# Ch5 Image Transforms

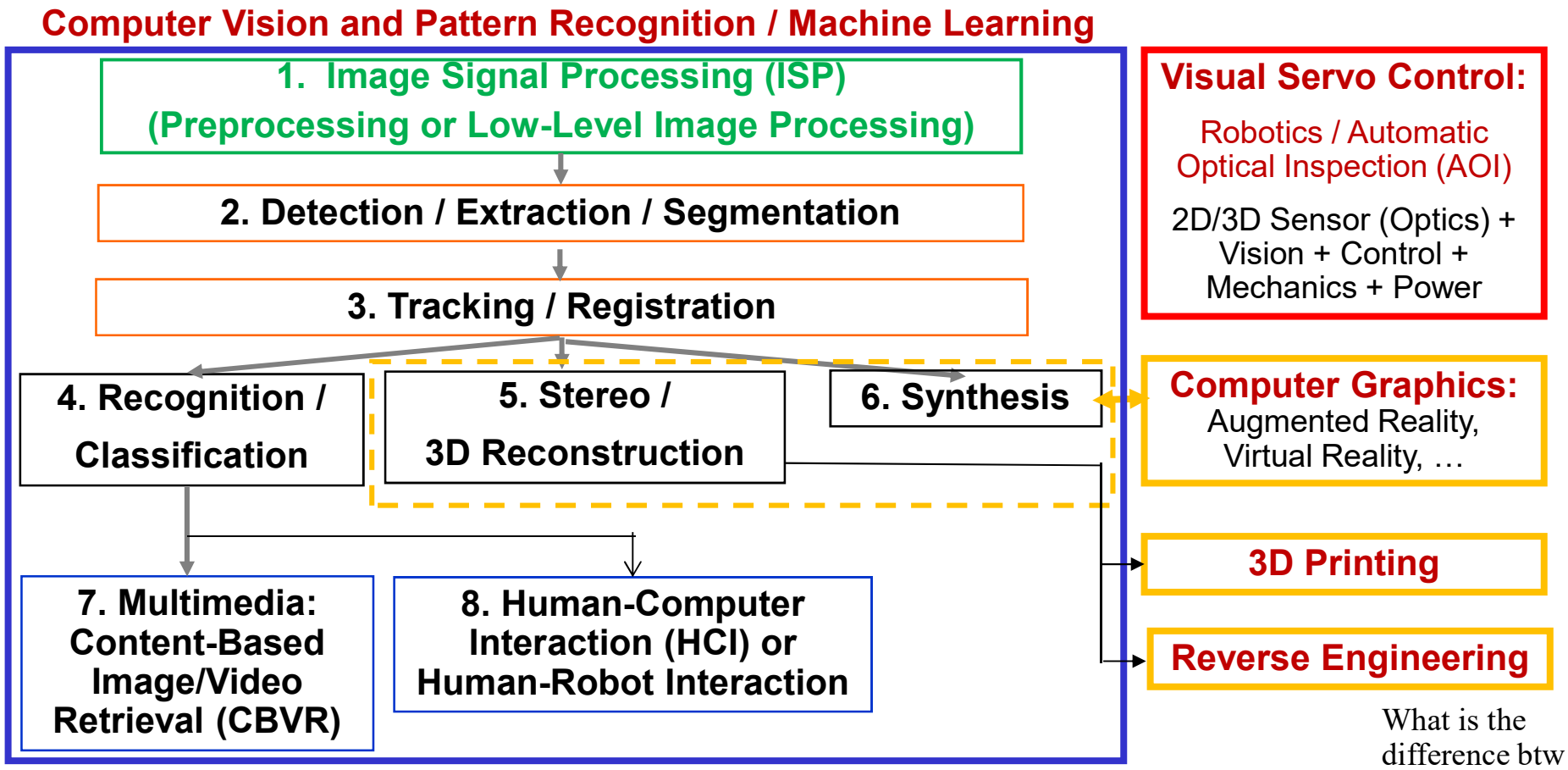
老師：連震杰 教授

Robotics Lab
Institute of Computer Science and Information Engineering,
National Cheng Kung University, Tainan, Taiwan, R.O.C.

# From Computer Vision to Visual Servo Control Techniques

CSIE, EE or ME students…

Welcome to Join RL!!




What is the difference btw CV and CG?

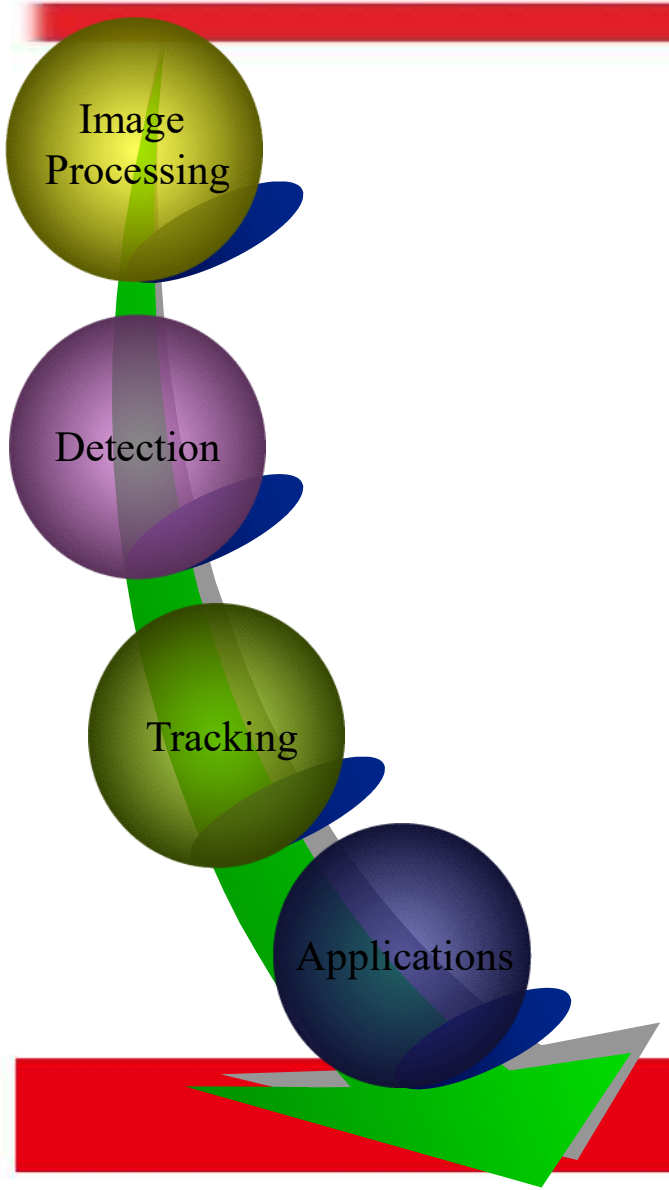
OpenCV

OpenGL

**ISP (Image Signal Processing): Function Block 1**
**Embedded IVA (Intelligent Video Analytics): Function Blocks 2~7**

**ARM: Function Blocks 1~3**
**DSP: Function Blocks 3~7**

# 1.1 Introduction to ISP and Embedded IVA: Functions (1/6)




## Image Processing

- Contrast enhancement.
- 3D enhancement.
- Color enhancement.
- High dynamic range.
- White balance.
- Shadow removing.
- Super resolution.
- Video panorama, Stabilization.
- Calibration: Omni camera.
- Deblurring, denoise, edge...

## Detection

- Face detection.
- Feature point detection.
- Skin color detection.
- Pose estimation.
- Vehicle detection.
- Motion detection.
- Akkon (COF/COG) detection.
- Mura & defect detection.
- Solar slide counter.
- Alignment.

## Tracking

- Face tracking.
- Feature point tracking.
- Vehicle tracking.

## Applications

Recognition/Classification:

- Face recognition.
- Facial expression.
- License plate recognition.
- Solar slide color classification.
- Fingerprint recognition.
- Multiple camera module.

Multimedia:

- Shot cut, Image retrieval.
- Semantic analysis.

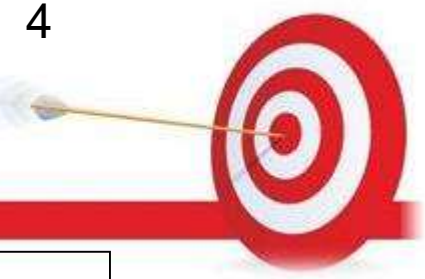
Synthesis:

