

# Computer Vision Notes

*Huu Duc Nguyen M.Sc.*

29 March 2022

# Contents

<b>Abbreviations</b>	<b>1</b>
<b>1 Introduction</b>	<b>2</b>
<b>2 Mathematics Backgrounds</b>	<b>3</b>
2.1 The Matrix Equation . . . . .	3
<b>3 Image Formation</b>	<b>4</b>
3.1 Camera Obscura . . . . .	4
3.2 Pinhole Camera . . . . .	4
3.3 Digital image . . . . .	5
3.4 Color Sensing . . . . .	5
3.4.1 Demosaicing . . . . .	5
<b>4 Image Processing</b>	<b>7</b>
4.1 Linear Filters . . . . .	7
4.2 Background . . . . .	8
4.3 Non-Linear Filters . . . . .	9
4.4 Multi-Scale Representations . . . . .	9
4.5 Filters as Templates . . . . .	11
4.6 Image Gradients . . . . .	11
4.7 Edge Detection . . . . .	11
4.8 Structure Extraction . . . . .	11
<b>5 Segmentation</b>	<b>12</b>
<b>6 Object Detection</b>	<b>13</b>
<b>7 Local Feature</b>	<b>14</b>
<b>8 Deep Learning for CV</b>	<b>15</b>
8.1 Image-to-Image Translation . . . . .	15
8.1.1 pix2pix . . . . .	15
8.1.2 CycleGAN . . . . .	16
8.2 Neural Style Transfer . . . . .	17
8.2.1 Artistic Style Transfer . . . . .	17

8.2.2	Artistic Style Transfer for Videos . . . . .	18
8.2.3	Fast Artistic Style Transfer . . . . .	18
8.3	Super Resolution . . . . .	19
8.3.1	SRCNN . . . . .	19
8.3.2	SRGAN . . . . .	20
8.3.3	ESRGAN . . . . .	20
8.4	Code Examples . . . . .	20
<b>9</b>	<b>3D Computer Vision</b>	<b>21</b>
9.1	Introduction . . . . .	21
9.2	Depth Extraction . . . . .	21
9.3	3D Shape representation . . . . .	22
9.3.1	Voxel Grid . . . . .	22
9.3.2	Point Cloud . . . . .	22
9.3.3	Mesh . . . . .	22
9.3.4	Occupancy . . . . .	22
9.4	Classic 3D Reconstruction . . . . .	22
9.4.1	Epipolar Geometry . . . . .	23
9.4.2	Stereo Image Rectification . . . . .	23
9.4.3	Correspondence Search . . . . .	23
9.4.4	Stereo Reconstruction . . . . .	24
9.4.5	Camera Calibration . . . . .	24
9.4.6	Eight Point Algorithm . . . . .	24
9.5	Deep Learning for 3D CV . . . . .	24
<b>10</b>	<b>Single Object Tracking</b>	<b>25</b>
<b>11</b>	<b>Bayesian Filtering</b>	<b>26</b>
<b>12</b>	<b>Multi Object Tracking</b>	<b>27</b>
<b>13</b>	<b>Visual Odometry</b>	<b>28</b>
<b>14</b>	<b>SLAM</b>	<b>29</b>
<b>15</b>	<b>Deep Learning for Video Analysis</b>	<b>30</b>
<b>16</b>	<b>Research Proposal</b>	<b>31</b>
16.1	Transfer Learning . . . . .	31
	<b>Bibliography</b>	<b>I</b>



# Abbreviations

<b>info.</b>	information
<b>a.k.a.</b>	also known as
<b>no.</b>	number of
<b>func.</b>	function
<b>vs.</b>	versus
<b>freq.</b>	frequency
<b>i.i.d.</b>	independent & identically distributed
<b>LSI</b>	linear shift invariant
<b>SVD</b>	Singular Value Decomposition
<b>CNN</b>	Convolutional Neural Network
<b>GAN</b>	Generative Adversarial Network
<b>CGAN</b>	Conditional Generative Adversarial Network
<b>SRGAN</b>	Super Resolution Generative Adversarial Network
<b>ESRGAN</b>	Enhanced Super Resolution Generative Adversarial Network
<b>SSIM</b>	Structural Similarity Index
<b>SRCNN</b>	Super Resolution Convolutional Neural Network

# 1 Introduction

**Goal:** The goal of computer vision is enabling machine to understand images & videos. There are two major tasks:

- measurement: compute properties of 3D world (distance, shape)
- perception & interpretation: recognize objects, people, activities, ..

## **Outlines**

- Chap. 2 presents mathematics backgrounds for computer vision
- [TODO: Chap. ?? do sth]

## 2 Mathematics Backgrounds

This chapter presents some mathematics backgrounds.

### 2.1 The Matrix Equation

**Problem:** Solve  $Ax = 0$

- Applying Singular Value Decomposition ([SVD](#)) for matrix  $A$

$$A = U.D.V^T = U. \begin{bmatrix} d_{11} & \cdots & d_{1N} \\ \vdots & \ddots & \vdots \\ d_{N1} & \cdots & d_{NN} \end{bmatrix} . \begin{bmatrix} v_{11} & \cdots & v_{1N} \\ \vdots & \ddots & \vdots \\ v_{N1} & \cdots & v_{NN} \end{bmatrix}^T$$

- Solution of  $Ax = 0$  is the null space vector of  $A$ , which corresponds to the smallest (last) singular vector of  $A$ :  $[v_{1N}, \cdots, v_{NN}]^T$ .

