Computer Vision Notes

Huu Duc Nguyen M.Sc.

29 March 2022

Contents

Abbreviations

CV	Computer Vision
info.	information
a.k.a.	also known as
no.	number of
func.	function
vs.	versus
freq.	frequency
i.i.d.	independent & identically distributed
LSI	linear shift invariant
SVD	Singular Value Decomposition
CNN	Convolutional Neural Network
GAN	Generative Adversarial Network
CGAN	Conditional Generative Adversarial Network
SRGAN	Super Resolution Generative Adversarial Network
ESRGAN	Enhanced Super Resolution Generative Adversarial Network
BatchNorm	Batch Normalization
AdaIN	Adaptive Instance Normalization
SSIM	Structural Similarity Index
SRCNN	Super Resolution Convolutional Neural Network
PPL	Perceptual path length
FID	Fréchet inception distance

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)

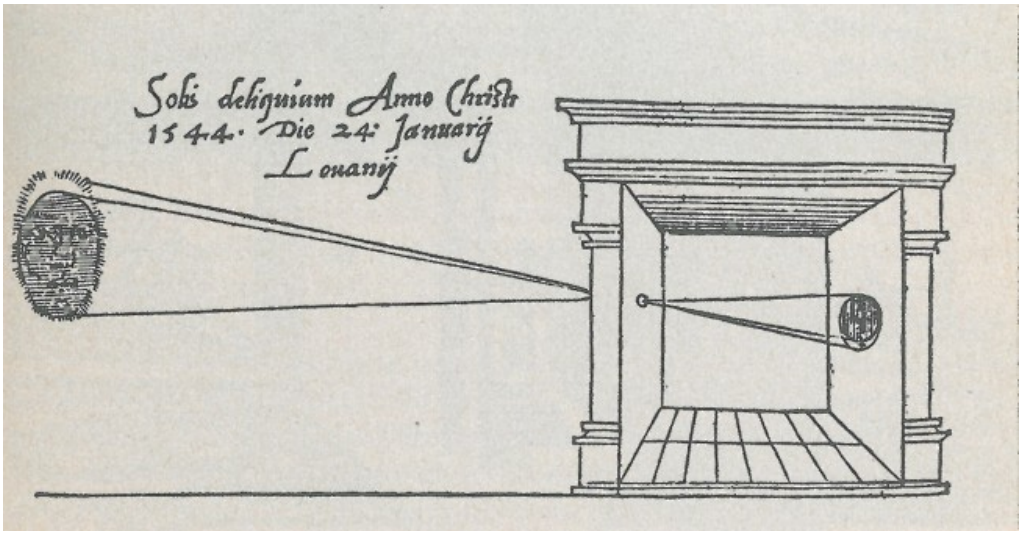


Figure 3.1: Camera obscura [Fri45].

3.2 Pinhole Camera

- Pinhole size = aperture
  - too big ⇒ blurring
  - too small ⇒ also blur, but because of diffraction
    but then, ***image is dark***

  ⇒ 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 ⇒ wide-range image
  - f gets greater ⇒ telescopic image

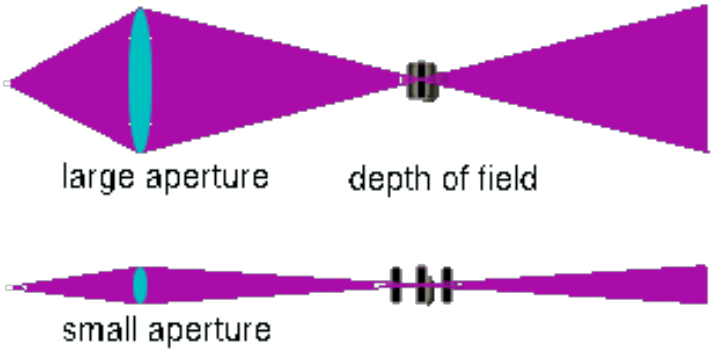


Figure 3.2: Varied depths of field depending on aperture size.

3.3 Digital image

- Discretize the image into a grid of pixels
- Quantize light intensities ⇒ 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) ⇒ 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.

