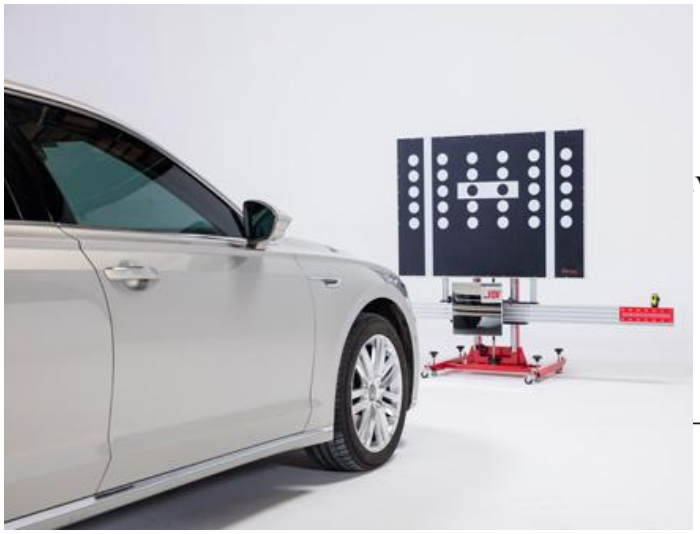
How to calibrate a binocular vision itself?

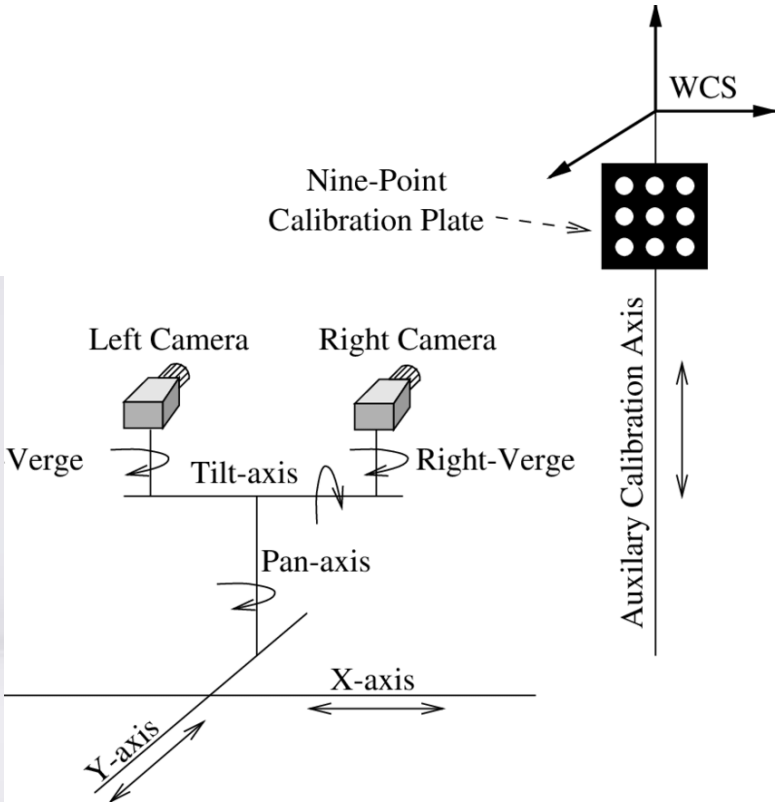
- The Details of Binocular Vision Calibration:
  - There are 19 coefficients inside B matrix.
  - One pair of data provides 3 equations.
  - Seven (7) pairs of data are sufficient enough.

One pair of $\{(u_l, v_l, u_r, v_r)$, (X,Y,Z)$\}$ provides three equations for calibrating binocular vision.

$$\begin{bmatrix} k\,{}^{r}X \\ k\,{}^{r}Y \\ k\,{}^{r}Z \\ k \end{bmatrix} = B_{4\times5} \begin{bmatrix} u_l \\ v_l \\ u_r \\ v_r \\ 1 \end{bmatrix}$$