# 3 Image Formation

## 3.1 Camera Obscura

also known as (a.k.a.) the "Dark Chamber" (Leonardo Da Vinci, 1545)

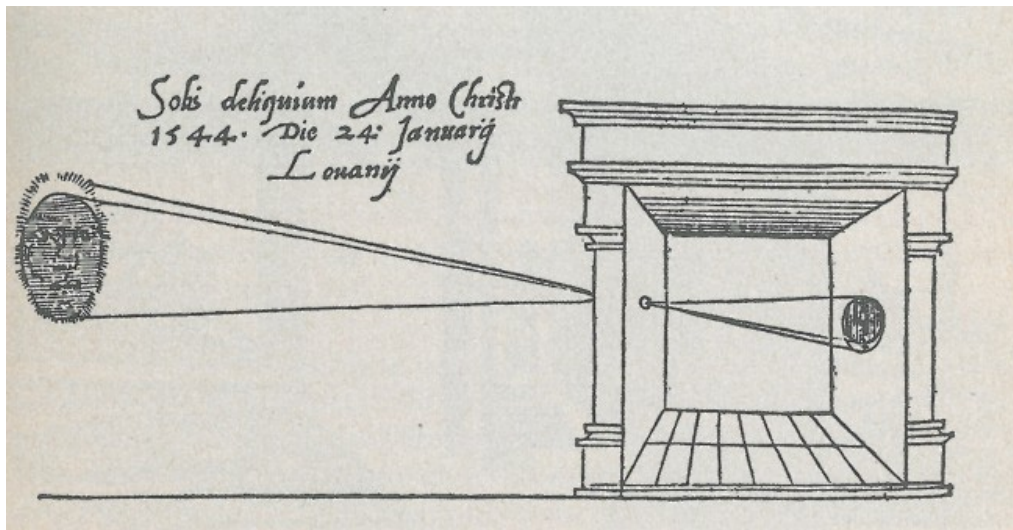
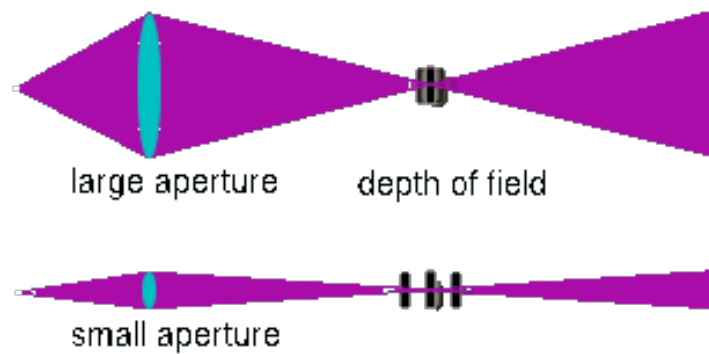


Figure 3.1: Camera obscura [Fri45].

## 3.2 Pinhole Camera

- Pinhole size = aperture
  - too big  $\Rightarrow$  blurring
  - too small  $\Rightarrow$  also blur, but because of diffraction
  - but then, ***image is dark***
- $\Rightarrow$  Use lenses: keep image sharp while ***capture more light***
- The thin lens
- Focus & Depth of Field:
  - Large aperture: small depth of field  
(only object within the correct distance will be at focus, while background is blur)
  - Small aperture: large depth of field, but need more light
- The lens focus  $f \gtrless$  field of view
  - $f$  gets smaller  $\Rightarrow$  wide-range image
  - $f$  gets greater  $\Rightarrow$  telescopic image



**Figure 3.2:** Varied depths of field depending on aperture size.

### 3.3 Digital image

- Discretize the image into a grid of pixels
- Quantize light intensities  $\Rightarrow$  pixel values
- Resolution: number of (no.) pixels (most commonly understand)

### 3.4 Color Sensing

Referring to the process of assigning pixel values from color information of world objects.

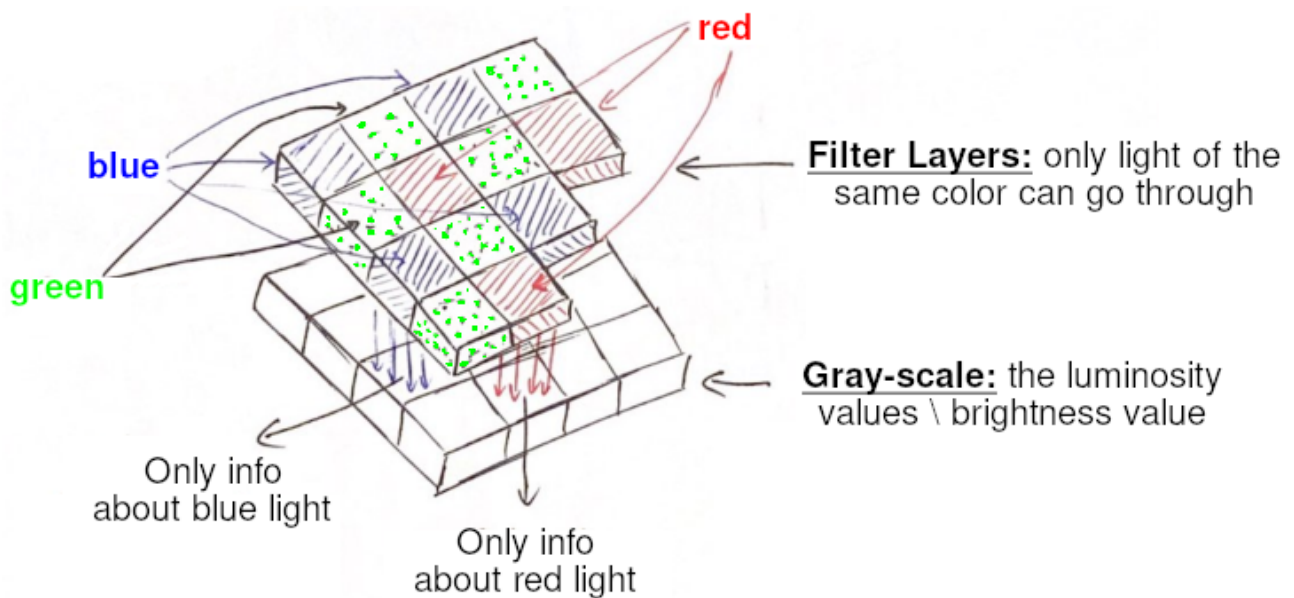
- Color image: RGB is just 1 of many color spaces, e.g., LUV, XYZ ([Wikipedia](#)).
- Grey-scale image