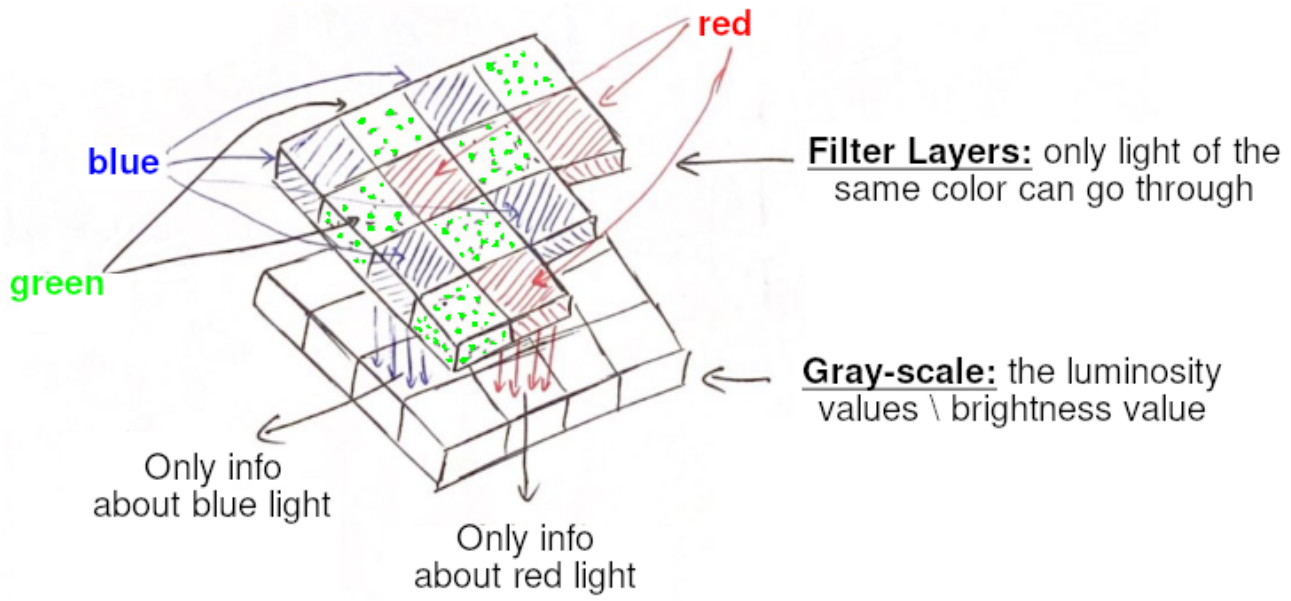


Figure 3.3: E.g. Bayer Filter. In the raw image , which lies below the filter layers, each pixel only has information (info.) of only 1 among 3 light sources. Demosaicing uses the values of surrounding pixels to infer the brightness of other light sources.

4 Image Processing

4.1 Linear Filters

Types of noise:

- Salt & pepper noise
- Impulse noise
- Gaussian noise
  $noise = randn(size(img)) \times \sigma$
  $output = img + noise$
- **Basic assumption:** independent & identically distributed (i.i.d.)

Types of filter:

- Correlation Filter: $G[i,j] = \frac{1}{(2k+1)^2} \sum_{u=-k}^{k} \sum_{v=-k}^{k} F[i+u, j+v]$
  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 $H[u,v] = H[-u,-v] \Rightarrow$ correlation $\equiv$ 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 6σ
  More noise $\Rightarrow \uparrow \sigma \Rightarrow$ 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)

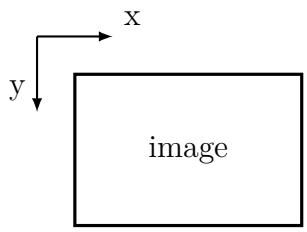
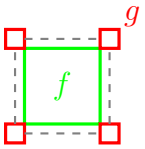


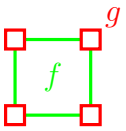
Figure 4.1: Image coordinate system in `Python`

- Boundary issues:

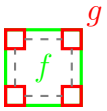
  - Full: output size = $f + g$

  - Same: output size = f

  - Valid: output size = $f - g$

  Pixel near boundary:
  - Clip filter (black) ⇒ dark border
  - Wrap around
  - Copy edge ⇒ 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 $f * g \multimap F.G$ and $f.h \multimap F * H$
  - With impulse image, Conv reproduces itself, while Corr reflects itself.