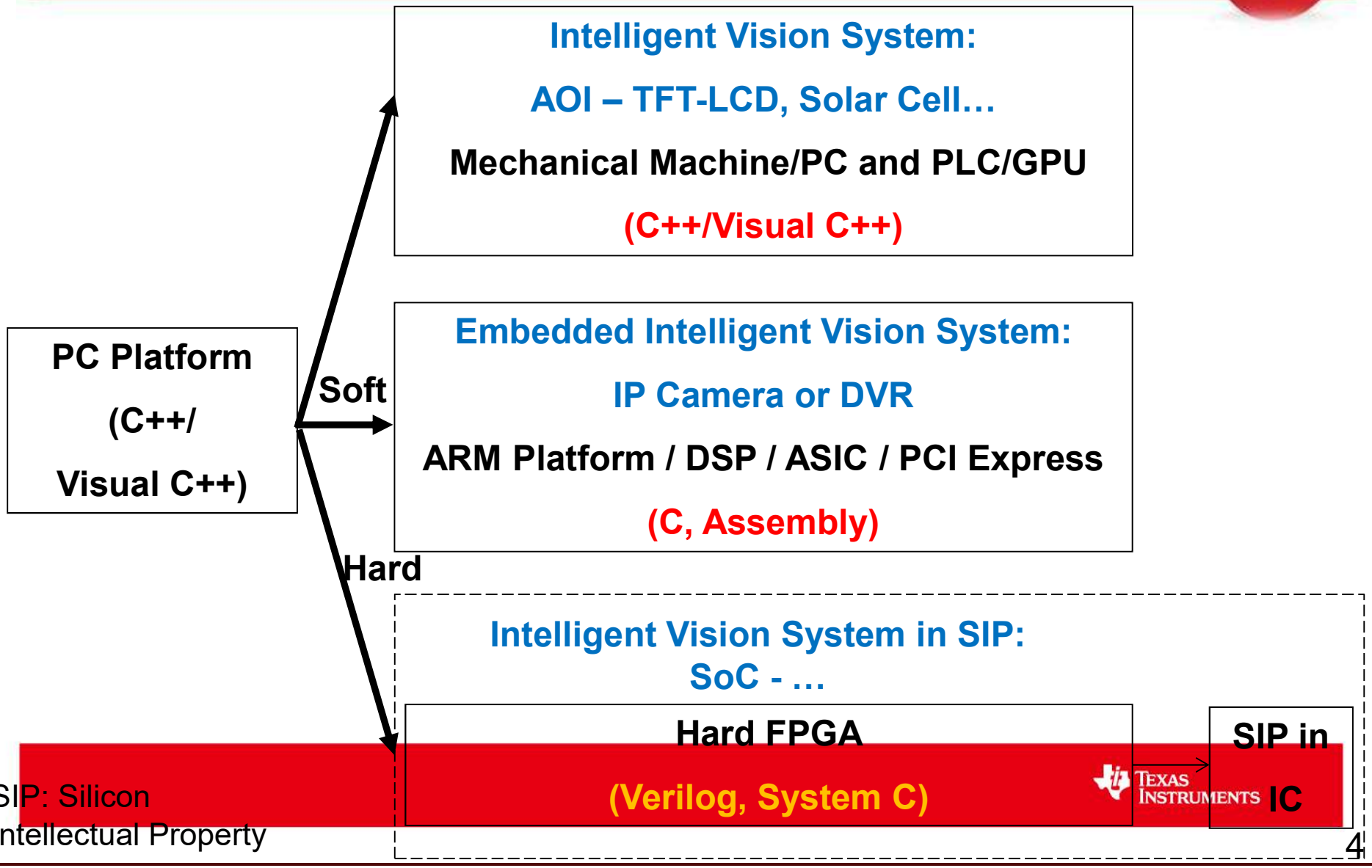
- Image completion.
- Cartoon.
- Exaggeration.
- Frontal view Synthesis.
- 2D → 3D HCI.

3D Reconstruction:

- 3D reconstruction.

# 1.0 Introduction to ISP and Embedded IVA: Apply to Different Platforms (4/4)

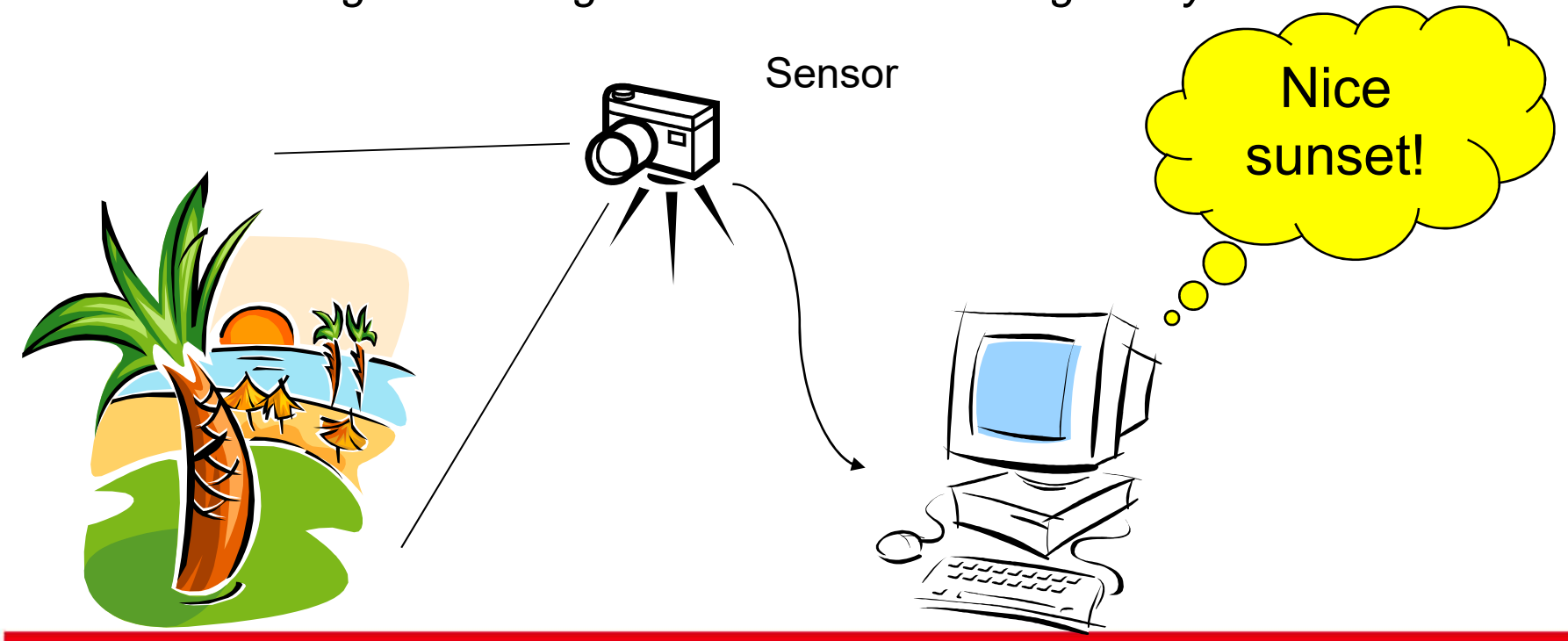





SIP: Silicon Intellectual Property

# 1.0 Introduction to ISP and Embedded IVA: Definition of Computer Vision (2/4)

- What is computer vision?
  - *Sensor + Brain: "Making computers see", cognition.*
  - *Input: 2D Image/Video -> Output: Description of the 3D world*
  - *Intelligence: Image/Video Understanding/Analytics*



# 2.0 Introduction to DM365 and DSP DM6437 (1/2)

| | Texas Instruments (TI) DM365 | Texas Instruments (TI) DSP DM6437 |
|---|---|---|
| **Platform** | Evaluation Module (EVM) | Evaluation Module (EVM) |
| **CPU** | TMS320DM365 (ARM9 300 MHz、150 MIPS) | TMS320DM6437 (DaVinci 600 MHz、600 MIPS) |
| **RAM** | DDR2-270MHz 128 MB | DDR2-400MHz 128 MB |
| **Video Codec Format** | H.264, MPEG4, MPEG2, MJPEG, JPEG, WMV9/VC1 | H.264, MEPG4, MJPEG, JPEG |
| **Video Input Format** | YUV 420P SEMI | YUV 422 (UYVY) |
| **Video Input Resolution** | 5M (Max) | 320×240、640×480、720×480 |
| **Video Input Frame Rate** | 30 fps (720p) | 30 fps |

# 2.0 Introduction to DM365 and DSP DM6437 (2/2)

| | Texas Instruments (TI) DM365 | Texas Instruments (TI) DSP DM6437 |
|---|---|---|
| **Connect to CMS** | Yes | No |
| **Connect to DSP** | Yes | - |
| **Image Signal Processing** | 2D Denoise, 2A, Stabilization… | - |
| **Embedded Motion Object Detection Algorithm** | Yes, | Yes |
| **Embedded AdaBoost (Integral Image) Detection Algorithm** | No, but has face detection function using hardware solution. | Yes, and has AdaBoost (Integral Image) accelerator |
| **Embedded Simple Tracking Algorithm** | Yes, | Yes |
| **Embedded Robust Tracking Algorithm** | No, | Yes |
| **Embedded Recognition Algorithm** | No, | Yes |

# 3.0 History of ISP and IVA (1/6)

**Real-Time CV/ML at Embedded Chip or by Hardware Solution in Infant Period**

- HCI+Stereo (Kinect) at Microsoft
- IVA at TI Platform
- Face Detection Using AdaBoost
- ISP Using Machine Learning

**Real-Time CV/AI at PC Base (After PII 300M Hz)**

- DARPA HID
- Face Detection and Recognition in Market
- DARPA FERET Competition
- DARPA VSAM (Surveillance): Omni-Camera, Stereo Camera
- Human-Computer Interaction Using HMM
- Face Expression Recog. Using HMM
- Face Detection Using NN
- Face Recog. Using PCA
- Face Detection Using Color and Motion
- ISP Using Image Processing

**CV in Infant Period**

- Face Recog. Using Image Processing

1980 | 1990 | 2000 | 2010

1993 J.J. Lien at RI, SCS, CMU

1998 L1-Identity (Omron)

2002 NCKU

CV: Computer Vision
ML: Machine Learning
AI: Artificial Intelligent

# 3.0 History of ISP and IVA: Real Time at PC Base (2/6)

- 1979: Face Recognition Using Image Processing,
  - T. Kanade, Japan

- 1990: Image and Signal Processing (ISP) Using Image Processing
- 1990: Face Detection Using Color and Motion

- 1990/1991: Face Recognition Using PCA,
  - Kirby and Sirovich; M. Turk and A. Pentland, MIT
- 1995: Face Detection Using Neural Networks (NN)
  - H. Rowley and T. Kanade, CMU, and Sung and T. Poggio, MIT
- 1995: Facial Expression Recognition Using Hidden Markov Model (HMM)
  - J.J. Lien and T. Kanade, CMU.
- 1995: Human-Computer Interaction (HCI) Using Hidden Markov Model
  - Media Lab, MIT, T. Huang, UIUC, Robotics Lab, CMU