#### **3.4.1 Demosaicing**

Digital camera takes in light through a filter (Bayer or Xtrans)  $\Rightarrow$  we get a gray-scale image (Fig. 3.3). We need to apply demosaicing based on the filter's pattern to get the color image from the raw image. Sources: [YouTube](#), [Wikipedia](#).

**NOTE:** Raw image has a ***green cast***

Twice many green as red & blue, because human eyes are twice as sensitive to the green part to other red or blue part.



**Figure 3.3:** E.g. Bayer Filter. In the raw image , which lies below the filter layers, each pixel only has information (info.) of only 1 among 3 light sources. Demosaicing uses the values of surrounding pixels to infer the brightness of other light sources.

# 4 Image Processing

## 4.1 Linear Filters

Types of noise:

- Salt & pepper noise
- Impulse noise
- Gaussian noise

$$\text{noise} = \text{randn}(\text{size}(\text{img})) \times \sigma$$

$$\text{output} = \text{img} + \text{noise}$$

- **Basic assumption:** independent & identically distributed (i.i.d.)

Types of filter:

- Correlation Filter:  $G[i, j] = \frac{1}{(2k+1)^2} \sum_{u=-k}^k \sum_{v=-k}^k F[i+u, j+v]$

$$\text{different weights: } G[i, j] = \sum_{u=-k}^k \sum_{v=-k}^k H[u, v] F[i+u, j+v] \Rightarrow \boxed{G = H \otimes F}$$

with  $H[u, v]$  as non-uniform weights

**Matlab:** filter2, imfilter

- Convolution:  $G[i, j] = \sum_{u=-k}^k \sum_{v=-k}^k H[u, v] F[i-u, j-v] \Rightarrow \boxed{G = H * F}$

**Matlab:** conv2

**If  $H[u, v] = H[-u, -v] \Rightarrow \text{correlation} \equiv \text{convolution}$**

- Averaging Filter: **Ringing Artifacts??**

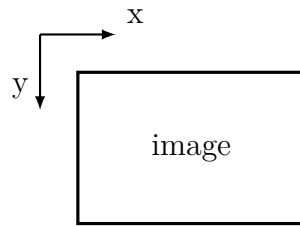
- Gaussian Filter:  $\frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$

**Rule of thumb:** set the filter width to  $6\sigma$

**More noise  $\Rightarrow \uparrow \sigma \Rightarrow \text{blurring effect}$**

**NOTE:**

- **$k$  is from the window size  $(2k+1) \times (2k+1)$**
- **Efficient implementation:** if filter is separable  $\Rightarrow$  apply 1D filter 2 times to have a 2D filter  $\Rightarrow$  Reduce the computational cost from  $\mathcal{O}(K^2)$  to  $\mathcal{O}(2K)$ , with  $K$  as the kernel size
- When coding with **Python**, the origin of image plane is top left corner,  $x$ -axis goes left,  $y$ -axis goes downward (Fig. 4.1)



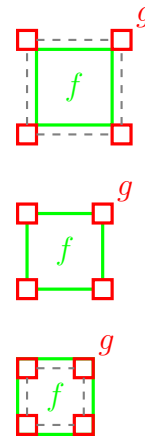
**Figure 4.1:** Image coordinate system in Python

- Boundary issues:

- Full:        output size =  $f + g$

- Same:       output size =  $f$

- Valid:       output size =  $f - g$



**Pixel near boundary:**

- Clip filter (black)  $\Rightarrow$  dark border
- Wrap around
- Copy edge  $\Rightarrow$  Strong edge response
- Reflect across edge
- Correlation versus (vs.) convolution:
  - Both are linear shift invariant linear shift invariant (LSI):
 
$$h \circ (f_0 + f_1) = h \circ f_1 + h \circ f_0$$
  - Conv is better, it has additional nice properties
    - \* commutative:  $f * g = g * f$
    - \* associative:  $(f * g) * h = f * (g * h)$
    - \* Fourier transform  $f * g \rightarrow F \cdot G$  and  $f \cdot h \rightarrow F * H$
  - With impulse image, Conv reproduces itself, while Corr reflects itself.

## 4.2 Background

- Taking the Fourier Transform of a signal  $\Rightarrow$  Frequency coefficients  $\Rightarrow$  **Frequency Spectrum**
- Duality:** The **better** a function is **localized** in one domain the **worse** it is **localized** in the other domain.

- Effect of Convolution:  $f * g \mapsto F \cdot G$   
 taking convolution in one domain is equivalent to multiplication in the other domain  
 A Gaussian has compact support in both domains  
 $\Rightarrow$  convenient choice for **low-pass filter**
- Sharpening filter (**high-pass filter**): emphasizes noise as well, since noise is high frequency (freq.) signal.

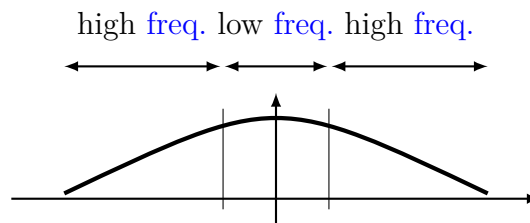


Figure 4.2: Frequency domain (Fourier).