4.2 Background

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

- Effect of Convolution: $f * g \multimap F \cdot G$
  taking convolution in one domain is equivalent to multiplication in the other domain
  A Guassian 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.

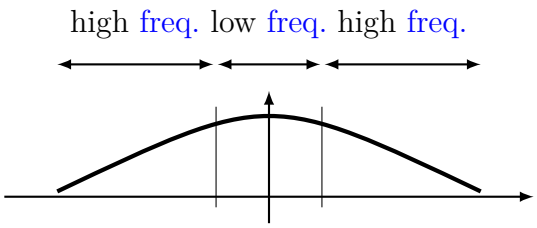


Figure 4.2: Frequency domain (Fourier).

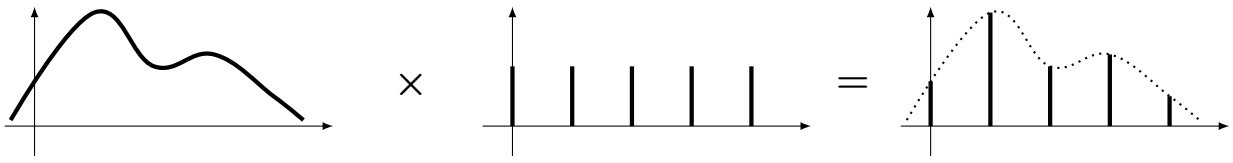
4.3 Non-Linear Filters

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

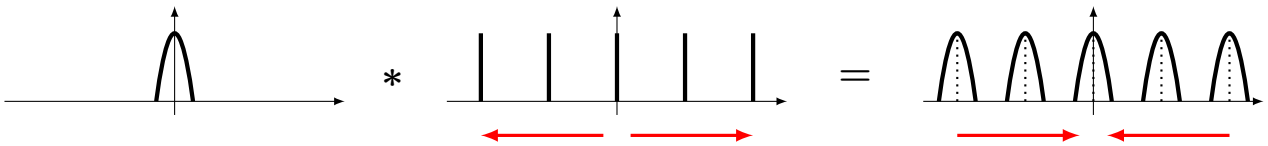
  NOTE: If we increase the Median filter's filter size $\Rightarrow$ reduce structure and loose details

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.***.

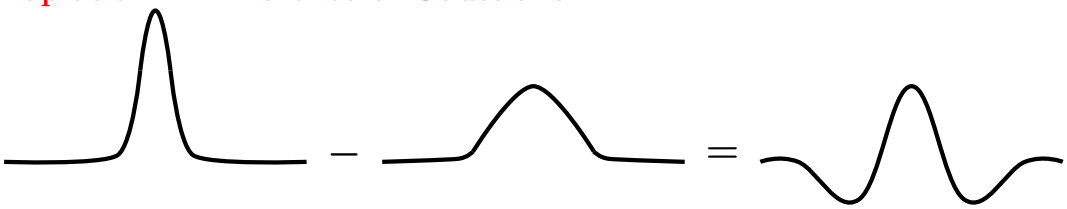


$\Rightarrow$ when we sampling with lower freq., the spikes will get further from each others. Due to duality in Sec. 4.2, the magnitude spectrum will be overlapped $\Rightarrow$ we will not be able to reconstruct the original signal / data.

- ***Nyquist theorem and limit:*** to recover a certain freq. f, you have to take sample with at least with $2f$.
  $\Rightarrow$ **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 $\Rightarrow$ then down-sampling [**TODO: Image**]
- The Laplacian Pyramid: [**TODO: Image**]

$$L_i = G_i - expand(G_{i+1})$$
$$G_i = L_i + expand(G_{i+1})$$
$$L_n = G_n$$

  $\Rightarrow L_0 \to L_{n-1}$ contain ***high freq. info.***
  NOTE: Images in Laplacian Pyramid ***can be compressed further*** than the corresponding Gaussian Pyramid images.
- **Laplacian $\sim$ Difference of Gaussians**



  $\Rightarrow$ detect high-freq. $\approx$ edges
  The name Laplace $\Rightarrow$ from a combinations of 2nd derivatives
  Laplacian: $\boxed{\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}}$

$$\begin{aligned}\frac{\partial^2 f}{\partial x^2} &= [f(x+1,y) - f(x,y)] - [f(x,y) - f(x-1,y)] \\ &= f(x+1,y) + f(x-1,y) - 2f(x,y) \\ \Rightarrow \nabla^2 f &= f(x \pm 1, y) + f(x, y \pm 1) - 4f(x,y)\end{aligned}$$