# 3.0 History of ISP and IVA: Real Time at PC Base (3/6)

- 1997: DARPA Video Surveillance and Monitoring (VSAM) Project
  - Omni-Directional Camera
  - Stereo Camera
  - CMU,…,
  - After VSAM: Object Video
- 1998, 2000: DARPA FERET Face Recognition Competitions
  - Visionics 1st, MIT Media Lab (Viisage) 2nd
  - Visionatics + Viisage = L-1 Identity
- **1998: Face Detection and Recognition in Real Time at PII 300 M Hz in the Market**
  - L1-Identity
- 2000: DARPA Human Identification at a Distance (HID) Project
- 2000: Omron Starts to Do Face Detection with L1-Identity.
- 2004: Image and Signal Processing (ISP) Using Machine Learning

# 3.0 History of ISP and IVA: Real Time at Embedded Base with Machine Learning (4/6)

- **200?: Face Detection (Using AdaBoost) Chip in the Market**
  - **Omron**
- **2008: TI DM365 and DSP DM6437 for Embedded IVA**
  - **Infant Period: Embedded Computer Vision/Machine Learning with Simple Accelerators**
- 2010: Human-Computer Interaction – Gesture Recognition: Stereo Camera Embedded in Chip
  - Microrsoft, Kinect.
- **2012: ISP and IVA with OpenGL and OpenCV (Machine Learning) in Chip**
  - **OpenGL: 3D GUI Acelerator**
  - **OpenCV: ML Accelerators**

# 3.0 History of ISP and IVA: Embedded Base Process - GPU Vs. DSP (5/6)

- Mobile Computing (Qualcomm, nVidia):
  - GPU with OpenGL and OpenCV Accelerators
- Not Mobile Computing (TI):
  - ARM NEON Multi-Media Processor
  - DSP with OpenCV Accelerators
  - OpenGL Accelerators

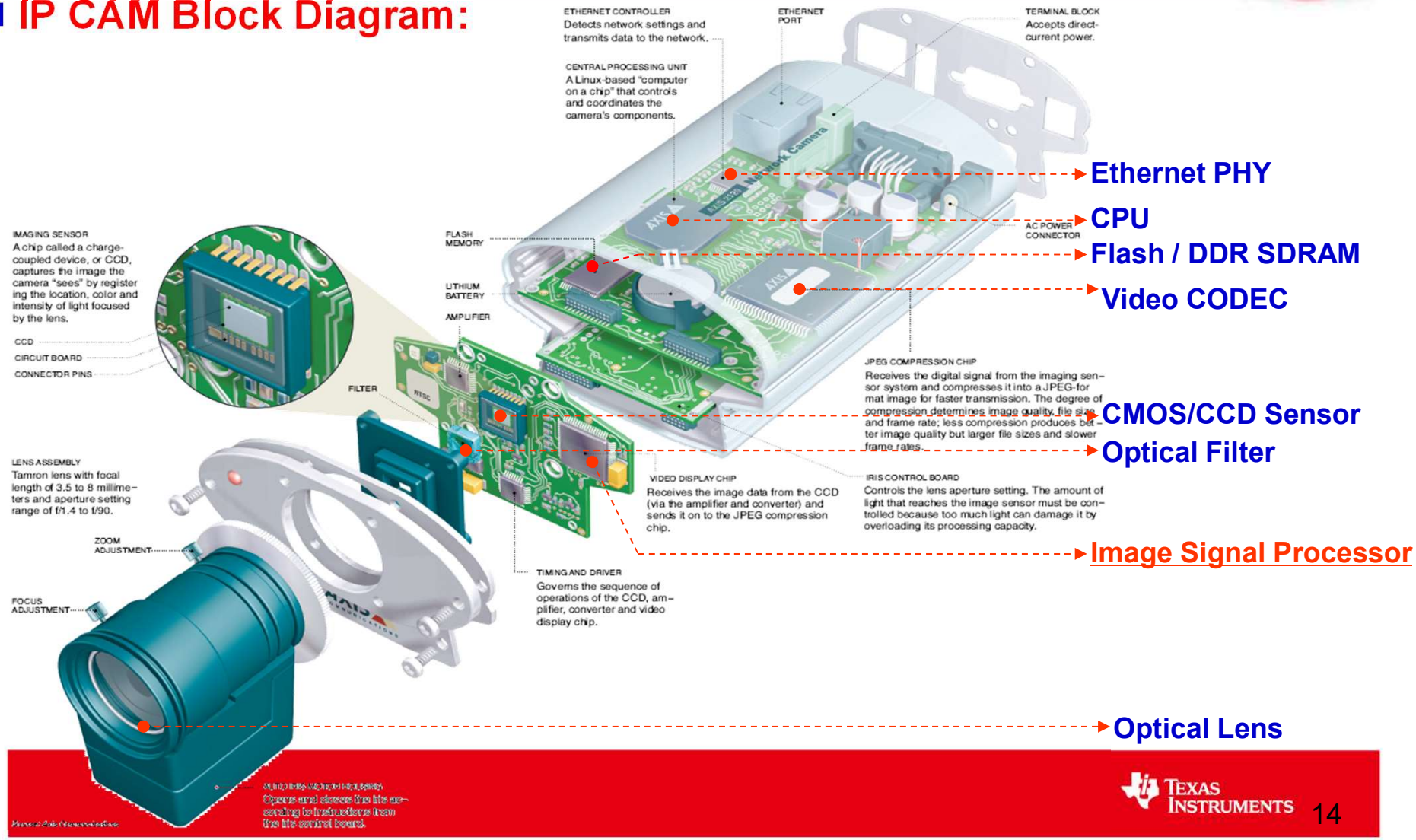
OpenGL: Support 3D GUI. Example – Augmented Reality (AR) display

# 3.0 History of ISP and IVA: Future (6/6)

- Robot => LEGO MindStorms NXT2.0, Mars Robot
  1. Embedded Computer Vision
  2.1 Mobile Computing
  2.2 Wireless Communication
  3. Cloud Computing
  4. Mechanical Control

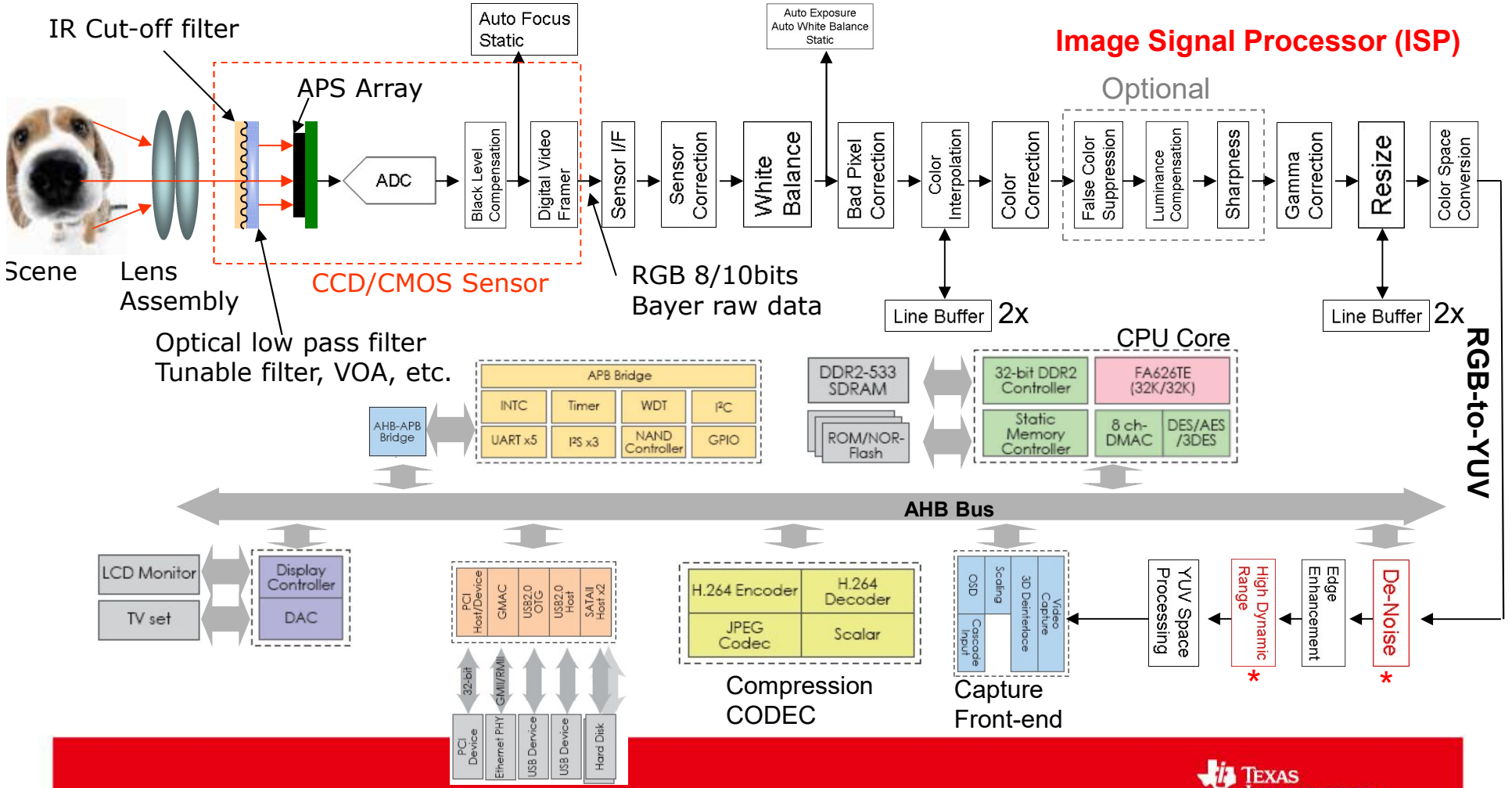
# 1.0 Image and Signal Processing (ISP) (1/2)

- IP CAM Block Diagram:



# 1.0 Image and Signal Processing (ISP) (2/2)

## IP CAM Function Block Diagram:



# 2.0 ISP: 3A

- 3A: Auto Focus, Auto Exposure, Auto White Balance

White Balance

# 3.0 ISP: 3D Denoise

- 3D Denoise: 2D in Spatial Domain + 1D in Temporal Domain
- Noise is filtered without losing edge information.