### 4.3 Non-Linear Filters

- Median filter: replace each pixel by the median of the neighbors.
  - **remove spikes** (good for impulse, salt & pepper noise)
  - **edge preserving** (unlike mean filter)

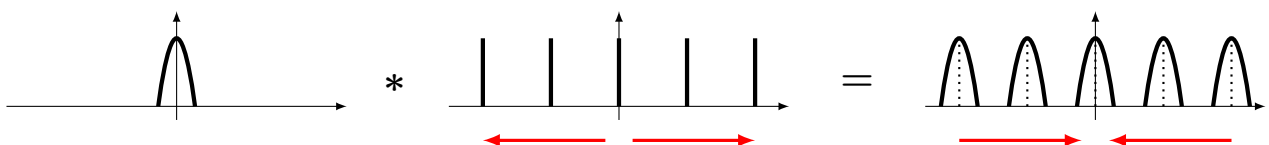
NOTE: If we increase the Median filter's filter size  $\Rightarrow$  reduce structure and loose details

### 4.4 Multi-Scale Representations

- Image pyramid: very little overhead (in terms of computational cost).
- Fourier Interpretation: Discrete Sampling  
 Sampling in spatial domain is like multiplying with a spike function (func.).



$\Rightarrow$  Sampling in the frequency domain is like convolving with a spike func.



$\Rightarrow$  when we sampling with lower **freq.**, the spikes will get further from each others. Due to duality in Sec. 4.2, the magnitude spectrum will be overlapped  $\Rightarrow$  we will not be able to reconstruct the original signal / data.

- **Nyquist theorem and limit:** to recover a certain **freq.**  $f$ , you have to take sample with at least with  $2f$ .

⇒ **Aliasing artifacts in Graphics:** overlapped signal (because sampling with too low frequency)

**NOTE:** We can't recover high **freq.** (edges), but we can avoid artifacts by prior smoothing before sampling.

- The Gaussian Pyramid: perform blurring & smoothing ⇒ then down-sampling [**TODO: Image**]
- The Laplacian Pyramid: [**TODO: Image**]

$$L_i = G_i - \text{expand}(G_{i+1})$$

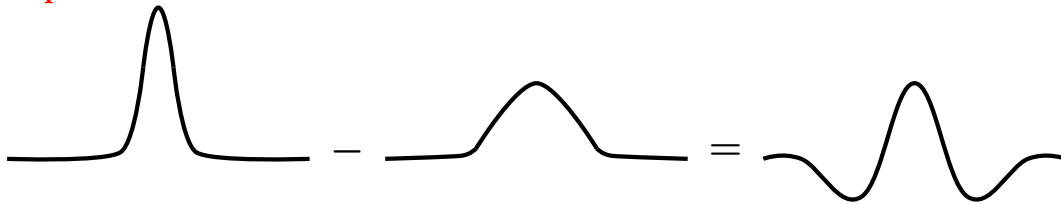
$$G_i = L_i + \text{expand}(G_{i+1})$$

$$L_n = G_n$$

⇒  $L_0 \rightarrow L_{n-1}$  contain **high freq. info.**

**NOTE:** Images in Laplacian Pyramid can be compressed further than the corresponding Gaussian Pyramid images.

- **Laplacian  $\sim$  Difference of Gaussians**



⇒ detect high-**freq.**  $\approx$  edges

The name Laplace ⇒ from a combinations of 2nd derivatives

Laplacian:  $\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$

$$\frac{\partial^2 f}{\partial x^2} = [f(x+1, y) - f(x, y)] - [f(x, y) - f(x-1, y)]$$

$$= f(x+1, y) + f(x-1, y) - 2f(x, y)$$

$$\Rightarrow \nabla^2 f = f(x \pm 1, y) + f(x, y \pm 1) - 4f(x, y)$$

0	1	0
1	-4	1
0	1	0

⇒ Laplacian filter:

## 4.5 Filters as Templates

Correlation filtering as Template Matching.

## 4.6 Image Gradients

- Differentiation & Convolution:  $\frac{\partial f(x, y)}{\partial x} \approx \frac{f(x+1, y) - f(x, y)}{1}$

$\Rightarrow$  Filter:  $\begin{bmatrix} 1 & -1 \end{bmatrix}$

**Problem:** it shifts the image

$\Rightarrow$  Prewitt, Sobel, Robert filters:

– Prewitt filter:  $\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}; \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$

– Sobel filter:  $M_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}; M_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$

– Robert  $\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}; \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$

- With noise, we need to smooth the image first

## 4.7 Edge Detection

## 4.8 Structure Extraction

[TODO: missing content]

# 5 Segmentation

[TODO: ]

## 6 Object Detection

[TODO: ]

## 7 Local Feature

[TODO: ]

# 8 Deep Learning for CV

[TODO: ]