  $\Rightarrow$ ***Laplacian filter:***

0	1	0
1	-4	1
0	1	0

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 Loss Functions

L1-L2 loss doesn't work well with different Computer Vision (CV) problems [WB09]. This section proposes some alternatives

8.1.1 SSIM

Structural Similarity Index (SSIM) measures the similarity between two images. [WBS+04]

- The SSIM index is calculated on various windows of an image.
- Formula: Given two windows $x, y \in \mathbb{R}^{N \times N}$ (check Wikipedia)

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{8.1}$$

$$\text{SSIM}(x, y) = luminance(x, y) \cdot contrast(x, y) \cdot structure(x, y) \tag{8.2}$$

- It is used in super-resolution problem.

8.1.2 Perceptual Loss

- Perceptual Loss, a.k.a. content loss or VGG loss, is the loss between the feature maps of two images, after passing through a few pretrained Convolutional Neural Network (CNN) layers (Fig. 8.1).
- This loss is used in super-resolution problem

8.1.3 FID

In CV, Fréchet inception distance (FID) is used to measure the distance between two distributions of two datasets, which are usually the original and the generated image datasets. [HRU+17]

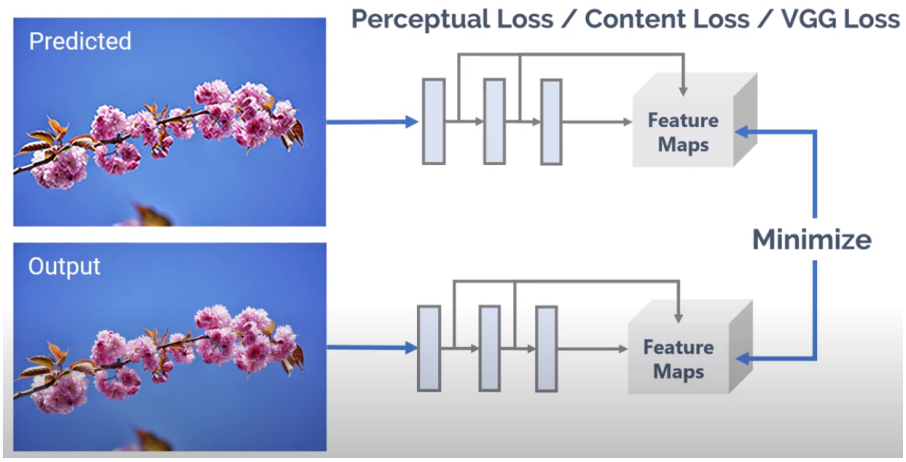


Figure 8.1: Perceptual loss is the loss between the feature maps, which are from passing images through a few frozen CNN layers from VGG network.

- Pass the images through the pretrained Inception network, take the output right before the last layer.
- Calculate the mean and covariance matrices: $(\boldsymbol{m}, \boldsymbol{C})$ and $(\boldsymbol{m}_w, \boldsymbol{C}_w)$

$$d^2((\boldsymbol{m}, \boldsymbol{C}), (\boldsymbol{m}_w, \boldsymbol{C}_w)) = ||\boldsymbol{m} - \boldsymbol{m}_w||_2^2 + \mathrm{Tr}(\boldsymbol{C} + \boldsymbol{C}_w - 2(\boldsymbol{C}\boldsymbol{C}_w)^{1/2}) \tag{8.3}$$

- The smaller the FID value the better the generated images, obviously.

8.1.4 PPL

The high-level idea is such, when interpolation in the latent space may lead to non-linear changes in the image. Perceptual path length (PPL) is a metric to measure the perceptually-based pairwise image distance. [KLA19]

$$l_{\mathcal{Z}} = \mathbb{E}\left[\frac{1}{\epsilon^2} d\left(G(\mathrm{slerp}(\mathbf{z}_1, \mathbf{z}_2; t)), G(\mathrm{slerp}(\mathbf{z}_1, \mathbf{z}_2; t + \epsilon))\right)\right] \quad \text{average in } \mathcal{Z} \text{ space} \tag{8.4}$$

$$l_{\mathcal{W}} = \mathbb{E}\left[\frac{1}{\epsilon^2} d\left(g(\mathrm{lerp}(f(\mathbf{z}_1), f(\mathbf{z}_2); t)), g(\mathrm{lerp}(f(\mathbf{z}_1), f(\mathbf{z}_2); t + \epsilon))\right)\right] \quad \text{average in } \mathcal{W} \text{ space} \tag{8.5}$$