Before NR

After NR

# 1. Noise Type

- **Gaussian noise**
  - A Gaussian distribution of noise samples.
- **Impulse noise**
  - Also called salt and pepper noise.
  - It represents itself as randomly occurring white and black pixels.
  - An effective noise reduction method for this type of noise involves the usage of a median filter.
- **We focus on reduce the Gaussian noise and impulse (slat and pepper) noise.**

# 1. Noise Type

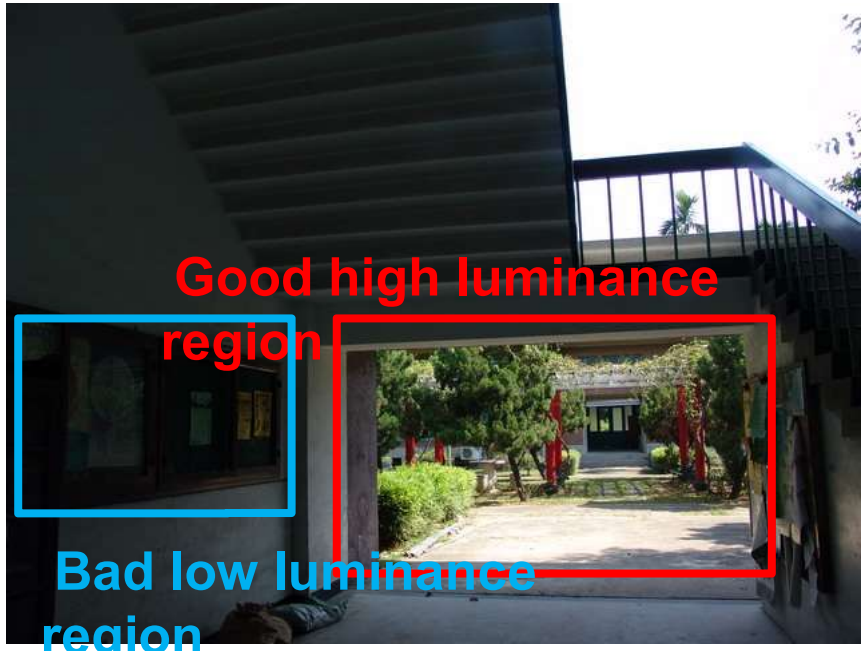
Noise free

With Gaussian noise

With impulse noise

# 4.0 ISP: Wide Dynamic Range (WDR) (1/2)

- Dynamic Range: World scene > Record or display device
- Tone Reproduction: Reduce the gap between display image and scene




$I_L$ : Input image with low exposure

Low-exposed / Unacceptable color / Noise

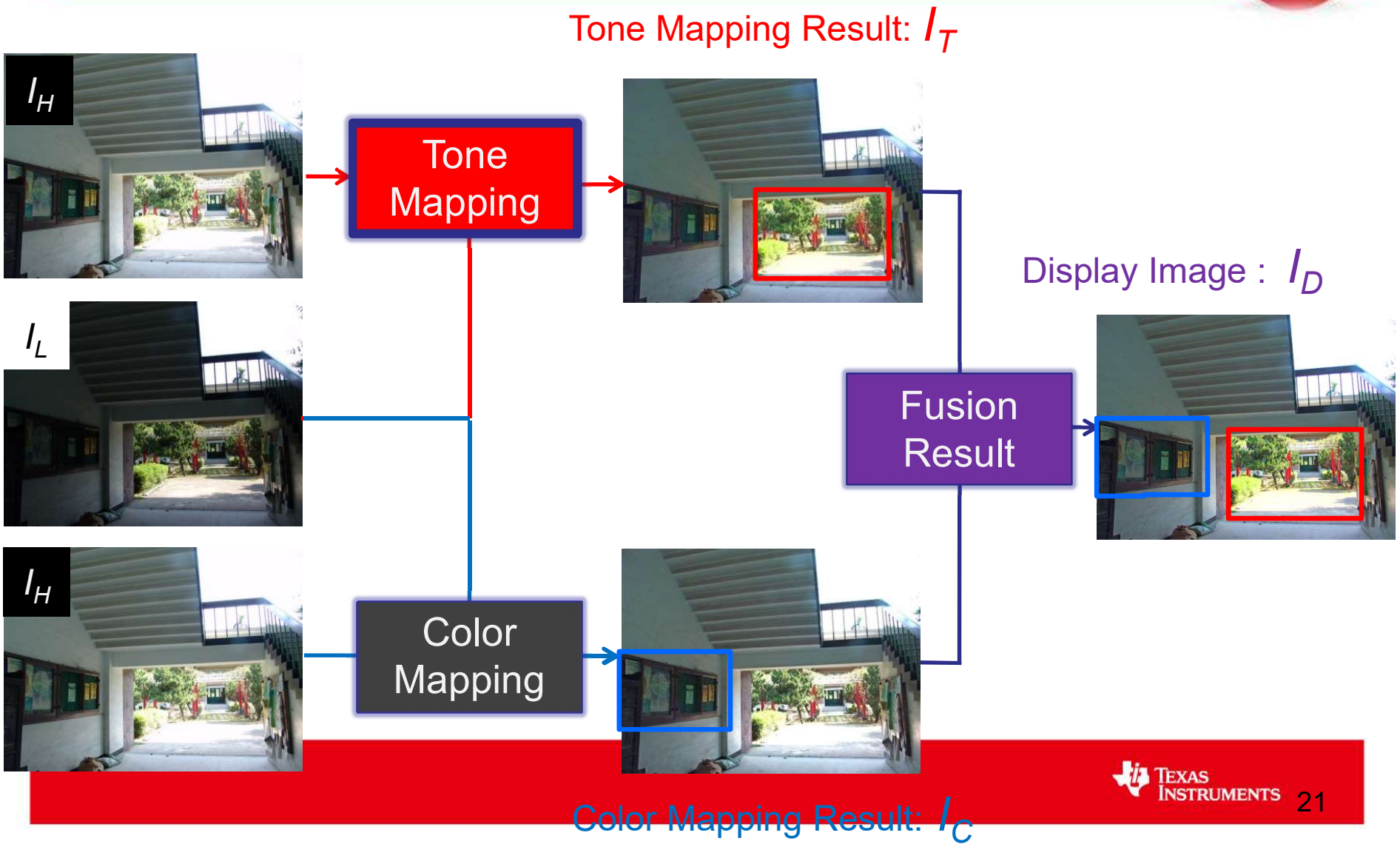
Good details / Saturated color


$I_H$ : Input image with high exposure

Over-exposed / Lose details / Washed out

Less noise / Correct color

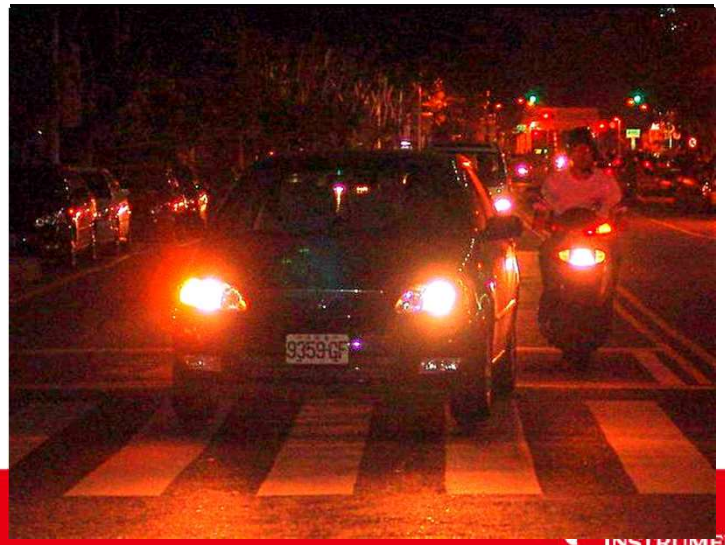
# 4.0 ISP: Wide Dynamic Range (WDR) (2/2)



# 4.0 ISP: Wide Dynamic Range (WDR) - Examples (1/3)

# 4.0 ISP: Wide Dynamic Range (WDR) - Examples (2/3)





# 4.0 ISP: Wide Dynamic Range (WDR) - Examples (3/3)

# 5.0 ISP: Stabilization

5.1 Fixed Scene Stabilization Definition:

1) Usage: For video which is captured by fixed camera.
2) Approach: Reduce the camera motion which consists of jiggle motion and intentional motion

5.2 Panning Stabilization Definition:

1) Usage: For video which is captured by handheld camera.
2) Approach: Reduce jiggle motion but keep intentional motion.