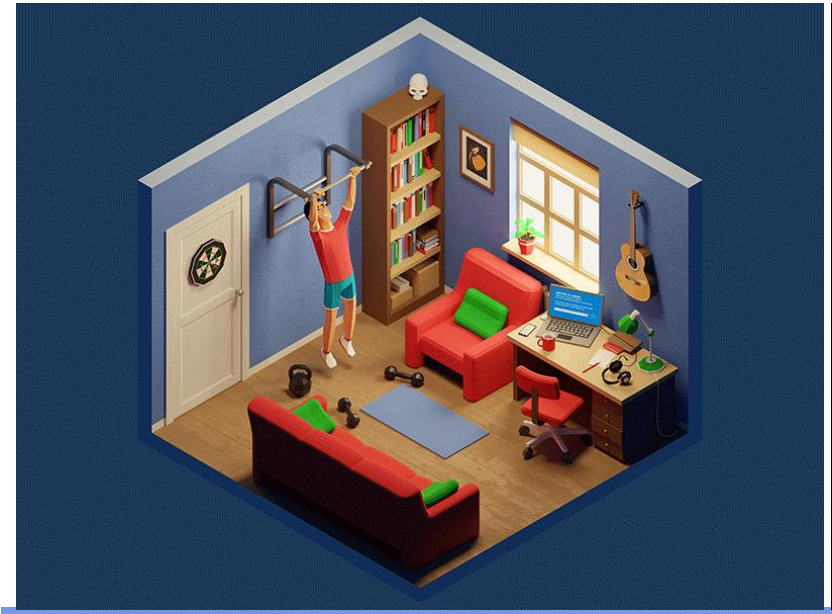


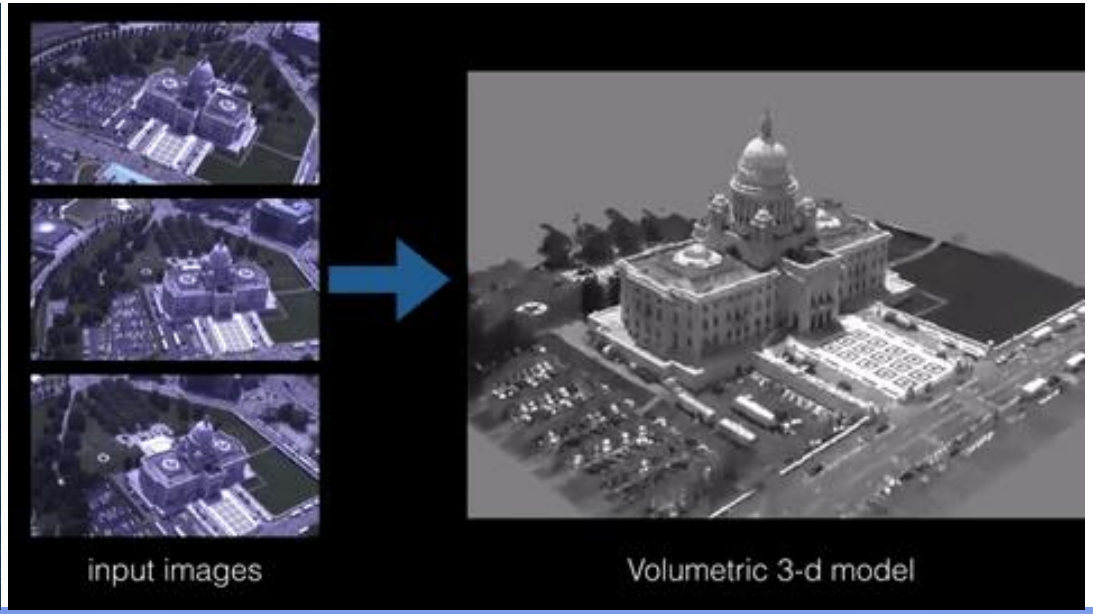




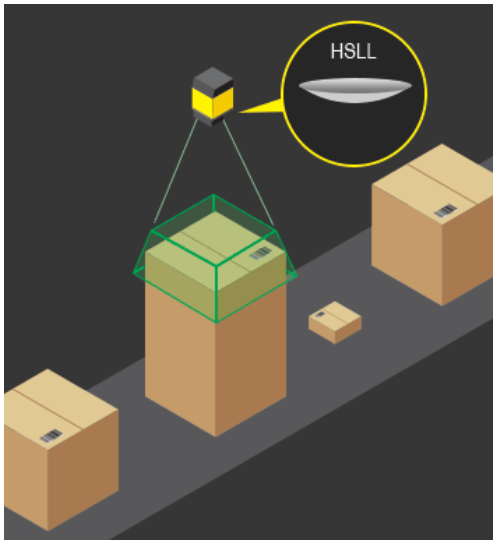
Summary of Lecture 6

- Concept of Coordinate Transformation
- Transformation from Scene to Camera Space
- Transformation from Camera Space to Image Space
- Equation of Binocular Vision
- Calibration of Binocular Vision

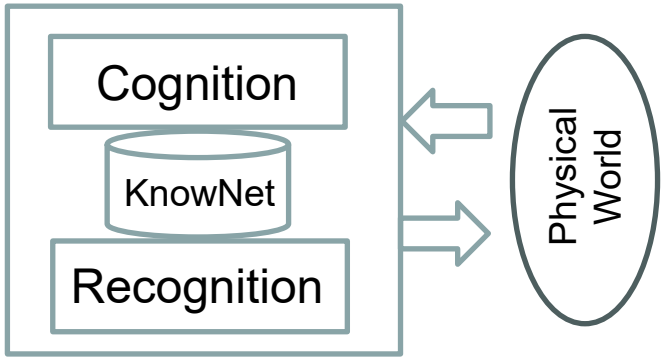




Summary of Module 4



- Lecture 1:
  - Geometry-Driven Computation
- Lecture 2:
  - Fuzziness-Driven Computation
- Lecture 3:
  - Cognition-Driven Computation
- Lecture 4:
  - Recognition-Driven Computation
- Lecture 5:
  - Computation Using Monocular Vision
- Lecture 6:
  - Computation Using Binocular Vision



"Ask not what your country can do for you – ask what you can do for your country," - John F. Kennedy

"Do not think that you are needy – think that you are needed in the world", - Manis Friedman

"Study will make you knowledgeable, resourceful, and hence more needed", - Xie Ming

Thank You for Listening!