- Latent vectors $\mathbf{z}_1, \mathbf{z}_2 \sim P(\mathbf{z})$
- Interpolation functions: slerp and lerp.
  Basically, they also output a latent vector that lie on the interpolation path from $\mathbf{z}_1$ to $\mathbf{z}_2$

$$\mathbf{z}_t = \mathrm{slerp}(\mathbf{z}_1, \mathbf{z}_2; t) \quad \text{with } t \sim U(0, 1)$$

- G is the generator, g is the synthesis network, f is the mapping network, thus $G = g \circ f$ and $image = G(\mathbf{z})$ (Fig. 8.8a)
- d evaluates the perceptual loss given two generated images (Subsec. 8.1.2)

8.2 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 ↔ photo
- Map ↔ aerial photo
- Edges → photo
- BW → color photos
- Day → night
- Photo with missing pixels → inpainted photo (recovering)

8.2.1 pix2pix

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

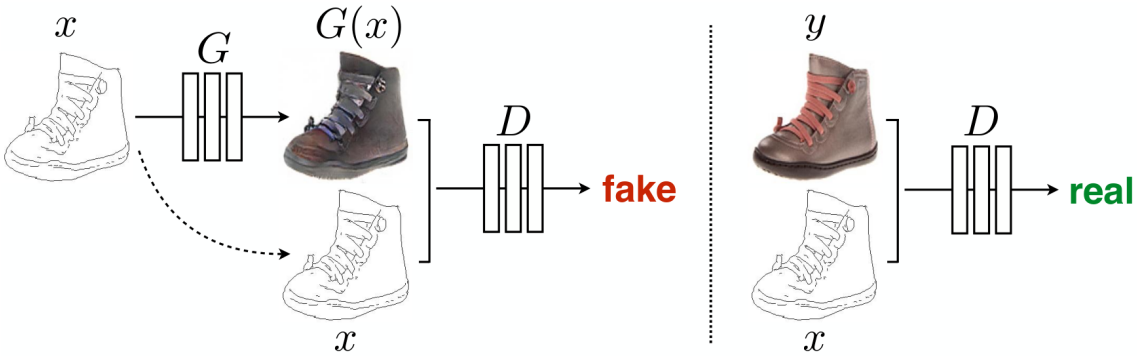


Figure 8.2: Training a CGAN to map edges → photo. Both the discriminator and generator are conditioned on the input x. [IZZ+17]

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.

$$\mathcal{L}_{GAN}(G,D) = \mathbb{E}_y[\log D(y)] + \mathbb{E}_z[\log(1 - D(G(z)))] \tag{8.6}$$
$$\mathcal{L}_{CGAN}(G,D) = \mathbb{E}_{x,y}[\log D(x,y)] + \mathbb{E}_{x,z}[\log(1 - D(x,G(x,z)))] \tag{8.7}$$
$$\mathcal{L}_{L1}(G) = \mathbb{E}_{x,y,z}\Big[||y - G(x,z)||_1\Big] \tag{8.8}$$
$$G* = \arg\min_G \max_D \mathcal{L}_{CGAN}(G,D) + \lambda \mathcal{L}_{L1}(G) \tag{8.9}$$

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

8.2.2 CycleGAN

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

$$G : X \to Y \quad \text{– mapping from domain } X \text{ to domain } Y \tag{8.10}$$
$$F : Y \to X \quad \text{– mapping from domain } Y \text{ to domain } X \tag{8.11}$$
$$F(G(x)) \approx x \quad \text{– forward cycle consistency} \tag{8.12}$$
$$G(F(y)) \approx y \quad \text{– backward cycle consistency} \tag{8.13}$$