Fixed Scene Stabilization

Panning Stabilization

# 5.1 ISP: Fixed Scene Stabilization Demo (1/3)

Original Input

Stabilized Result

Original Input

Stabilized Result

# 5.1 ISP: Fixed Scene Stabilization Demo (2/3)

Original Input

Stabilized Result

Original Input

Stabilized Result

# 5.1 ISP: Fixed Scene Stabilization Demo (3/3)

Original Input

Stabilized Result

Original Input

Stabilized Result

# 5.2 ISP: Panning Stabilization Demo (1/3)

Original Input

Stabilized Result

Original Input

Stabilized Result

# 5.1 ISP: Panning Stabilization Demo (2/3)

Original Input

Stabilized Result

Original Input

Stabilized Result

# 5.1 ISP: Panning Stabilization Demo (3/3)

Original Input


Stabilized Result

Original Input

Stabilized Result

$$\because e^{j\frac{2\pi}{N}n(k+rN)} = e^{j\frac{2\pi}{N}kn} \times e^{j2\pi rn} = e^{j\frac{2\pi}{N}kn}$$

$$\theta = \omega t = \Omega t = 2\pi f t = \frac{2\pi}{T}t$$

# . Discrete Fourier Series (3/3) ??

- DFS正反變換關係

$$e^{-j\frac{2\pi}{N}kn} = W_N^{kn}$$

$n \in N$ (*total samples* for 1 *period* in time T doma

Freq. Domain

$$\tilde{X}(k) = DFS[\tilde{x}(n)] = \sum_{n=0}^{N-1} \tilde{x}(n) e^{-j\frac{2\pi}{N}kn} = \sum_{n=0}^{N-1} \tilde{x}(n) W_N^{kn}$$

$k \in N$ (*total samples* for 1 *period* in f domain)

Time Domain

$$\tilde{x}(n) = IDFS[\tilde{X}(k)] = \frac{1}{N}\sum_{k=0}^{N-1} \tilde{X}(k) e^{j\frac{2\pi}{N}kn} = \frac{1}{N}\sum_{n=0}^{N-1} \tilde{X}(k) W_N^{-kn}$$

$$\tilde{X}(k+mN) = \sum_{n=0}^{N-1} \tilde{x}(n) e^{-j\frac{2\pi}{N}(k+mN)n} = \sum_{n=0}^{N-1} \tilde{x}(n) e^{-j\frac{2\pi}{N}kn} = \tilde{X}(k)$$

$\tilde{X}(k)$也是一個週期為N的週期序列

# Property of DFS (3/3) jj, ??

- Periodic convolution

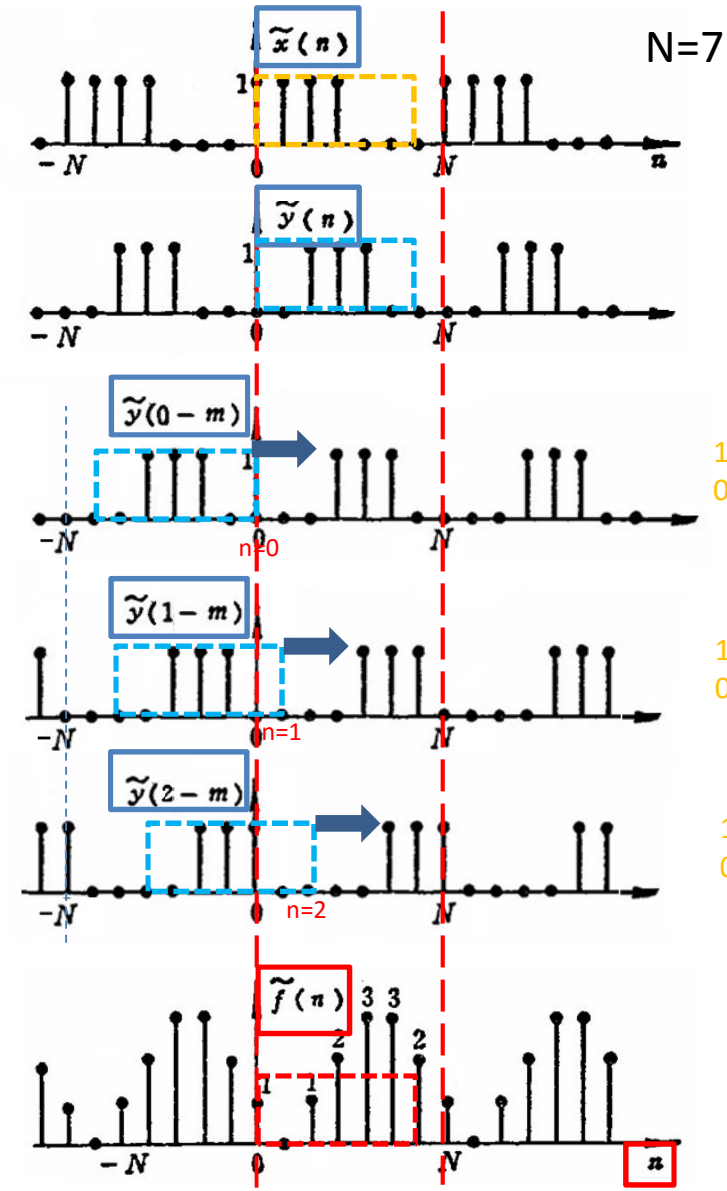
$$\tilde{f}(n)=\sum_{m=0}^{N-1}\tilde{x}(m)\tilde{y}(n-m)$$

n=0 $$\tilde{f}(0)=\sum_{m=0}^{N-1}\tilde{x}(m)\tilde{y}(0-m)=1$$

why Inverse ??

n=1 $$\tilde{f}(1)=\sum_{m=0}^{N-1}\tilde{x}(m)\tilde{y}(1-m)=0$$

n=2 $$\tilde{f}(2)=\sum_{m=0}^{N-1}\tilde{x}(m)\tilde{y}(2-m)=1$$




??

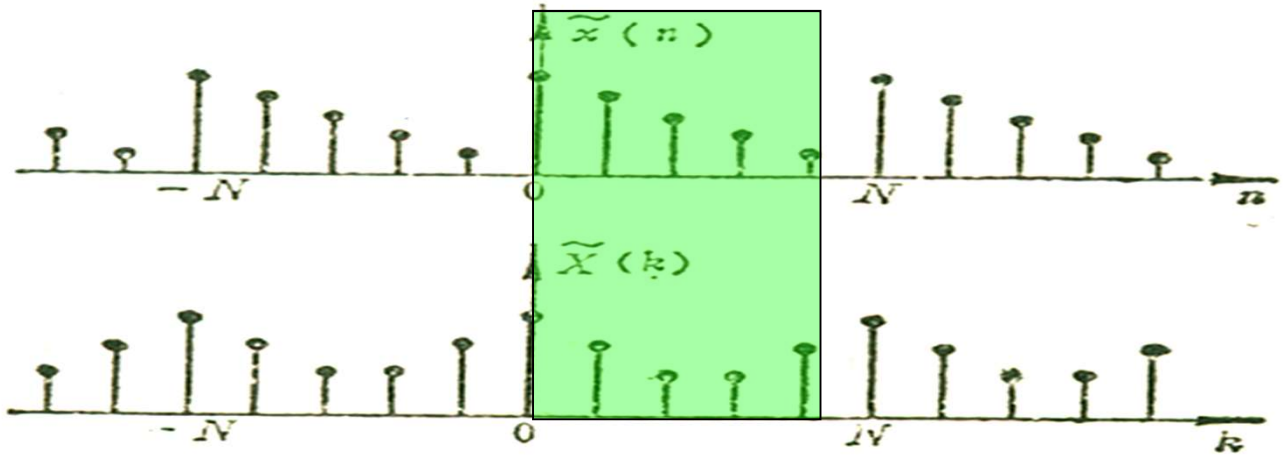
1x0+1x0+1x0+1x1+
0x1+0x1+0x0 =1

1x0+1x0+1x0+1x0+
0x1+0x1+0x1 =0

1x1+1x0+1x0+1x0+
0x1+0x1+0x1 =1

# Discrete Fourier Transform (1/5)

- **Periodic sequence and its DFS**




Periodic sequence is infinite length.
But only N(=5 ex.) sequence values contain information.

# Discrete Fourier Transform (2/5)

DFS
- Periodic sequence can be seen as periodically copies of finite-length sequence.

DFT
- Finite-length sequence can be seen as extracting one period from periodic sequence.

DFS

**Periodic Sequence**

DFT

**Finite-duration Sequence**


Main period

# Discrete Fourier Transform (3/5)

DFT DFS

- Suppose $x(n)$ is a finite-length sequence; $\tilde{x}(n)$ is a periodic sequence;

$$x(n) = \tilde{x}(n)R_N(n)$$

*R*: Mask, filter as Sobel edge detection filter…
Impulse, square-wave, causal…..

$R_N(n)$ is a square-wave sequence

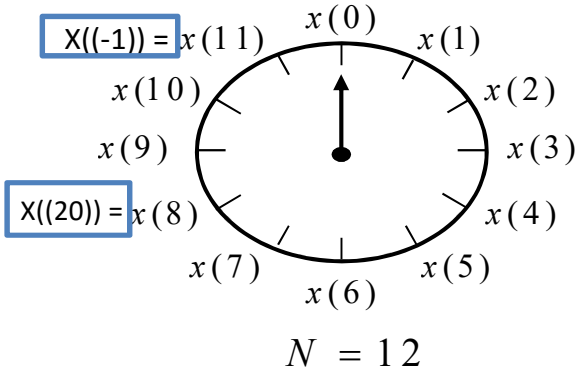
$$R_N(n) = \begin{cases} 1, & 0 \le n \le N-1 \\ 0, & otherwise \end{cases}$$

$$\tilde{x}(n) = \sum_{r=-\infty}^{r=\infty} x(n+rN) = x\left(\left(n\right)\right)_N$$

n modulo N

Example: N=12

$$x\left(\left(5\right)\right)_{12} = x(5)$$
$$x\left(\left(20\right)\right)_{12} = x(8)$$
$$x\left(\left(-1\right)\right)_{12} = x(11)$$



# Discrete Fourier Transform (4/5) jj

- Computation of DFT by extracting one period of DFS

To a finite-length sequence:

$R_N(n)$ is a square-wave sequence

$$R_N(n) = \begin{cases} 1\,, 0 \le n \le N-1 \\ 0\,, \quad otherwise \end{cases}$$

$$X(k) \Leftrightarrow DFS[x((n))_N] \; R_N(k)$$

DFT

**Periodical copies**

**DFS of periodic sequence**

**Get DFT by extracting one period of DFS**

# Discrete Fourier Transform (5/5) jj

$$\tilde{X}(k) = DFS[\tilde{x}(n)] = \sum_{n=0}^{N-1} \tilde{x}(n) e^{-j\frac{2\pi}{N}kn} = \sum_{n=0}^{N-1} \tilde{x}(n) W_N^{kn}$$

DFS

$$\tilde{x}(n) = IDFS[\tilde{X}(k)] = \frac{1}{N}\sum_{k=0}^{N-1} \tilde{X}(k) e^{j\frac{2\pi}{N}kn} = \frac{1}{N}\sum_{n=0}^{N-1} \tilde{X}(k) W_N^{-kn}$$