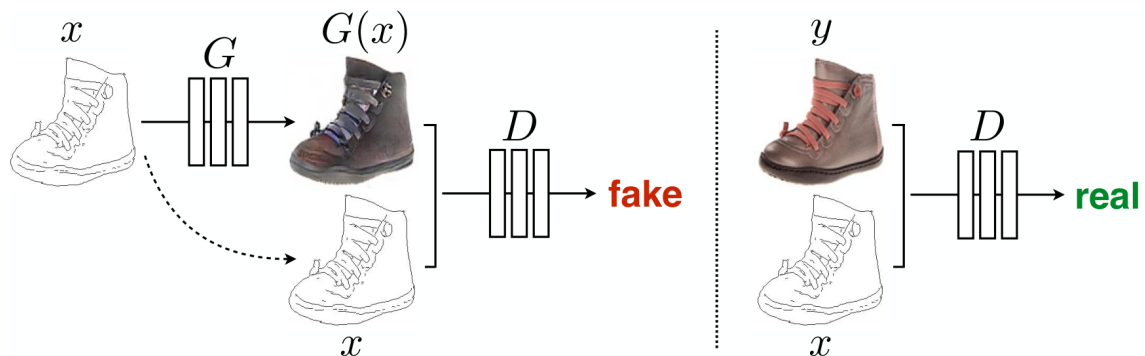
## 8.1 Image-to-Image Translation

Image-to-image translation is a class of computer vision problems where the goal is to learn the mapping between an input image and an output image. Recent approaches utilize Generative Adversarial Network (GAN). It has various applications [IZZ+17; ZPI+17], e.g.:

- Domain adaptation
- Semantic label  $\leftrightarrow$  photo
- Map  $\leftrightarrow$  aerial photo
- Edges  $\rightarrow$  photo
- BW  $\rightarrow$  color photos
- Day  $\rightarrow$  night
- Photo with missing pixels  $\rightarrow$  inpainted photo (recovering)

### 8.1.1 pix2pix

pix2pix uses Conditional Generative Adversarial Network (CGAN) idea with U-Net architecture [IZZ+17].



**Figure 8.1:** Training a CGAN to map edges  $\rightarrow$  photo. Both the discriminator and generator are conditioned on the input  $x$ . [IZZ+17]

The loss function of **pix2pix** combines **CGAN** objective with L1 distance with ground-truth images. L1 distance is prefer over L2 because L1 encourages less blurring effect.

$$\mathcal{L}_{GAN}(G, D) = \mathbb{E}_y[\log D(y)] + \mathbb{E}_z[\log(1 - D(G(z)))] \quad (8.1)$$

$$\mathcal{L}_{CGAN}(G, D) = \mathbb{E}_{x,y}[\log D(x, y)] + \mathbb{E}_{x,z}[\log(1 - D(x, G(x, z)))] \quad (8.2)$$

$$\mathcal{L}_{L1}(G) = \mathbb{E}_{x,y,z}[\|y - G(x, z)\|_1] \quad (8.3)$$

$$G^* = \arg \min_G \max_D \mathcal{L}_{CGAN}(G, D) + \lambda \mathcal{L}_{L1}(G) \quad (8.4)$$

**NOTE:** In implementation, the noise  $z$  is accounted as DropOut percentage.

### 8.1.2 CycleGAN

Cycle**GAN** addresses the problem when there is no available paired training data. By considering cycle consistency losses, it limits the mapping functions. [ZPI+17]

$$G : X \rightarrow Y \quad - \text{mapping from domain } X \text{ to domain } Y \quad (8.5)$$

$$F : Y \rightarrow X \quad - \text{mapping from domain } Y \text{ to domain } X \quad (8.6)$$

$$F(G(x)) \approx x \quad - \text{forward cycle consistency} \quad (8.7)$$

$$G(F(y)) \approx y \quad - \text{backward cycle consistency} \quad (8.8)$$

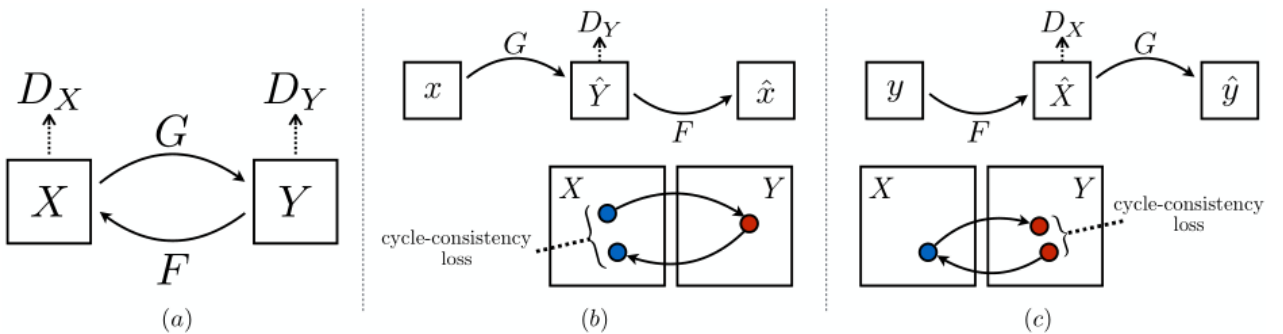
$$\mathcal{L}_{GAN_1}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{data}(y)}[\log D_Y(y)] + \mathbb{E}_{x \sim p_{data}(x)}[\log(1 - D_Y(G(x)))] \quad (8.9)$$

$$\mathcal{L}_{GAN_2}(F, D_X, X, Y) = \mathbb{E}_{x \sim p_{data}(x)}[\log D_X(x)] + \mathbb{E}_{y \sim p_{data}(y)}[\log(1 - D_X(F(y)))] \quad (8.10)$$

$$\mathcal{L}_{cyc}(G, F) = \mathbb{E}_{x \sim p_{data}(x)}[\|F(G(x)) - x\|_1] + \mathbb{E}_{y \sim p_{data}(y)}[\|G(F(y)) - y\|_1] \quad (8.11)$$

$$\mathcal{L}(G, F, D_X, D_Y) = \mathcal{L}_{GAN_1}(G, D_Y, X, Y) + \mathcal{L}_{GAN_2}(F, D_X, X, Y) + \lambda \mathcal{L}_{cyc}(G, F) \quad (8.12)$$

$$G^*, F^* = \arg \min_{G, F} \max_{D_X, D_Y} \mathcal{L}(G, F, D_X, D_Y) \quad (8.13)$$



**Figure 8.2:** Cycle**GAN** structure with 4 networks. [ZPI+17]

**NOTE:**

- The authors mention that experiment the cycle consistency loss as adversarial loss leads to no improved performance.
- CycleGAN's results are not significantly better than pix2pix's.
- Perform well on tasks relating color transformation (e.g. style transfer: picture  $\leftrightarrow$  paintings, horse  $\leftrightarrow$  zebra, winter  $\leftrightarrow$  summer), but not so good with geometric changes (dog  $\leftrightarrow$  cat).