$$\mathcal{L}_{GAN_1}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{data}(y)}[\log D_Y(y)] + \mathbb{E}_{x \sim p_{data}(x)}[\log(1 - D_Y(G(x)))] \tag{8.14}$$
$$\mathcal{L}_{GAN_2}(F, D_X, X, Y) = \mathbb{E}_{x \sim p_{data}(x)}[\log D_X(x)] + \mathbb{E}_{y \sim p_{data}(y)}[\log(1 - D_X(F(y)))] \tag{8.15}$$
$$\mathcal{L}_{cyc}(G,F) = \mathbb{E}_{x \sim p_{data}(x)}\Big[||F(G(x)) - x||_1\Big] + \mathbb{E}_{y \sim p_{data}(y)}\Big[||G(F(y)) - y||_1\Big] \tag{8.16}$$
$$\mathcal{L}(G, F, D_X, D_Y) = \mathcal{L}_{GAN_1}(G, D_Y, X, Y) + \mathcal{L}_{GAN_2}(F, D_X, X, Y) + \lambda \mathcal{L}_{cyc}(G,F) \tag{8.17}$$
$$G^*, F^* = \arg\min_{G,F} \max_{D_X, D_Y} \mathcal{L}(G, F, D_X, D_Y) \tag{8.18}$$

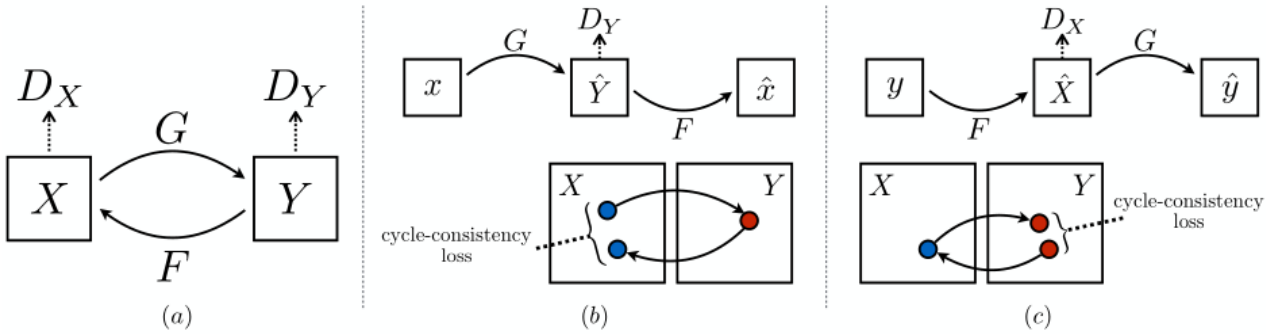


Figure 8.3: CycleGAN structure with 4 networks. [ZPI+17]

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 ↔ paintings, horse ↔ zebra, winter ↔ summer), but not so good with ***geometric changes*** (dog ↔ cat).

8.3 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.

8.3.1 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 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^l_{ij} 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.

$$\vec{p} \quad \text{– original image} \tag{8.19}$$

$$\vec{x} \quad \text{– generated image} \tag{8.20}$$

$$F^l_{ij} \quad \text{– the original image's content} \tag{8.21}$$

$$P^l_{ij} \quad \text{– the generated image's content} \tag{8.22}$$

$$\mathcal{L}_{content}(\vec{p}, \vec{x}, l) = \frac{1}{2} \sum_{i,j} \left(F^l_{ij} - P^l_{ij} \right)^2 \quad \text{– the content loss} \tag{8.23}$$

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

$$G^l_{ij} = \sum_k F^l_{ik} F^l_{jk} \tag{8.24}$$

$$\vec{a} \quad \text{– artwork} \tag{8.25}$$

$$\vec{x} \quad \text{– generated image} \tag{8.26}$$

$$A^l \quad \text{– the artwork's style representation} \tag{8.27}$$

$$G^l \quad \text{– the generated image's style representation} \tag{8.28}$$

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} \left(G_{ij}^l - A_{ij}^l\right)^2 \quad \text{– style representation loss at } l^{th} \text{ layer} \tag{8.29}$$

$$\mathcal{L}_{style}(\vec{a}, \vec{x}) = \sum_{l=0}^{L} w_l E_l \quad \text{– the style loss} \tag{8.30}$$

$$\mathcal{L}_{total}(\vec{p}, \vec{a}, \vec{x}) = \alpha \mathcal{L}_{content}(\vec{p}, \vec{x}) + \beta \mathcal{L}_{style}(\vec{a}, \vec{x}) \quad \text{– total loss} \tag{8.31}$$

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

8.3.2 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 ω_i^{i+1} denotes the warping function using the estimated optical flow.
- Temporal consistency loss
- Long-term consistency
- Multi-pass algorithm

8.3.3 Fast Artistic Style Transfer

The above style transfer approaches require an iterative optimization process for each pair of images. 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.