- finite-length sequence can be expressed in this way:

$$x(n) = \tilde{x}(n) R_N(n)$$

$R_N(n)$ is a square-wave sequence

$$R_N(n) = \begin{cases} 1, 0 \le n \le N-1 \\ 0, \quad otherwise \end{cases}$$

$$\therefore X(k) = \tilde{X}(k) R_N(k) = [\sum_{n=0}^{N-1} x((n))_N W_N^{kn}] R_N(k)$$

$$\text{then:} \quad X(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn}, 0 \le k \le N-1$$

$$\text{then:} \quad x(n) = \frac{1}{N}\sum_{k=0}^{N-1} X(k) W_N^{-kn}, 0 \le n \le N-1$$

**DFT Transform Pair**

$$\begin{cases} X(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn}, 0 \le k \le N-1 \\ x(n) = \frac{1}{N}\sum_{k=0}^{N-1} X(k) W_N^{-kn}, 0 \le n \le N-1 \end{cases}$$

# Property of DFT (1/6) jj, ??

Assume X[k]=DFT[x(n)], Y(k)=DFT[y(n)]

(1) Linearity:

$$DFT[ax(n)+by(n)]=aX(k)+bY(k),\quad a,b \text{ are coefficient}$$

(2) Circular Shift:

Circular shift of x(n) can be defined:

$$x_m(n)=x((n+m))_N R_N(n)$$

$R_N(n)$ is a square-wave sequence

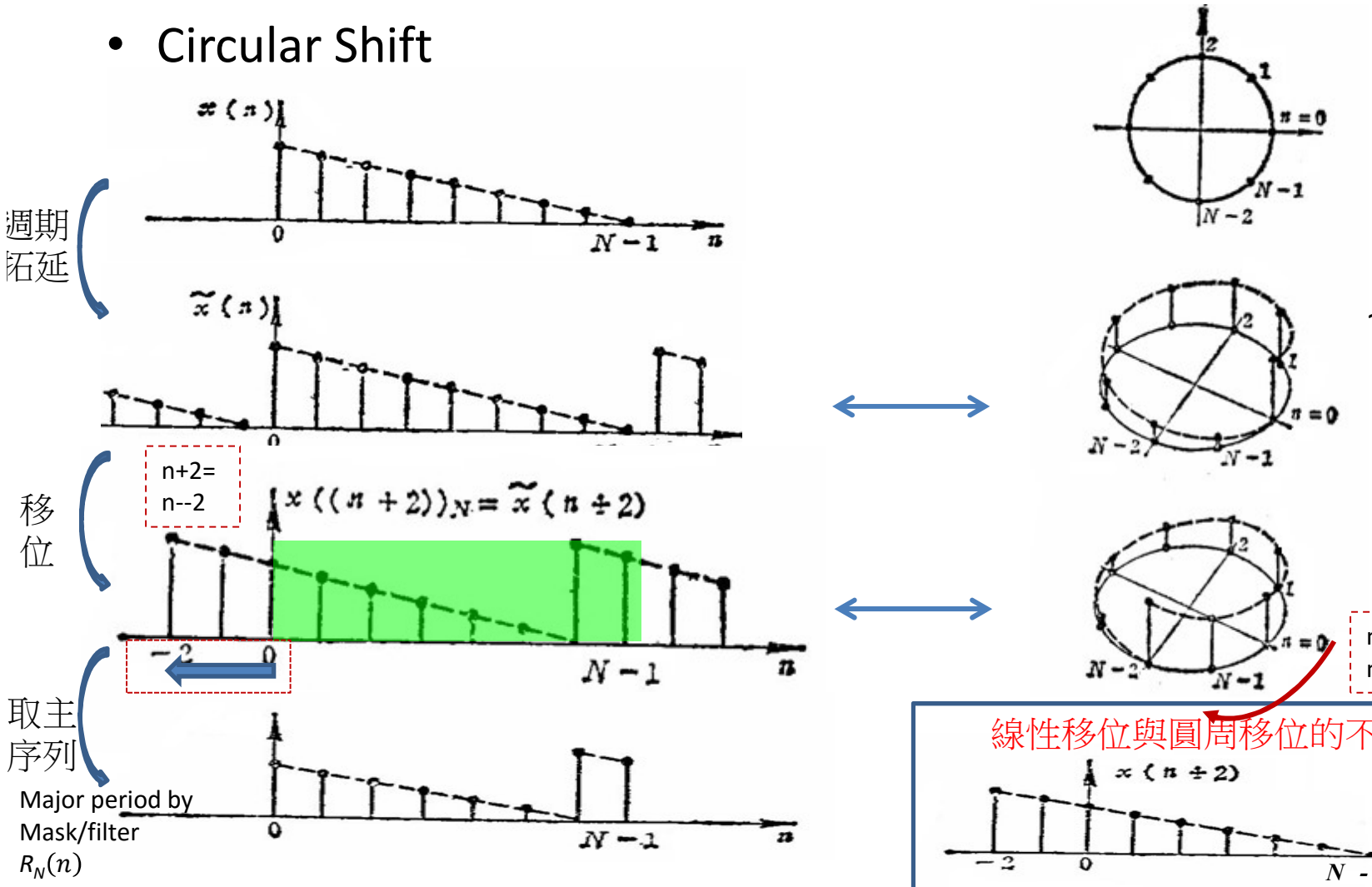
$$R_N(n)=\begin{cases}1, 0\le n\le N-1\\ 0, \quad otherwise\end{cases}$$

$$x(n) \xrightarrow[??]{\text{拓展}} \tilde{x}(n) \xrightarrow[\text{-: inverse}]{\text{移位 -(-m)}} \tilde{x}(n+m)=x((n+m))_N \xrightarrow{\text{取主值序列}} x_m(n)$$

Major period by Mask/filter $R_N(n)$

# Property of DFT (2/6) ??

- Circular Shift



# Property of DFT (4/6)

(3) Circular convolution (圓周卷積)

Suppose : $\mathrm{F}(k)=\mathrm{X}(k)Y(k)$

Frequency domain product
= Time domain periodic convolution

$$f(n) = IDFS[F(k)] = \sum_{m=0}^{N-1} x(m)\, y\big((n-m)\big)_N \; R_N(n)$$
$$= \sum_{m=0}^{N-1} y(m)\, x\big((n-m)\big)_N \; R_N(n)$$

Suppose : $f\ (n)=x(n)y(n)$

Time domain product
= Frequency domain periodic convolution

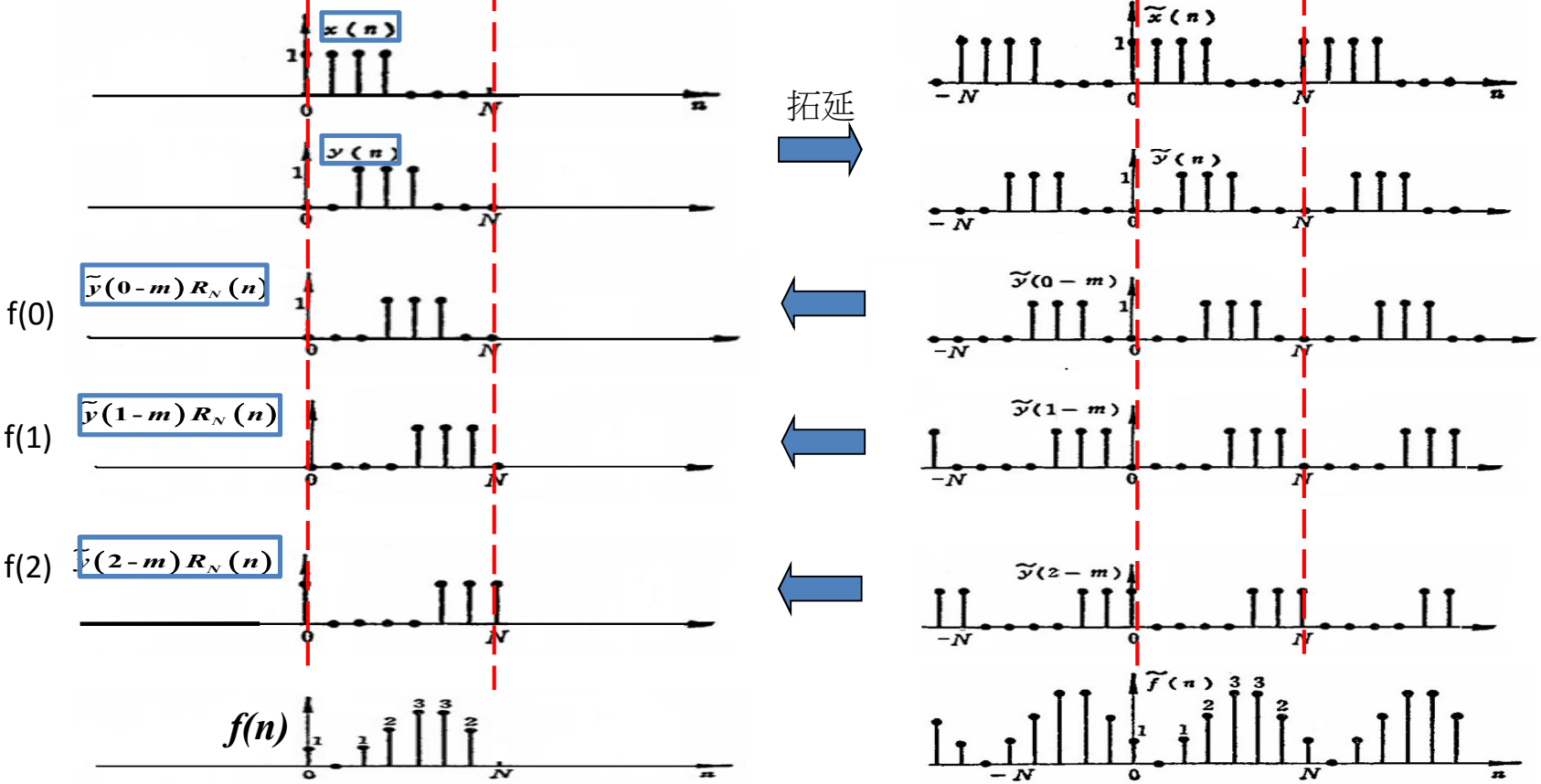
$$F\ (k) = DFS[f(n)] = \frac{1}{N}\sum_{l=0}^{N-1} X(l)\, Y\big((k-l)\big)_N \; R_N(n)$$
$$= \frac{1}{N}\sum_{l=0}^{N-1} Y(l)\, X\big((k-l)\big)_N \; R_N(n)$$