## 8.2 Neural Style Transfer

Style transfer is similar to image-to-image translation, but doesn't require a dataset from each style. It instead runs an iterative optimization procedure on two given images.

### 8.2.1 Artistic Style Transfer

The first work is by Gatys, Ecker, and Bethge (2015) [GEB15]. The authors manage to separate image content and image style. Given a Convolutional Neural Network (CNN), at the  $l^{th}$  layer, there is  $N_l$  distinct filters, thus, leads to  $N_l$  feature maps of size  $M_l$ .

- The image content is represented in matrix  $F^l \in \mathcal{R}^{N_l \times M_l}$ , which is the concatenation of these feature maps.  $F_{ij}^l$  is the activation of the  $i^{th}$  filter at position  $j$  in  $l^{th}$  layer. The authors prove this by trying to reconstruct the image from these feature maps.

$$\vec{p} \quad \text{— original image} \quad (8.14)$$

$$\vec{x} \quad \text{— generated image} \quad (8.15)$$

$$F_{ij}^l \quad \text{— the original image's content} \quad (8.16)$$

$$P_{ij}^l \quad \text{— the generated image's content} \quad (8.17)$$

$$\mathcal{L}_{content}(\vec{p}, \vec{x}, l) = \frac{1}{2} \sum_{i,j} (F_{ij}^l - P_{ij}^l)^2 \quad \text{— the content loss} \quad (8.18)$$

- The image style is represented in the Gram matrix  $G^l \in \mathcal{R}^{N_l \times N_l}$ , where  $G_{ij}^l$  is the correlation between feature map in the  $l^{th}$  layer:

$$G_{ij}^l = \sum_k F_{ik}^l F_{jk}^l \quad (8.19)$$

$$\vec{a} \quad \quad \quad - \text{artwork} \quad \quad \quad (8.20)$$

$$\vec{x} \quad \quad \quad - \text{generated image} \quad \quad \quad (8.21)$$

$$A^l \quad \quad \quad - \text{the artwork's style representation} \quad \quad \quad (8.22)$$

$$G^l \quad \quad \quad - \text{the generated image's style representation} \quad \quad \quad (8.23)$$

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} (G_{ij}^l - A_{ij}^l)^2 \quad \quad \quad - \text{style representation loss at } l^{th} \text{ layer} \quad \quad \quad (8.24)$$

$$\mathcal{L}_{style}(\vec{a}, \vec{x}) = \sum_{l=0}^L w_l E_l \quad \quad \quad - \text{the style loss} \quad \quad \quad (8.25)$$

$$\mathcal{L}_{total}(\vec{p}, \vec{a}, \vec{x}) = \alpha \mathcal{L}_{content}(\vec{p}, \vec{x}) + \beta \mathcal{L}_{style}(\vec{a}, \vec{x}) \quad \quad \quad - \text{total loss} \quad \quad \quad (8.26)$$

The algorithm applies gradient descent to minimize the above loss with  $\vec{x}$  as a white noise image in the beginning.

### 8.2.2 Artistic Style Transfer for Videos

Applying the above approach to video leads to terribly inconsistent results. Ruder, Dosovitskiy, and Brox (2016) [RDB16] improve by adding additional improvements:

- Short-term consistency by initialization: Estimate the optical flow between image  $p^{(i)}$  and  $p^{(i+1)}$ . The generated image  $x^{(i+1)}$  will not be initialized with a white noise image, but a warped image from the previous one:  $x'^{(i+1)} = \omega_i^{i+1}(x^{(i)})$ . Here  $\omega_i^{i+1}$  denotes the warping function using the estimated optical flow.
- Temporal consistency loss
- Long-term consistency
- Multi-pass algorithm

### 8.2.3 Fast Artistic Style Transfer

The above approaches for style transfer require an iterative optimization process for each image. Johnson, Alahi, and Fei-Fei (2016) [JAFF16] propose a training pipeline to simplify this procedure. By learning a network that minimize the same loss, the output now requires only one single run. It does lose some of the temporal consistency when applying to videos, but it's running in real-time.