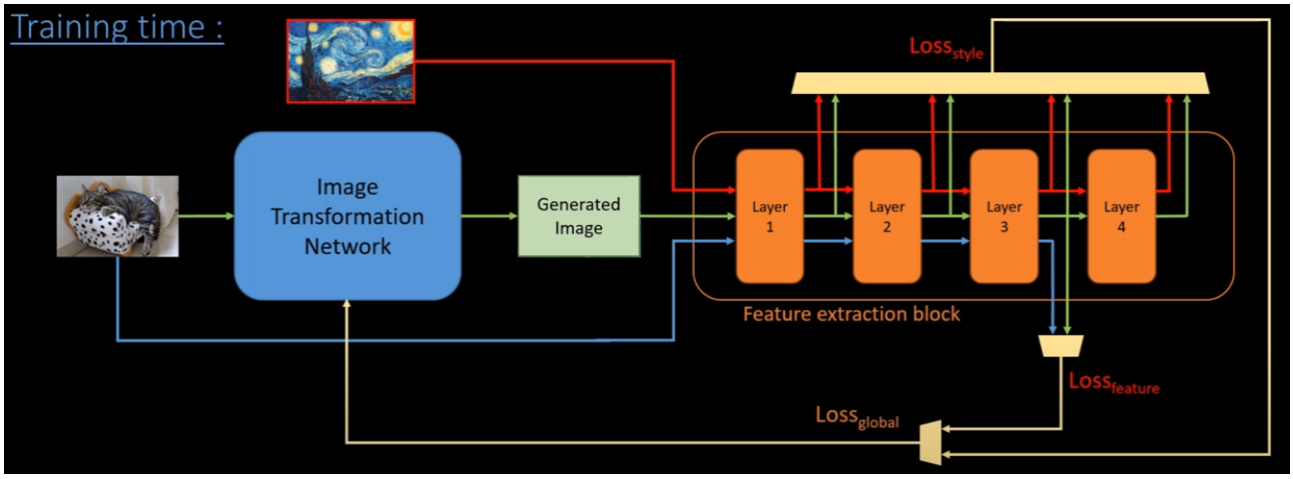


Figure 8.4: Training pipeline (src) [JAFF16]

+Running in real-time
−Loss some temporal consistency when applying to videos
−One single style for a network

8.3.4 Arbitrary Style-Transfer in Real-time

Huang and Belongie (2017) [HB17] improves prior methods::

- Using Adaptive Instance Normalization (AdaIN)
- Pros: +Running in real-time
  +A single network capable of transfer multiple arbitrary styles

Check their CVF presentation video.

8.4 Super Resolution

Super Resolution is a CV problem, in which we want to upscale a lower resolution image to a higher resolution one.

- Youtube: How Super Resolution Works
- Naive approach: nearest neighbor would simply spread the pixels out, then fill in the gap by copying values.
- Taking average, median would lead to better result, which is what bilinear and bicubic interpolation do.

8.4.1 SRCNN

Dong et al. (2015) [DLH+15]'s model is a CNN that (Fig. 8.5)

- takes in a lower resolution image and output a higher resolution image
- the loss is simply squared error between the output with ground truth image