# Property of DFT (5/6) ??

- Circular convolution

??

$$f(n)=\sum_{m=0}^{N-1} x(m)\, y\big((n-m)\big)_N\, R_N(n)$$




DFT

**Circular convolution**

DFS

**Periodic convolution**

# Property of DFT (6/6) jj

- Circular Convolution (圓周卷積) 沒有實際的物理意義，但是可以用它的圓周卷積特性來快速的計算兩個週期序列的卷積。

$$DFT[x(n)] = X(k)$$

$$DFT[y(n)] = Y(k)$$

$$IDFT[X(k)Y(k)] = x(n) \otimes y(n)$$

DFS ➢ **Periodic convolution is convolution of two sequences with period N in one period, so it is also a periodic sequence with period N.**

DFT ➢ **Circular convolution is acquired by extracting one period of periodic convolution , expressed by ⊗.**

# Spatial correlation and convolution (1/3) jj

w: 1D filter or window

Len(f)=8
Len(w)=5

(a) 1D function *f* & filter *w*

(b) Starting point

Length=
len(f)+2x(len(w)-1)

8+2x(5-1)=16

(c) Zero padding and correlation at 1st point

(d) 2nd correlation value

(e) Shit by 4

(f)Final point

Length=
len(f)+len(w)-1

8+2x(5-1)=16

(g) Full correlation result (f+w-1)

(h)Cropping with length f

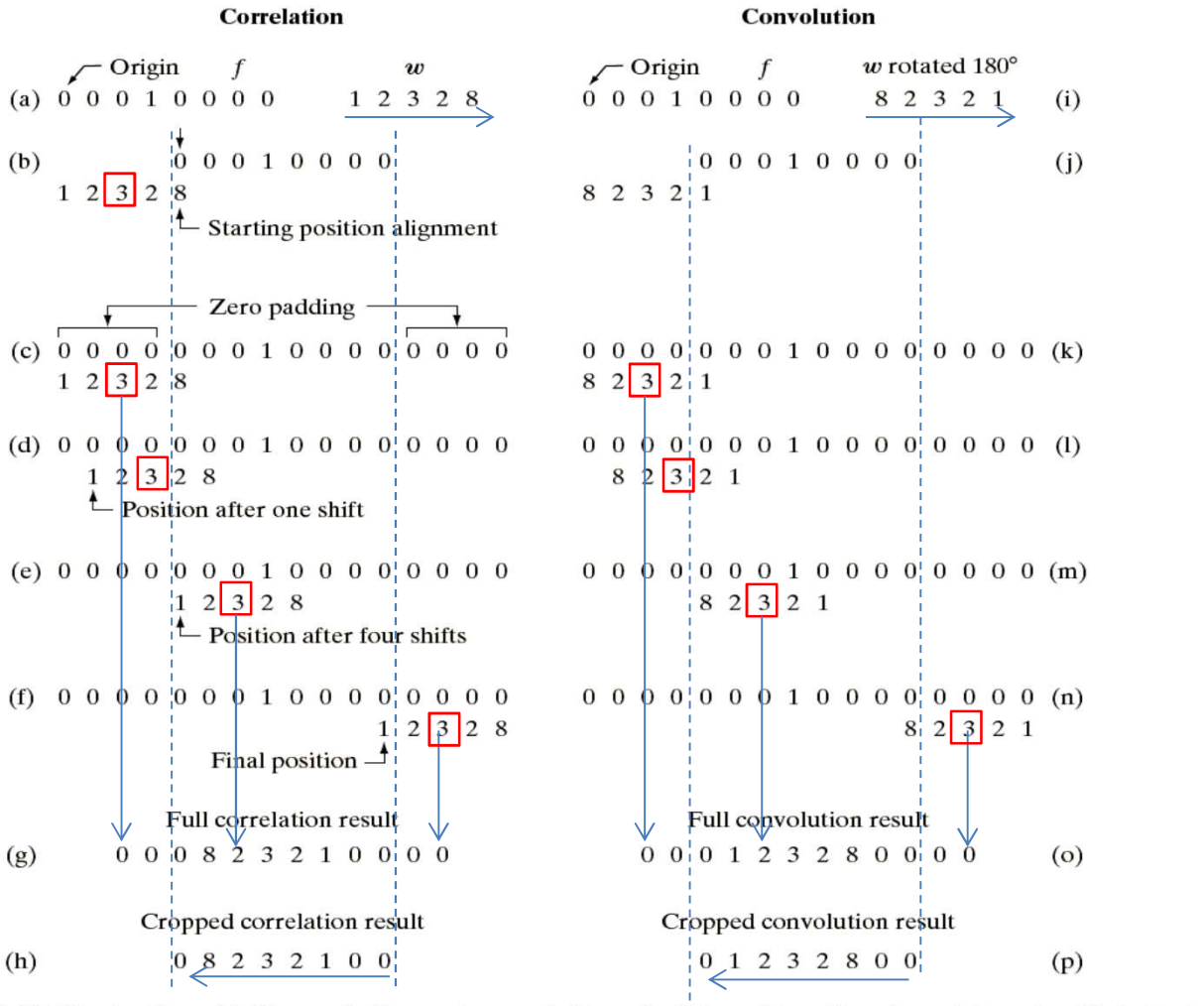



**FIGURE 3.29** Illustration of 1-D correlation and convolution of a filter with a discrete unit impulse. Note that correlation and convolution are functions of *displacement*.

# Spatial correlation and convolution (2/3) jj

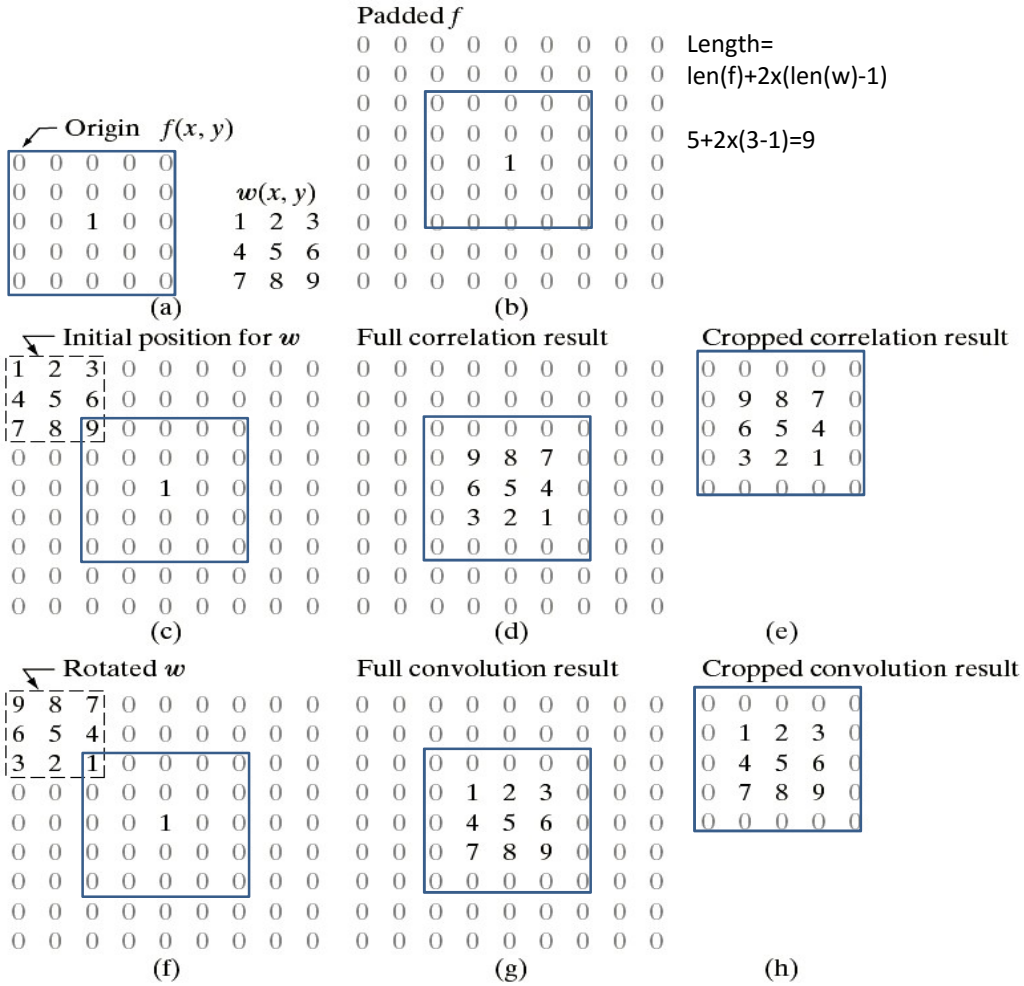
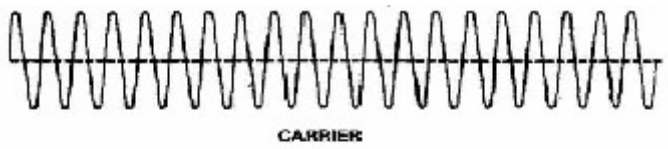



**FIGURE 3.30** Correlation (middle row) and convolution (last row) of a 2-D filter with a 2-D discrete, unit impulse. The 0s are shown in gray to simplify visual analysis.