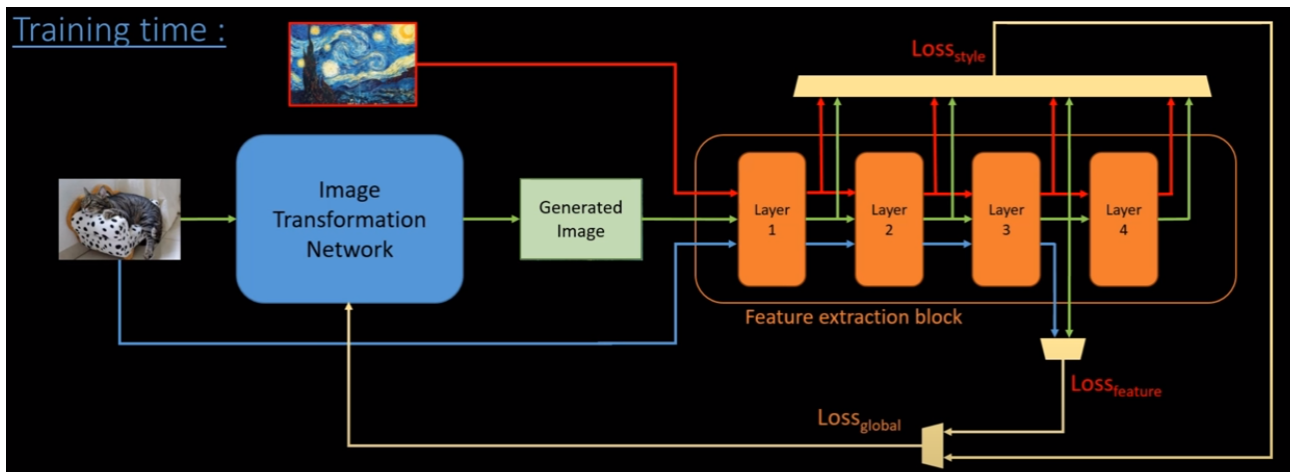


Figure 8.3: Training pipeline (src) [JAFF16]

## 8.3 Super Resolution

Youtube: [How Super Resolution Works](#)

- Use Structural Similarity Index (SSIM) [WB09; WBS+04]

### 8.3.1 SRCNN

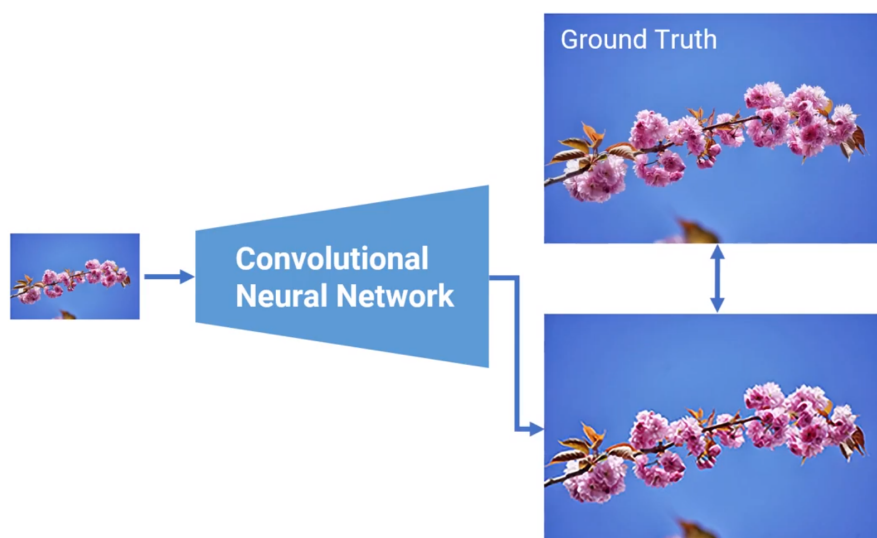


Figure 8.4: Super Resolution Convolutional Neural Network (SRCNN) training [DLH+15]



**Figure 8.5:** Super Resolution Generative Adversarial Network (SRGAN) training [LTH+17]

### 8.3.2 SRGAN

### 8.3.3 ESRGAN

Enhanced Super Resolution Generative Adversarial Network (ESRGAN) [WYW+18]

- Remove Batch Normalization
- More layers and connections: residual scaling
- Modify the VGG loss
- Relativistic discriminator

## 8.4 Code Examples

- [Tensorflow's tutorial: pix2pix](#)
- [Tensorflow's tutorial: CycleGAN](#)
- [Github source code: Artistic Style Transfer for Videos](#)
- [Tensorflow's tutorial: Neural style transfer](#)
- [Tensorflow's tutorial: Fast Style Transfer](#)

# 9 3D Computer Vision

## 9.1 Introduction

3D Computer Vision gives a representation that is closer of things that we interact in our lives. Thus, it will empower various novel applications in:

- Autonomous Driving
- Robotics
- Remote Sensing
- Medical Treatment
- Design Industry
- Augmented Reality

[**TODO:** ] Learning resources: [??](#).

3D computer vision problems includes:

- Depth extraction
- 3D Reconstruction
- Object Classification
- Object Detection
- Object Segmentation
- ??

Challenges of 3D computer vision:

- something here

## 9.2 Depth Extraction

***The goal:*** extract the depth, as the 3rd dimension for a 2D image.

The depth map is a simple grey image with values in range  $[0, 255]$ , 0 for point afar and 255 for points in near distances.



**Figure 9.1:** Example of a depth map [TSS+18].

### 9.3 3D Shape representation

There are explicit representations and implicit representations, where parametric functions are used to differentiate a specific point is inside or outside the shape, or the distance to the shape surface. Typically, the parametric functions are in form of neural networks

#### 9.3.1 Voxel Grid

#### 9.3.2 Point Cloud

#### 9.3.3 Mesh

#### 9.3.4 Occupancy

### 9.4 Classic 3D Reconstruction

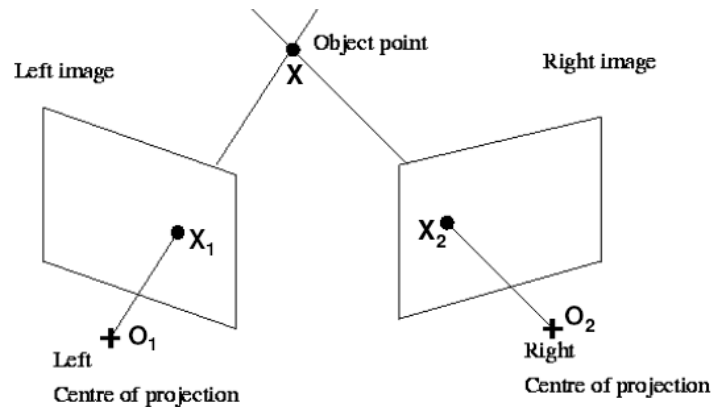
Geometric vision:

- Visual Cues (Details)
  - Shading
  - Texture
  - Focus
  - Perspective
  - Motion

- Stereo vision: process of extracting 3D information from multiple 2D views of a scene

### 9.4.1 Epipolar Geometry

Epipolar geometry is the geometry of stereo vision. The **basic principle** of epipolar geometry is **triangulation** of points. In Fig. 9.2,  $O_1$  and  $O_2$  are the camera poses,  $X_1$  and  $X_2$  are the



**Figure 9.2:** Example of triangulation ([src](#)). The lines connecting the camera poses with the correspondent points must intersect at the real object world space.

correspondent points on each image planes, and  $X$  is the real object point in world space.

[TODO: ]

### 9.4.2 Stereo Image Rectification

Re-project image planes on to a common plane, which is parallel to the baseline

⇒ Scan lines are epipolar lines.

[TODO: Add images]

### 9.4.3 Correspondence Search

Correspondence search simple means matching a point with another point in a different image.

**NOTE:** In practice, use both.

Dense Correspondence Search	Sparse Correspondence Search
<ul style="list-style-type: none"> <li>For <b>each pixel</b>, find correspondence</li> <li>Easy when epipolar lines are scan lines (apply <b>rectification</b>)</li> </ul>	<ul style="list-style-type: none"> <li>Only for a set of detected feature</li> <li>Use feature description (Harris, SIFT??)</li> </ul>
— Pros —	
<ul style="list-style-type: none"> <li><b>Simple</b> process</li> <li><b>More depth</b> <math>\Rightarrow</math> useful for surface reconstruction</li> </ul>	<ul style="list-style-type: none"> <li><b>Efficiency</b></li> <li>Can have more reliable matches</li> <li>Less sensitive to illumination <math>\Rightarrow</math> <b>robust</b></li> </ul>
— Cons —	
Problem with: <ul style="list-style-type: none"> <li><b>texture-less regions</b></li> <li>different <b>viewpoints</b></li> </ul>	<ul style="list-style-type: none"> <li>Have to know enough to pick good features</li> <li><b>Sparse</b> information</li> </ul>

#### 9.4.4 Stereo Reconstruction

Main steps:

- Calibrate cameras
- Rectify images
- Compute disparity
- Estimate depth

**This is just the ideal case.**

- What if, how can we get extrinsic **info.** from calibration?
- What to do when triangulation failed?

#### 9.4.5 Camera Calibration

#### 9.4.6 Eight Point Algorithm

### 9.5 Deep Learning for 3D CV

# 10 Single Object Tracking

[TODO: ]

# 11 Bayesian Filtering

[TODO: ]

# 12 Multi Object Tracking

[TODO: ]

# 13 Visual Odometry

[TODO: ]

# 14 SLAM

[TODO: ]

# 15 Deep Learning for Video Analysis

[TODO: ]

# 16 Research Proposal

There are levels of visual understanding

1. Object detection/classification
2. Detection of the state of an objects
3. Detection of the relationships/interactions/compositions of objects
4. Reasoning

## 16.1 Transfer Learning

# Bibliography

- [DLH+15] C. Dong, C. C. Loy, K. He, and X. Tang. “Image super-resolution using deep convolutional networks”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38.2 (2015), pp. 295–307.
- [Fri45] R. G. Frisius. *De radio astronomico et geometrico liber*. Ap. Gul Cavellat, 1545.
- [GEB15] L. A. Gatys, A. S. Ecker, and M. Bethge. “A neural algorithm of artistic style”. In: *arXiv preprint arXiv:1508.06576* (2015).
- [IZZ+17] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. “Image-to-image translation with conditional adversarial networks”. In: *Proc. of the IEEE/CVF Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 1125–1134.
- [JAFF16] J. Johnson, A. Alahi, and L. Fei-Fei. “Perceptual losses for real-time style transfer and super-resolution”. In: *European Conference on Computer Vision*. Springer. 2016, pp. 694–711.
- [LTH+17] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, et al. “Photo-realistic single image super-resolution using a generative adversarial network”. In: *Proc. of the IEEE/CVF Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 4681–4690.
- [RDB16] M. Ruder, A. Dosovitskiy, and T. Brox. “Artistic style transfer for videos”. In: *German Conference on Pattern Recognition*. Springer. 2016, pp. 26–36.
- [TSS+18] H. Tjaden, U. Schwanecke, E. Schömer, and D. Cremers. “A Gauss-Newton Approach to Real-Time Monocular Multiple Object Tracking”. In: (July 2018).
- [WB09] Z. Wang and A. C. Bovik. “Mean squared error: Love it or leave it? A new look at signal fidelity measures”. In: *IEEE Signal Processing Magazine* 26.1 (2009), pp. 98–117.
- [WBS+04] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. “Image quality assessment: from error visibility to structural similarity”. In: *IEEE Transactions on Image Processing* 13.4 (2004), pp. 600–612.
- [WYW+18] X. Wang, K. Yu, S. Wu, J. Gu, Y. Liu, C. Dong, Y. Qiao, and C. Change Loy. “ESRGANEnhanced super-resolution generative adversarial networks”. In: *Proc. of the European Conference on Computer Vision (ECCV)*. 2018, pp. 0–0.

## *Bibliography*

- [ZPI+17] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. “Unpaired image-to-image translation using cycle-consistent adversarial networks”. In: *Proc. of the IEEE Int. Conf. on Computer Vision (ICCV)*. 2017, pp. 2223–2232.