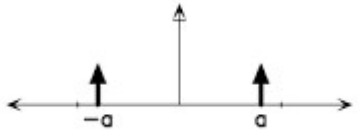
# Spatial correlation and convolution (3/3)

Convolution in Communication System (amplitude modulation)

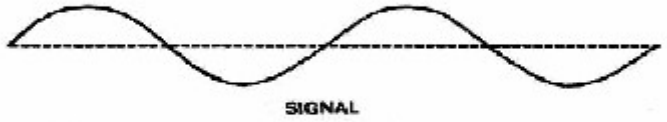
cos(Θ)




F{cos(Θ)}

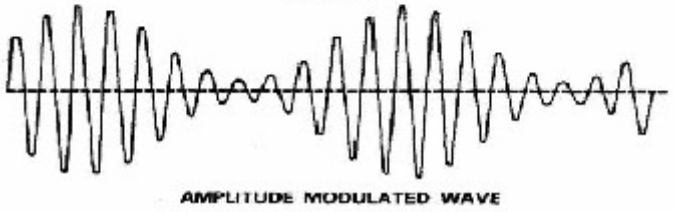



s(t)

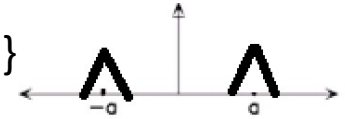



F{s(t)}

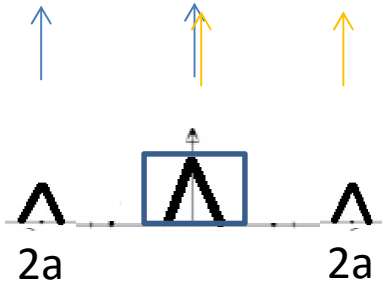
1) Transmit
s(t)•cos(Θ)




F{s(t)•cos(Θ)}




=F{cos(Θ)} convolution F{s(t)}




2) Receive
- After receiving modulated wave s(t)•cos(Θ) ,
mutiply cos(Θ) again can decode the modulated wave

4-connected component
8-connected component

# 2.1 Canny Edge Detection Algorithm jj3

➢ Canny Algorithm Steps:

Canny比 Sobel 多做了2.4.5項

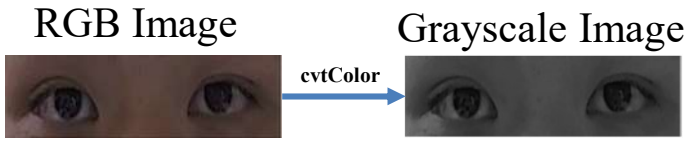
1. Image Convert to Grayscale
2. Smoothing: Noise Reduction (Smoothing)
3. Edge Detection: Compute Gradient Magnitude and Angle using Sobel
4. Thinning: Non-Maximum Suppression
5. Connectivity: Hysteresis Thresholding: Low and High Thresholds

➢ OpenCV Canny Function Call

➢ Edge detection performance:
-Wavelets edge detection (slow) > canny > sobel (fast)

# 2.1 Canny Algorithm – Steps (1/3) jj, ??

1. Image Convert to Grayscale




**cvtColor( image, gray_image, CV_BGR2GRAY )**
**Parameters:**
**1) image** – a source image (image)
**2) Gray_image** - a destination image (gray_image), in which we will save the converted image.
**3) CV_BGR2GRAY** - an additional parameter that indicates what kind of transformation will be performed.

High-pass filer, sum=0.0

2. Noise Reduction (Smoothing)

Low-pass filer, sum=1.0

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

$$\frac{1}{273}\begin{bmatrix}1 & 4 & 7 & 4 & 1\\ 4 & 16 & 26 & 16 & 4\\ 7 & 26 & 41 & 26 & 7\\ 4 & 16 & 26 & 16 & 4\\ 1 & 4 & 7 & 4 & 1\end{bmatrix}$$

-Note that this 5 x 5 filter is roughly equivalent to the Gaussian filter

$$\frac{1}{159}*\begin{bmatrix}2 & 4 & 5 & 4 & 2\\ 4 & 9 & 12 & 9 & 4\\ 5 & 12 & 15 & 12 & 5\\ 4 & 9 & 12 & 9 & 4\\ 2 & 4 & 5 & 4 & 2\end{bmatrix}$$

J: -Usually 3x3 is enough, normally, 7x7 is too big. We don't want it filters out the high-frequency components. So wavelet is one solution but not in this case.

3. Compute Gradient Magnitude and Angle

- Compute the derivatives (Dx(x, y) and Dy(x, y)) of the image in the x and y directions.
  Ex: Sobel filter

Derivative = slope, High freq.

$$Dx(x,y)=\begin{bmatrix}-1 & 0 & +1\\ -2 & 0 & +2\\ -1 & 0 & +1\end{bmatrix} \qquad Dy(x,y)=\begin{bmatrix}+1 & +2 & +1\\ 0 & 0 & 0\\ -1 & -2 & -1\end{bmatrix}$$

- Compute the gradient magnitude:

$$D=\sqrt{D_x^2(x,y)+D_y^2(x,y)}$$

J:?? Should D normalizes between 0~255 in order fro later threshold?

- Compute the angle of the gradient:

$$\theta=\arctan\left(\frac{D_y(x,y)}{D_x(x,y)}\right)$$

OpenCV 2.4.8.0 documentation » OpenCV Tutorials :

$$G_x=\begin{bmatrix}-1 & 0 & +1\\ -2 & 0 & +2\\ -1 & 0 & +1\end{bmatrix} \quad G_y=\begin{bmatrix}-1 & -2 & -1\\ 0 & 0 & 0\\ +1 & +2 & +1\end{bmatrix} \quad G=\sqrt{G_x^2+G_y^2} \quad \theta=\arctan\left(\frac{G_y}{G_x}\right)$$

After this process, what will the edge looks like?

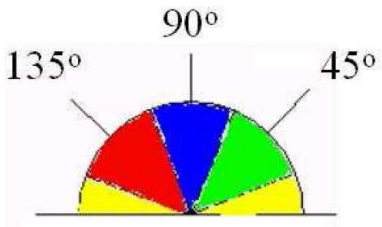
# 2.1 Canny Algorithm – Steps (2/3) jj, ??

4. Non-Maximum Suppression (thinning)

- Goal: The edges it finds can be either very thick or very narrow depending on the intensity across the edge and how much the image was blurred first.
- Non-Maximum Suppression:

Three pixels in a 3 × 3 around pixel (x, y) are examined, there are 4 conditions:

- If $\theta'(x, y) = 0°$, then the pixels $(x + 1, y)$, $(x, y)$, and $(x - 1, y)$ are examined.
- If $\theta'(x, y) = 90°$, then the pixels $(x, y + 1)$, $(x, y)$, and $(x, y - 1)$ are examined.
- If $\theta'(x, y) = 45°$, then the pixels $(x + 1, y + 1)$, $(x, y)$, and $(x - 1, y - 1)$ are examined.
- If $\theta'(x, y) = 135°$, then the pixels $(x + 1, y - 1)$, $(x, y)$, and $(x - 1, y + 1)$ are examined.




Compute $\theta'$ by rounding the angle $\theta$ to one of four directions $0°$, $45°$, $90°$, or $135°$.

Obviously for edges, $180° = 0°$, $225° = 45°$, etc.

- Choose edge pixel :

- If pixel (x, y) has the highest gradient magnitude of the three pixels examined for any one condition, it is kept as an edge.

- If one of the other two pixels has a higher gradient magnitude, then pixel (x, y) is not on the "center" of the edge and should not be classified as an edge pixel.(如果周圍比自己大就忽略)

J: Will this kind of judgment cause slow performance?

After this process, what will the edge looks like?

# 2.1 Canny Algorithm – Steps (3/3) jj

5. Hysteresis Thresholding: Two thresholds – $t_{low}$ and $t_{high}$ (connectivity)

- Existing problem (solve by canny): A simple threshold may actually remove valid parts of a connected edge, leaving a disconnected final edge image.

J: It wants to keep connected edge/components.

- Hysteresis Thresholding

  - If pixel $(x, y)$ has gradient magnitude less than $t_{low}$ discard the edge (write out black).
  - If pixel $(x, y)$ has gradient magnitude greater than $t_{high}$ keep the edge (write out white).
  - If pixel $(x, y)$ has gradient magnitude between $t_{low}$ and $t_{high}$ and any of its neighbors in a $3 \times 3$ region around it have gradient magnitudes greater than $t_{high}$, keep the edge (write out white).
  - If none of pixel $(x, y)$'s neighbors have high gradient magnitudes but at least one falls between $t_{low}$ and $t_{high}$, search the $5 \times 5$ region to see if any of these pixels have a magnitude greater than $t_{high}$. If so, keep the edge (write out white).
  - Else, discard the edge (write out black).

J: Will this kind of judgment cause slow performance?

# 2.1 Canny Algorithm – OpenCV Function Call jj, ??

➢ **Void Canny**(InputArray **detected_edges**, OutputArray **detected_edges**,

double **lowThreshold**, double **highThreshold**, int **kernel_size** )

Parameter:

1) **detected_edges**: Source image, grayscale

2) **detected_edges**: Output of the detector (can be the same as the input)

3) **lowThreshold**: The value entered by the user moving the Trackbar

4) **highThreshold**: Set in the program as three times the lower threshold

(following Canny's recommendation)

5) **kernel_size**: We defined it to be 3 (the size of the Sobel kernel to be used internally)

Ex:

| | | |
|---|---|---|
| -1 | 0 | +1 |
| -2 | 0 | +2 |
| -1 | 0 | +1 |

| | | |
|---|---|---|
| +1 | +2 | +1 |
| 0 | 0 | 0 |
| -1 | -2 | -1 |

Kernel = window(表格) = filter

Input grayscale image

Canny

$C_t = canny\,(E_t)$

Output Edge image

$$C_t(x,y) = \begin{cases} 1, & \text{edge} \\ 0, & \text{no - edge} \end{cases}$$

accumulated magnitude

theta

$$C_t(x,y) = \begin{cases} D, & \text{edge } (1 \sim 255 \text{ or ex. } 32 \sim 255) \\ 0, & \text{no - edge} \end{cases}$$