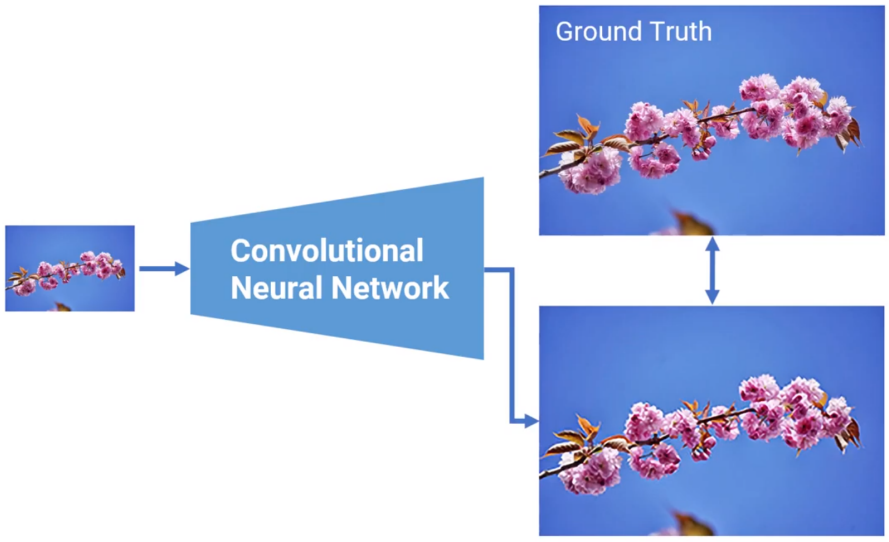


Figure 8.5: Super Resolution Convolutional Neural Network (SRCNN) training [DLH+15]

8.4.2 SRGAN

Ledig et al. (2017) [LTH+17] use GAN's framework (Fig. 8.6). The authors use VGG-based loss function

Figure 8.6: Super Resolution Generative Adversarial Network (SRGAN) training [LTH+17]

8.4.3 ESRGAN

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

- Remove Batch Normalization (BatchNorm)
- More layers and connections: residual scaling
- Modify the VGG loss: compare the feature maps before activation layers.
- Relativistic discriminator

8.4.4 Further Improvements

- Network interpolation [WYW+18; WYD+19]
- Contextual bilateral loss [ZCN+19]
- Real-ESRGAN [WXD+21]

8.4.5 Codes

- ESRGAN [WYW+18]
- Real-ESRGAN [WXD+21]

8.5 StyleGAN

8.5.1 Initial Work

- Watch Tero Karras FI Youtube video
- StyleGAN is image generation with style control. [KLA19]
- Key points on StyleGAN.v1's structure (Fig. 8.8a):
  - Add mapping and styles via mapping network f and AdaIN
  - Replace traditional input with a learnable constant $4 \times 4 \times 512$
  - Add noise inputs at different levels: coarse $(4^2 - 32^2)$ to fine layers $(64^2 - 1024^2)$ with different scaling B
  - Mixing regularization: use different style inputs from different latent vectors at different levels

8.5.2 Slider stuff

With some additional steps, one could create a slider to gradually change different attributes, e.g. gender, hair styles (Arxiv Insights).

8.5.3 Truncation Trick

Drawing latent vectors from a truncated sampling space tends to improve average image quality, with some loss in variation:

- Compute center of mass of $\mathcal{W}$: $\bar{\mathbf{w}} = \mathbb{E}_{\mathbf{z} \sim P(\mathbf{z})}[f(\mathbf{z})]$
- Given a vector $\mathbf{w}$, scale it: $\mathbf{w}' = \bar{\mathbf{w}} + \psi(\mathbf{w} - \bar{\mathbf{w}})$, where $|\psi| < 1$
- Varying ψ value also gives us the nice interpolation effect (Fig. 8.7)

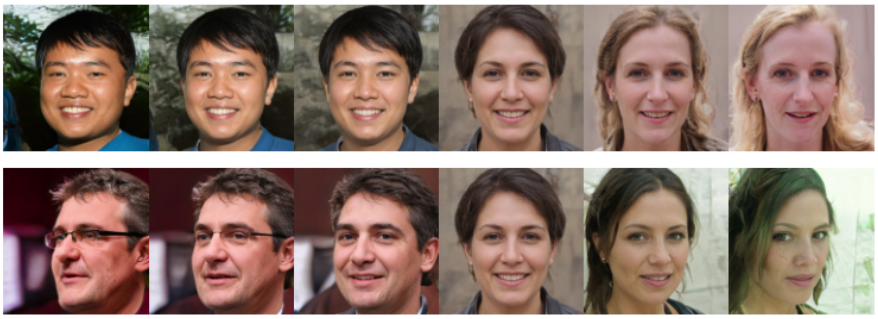


Figure 8.7: Effect of truncation trick

8.5.4 Improvements

Improvements to StyleGAN.v1 to handle droplet and phase artifacts:

- StyleGANv2 [KLA+20]
  - Replacing AdaIN with CNN weight demodulation and moving addition of noise outside of style area (Fig. 8.8b)

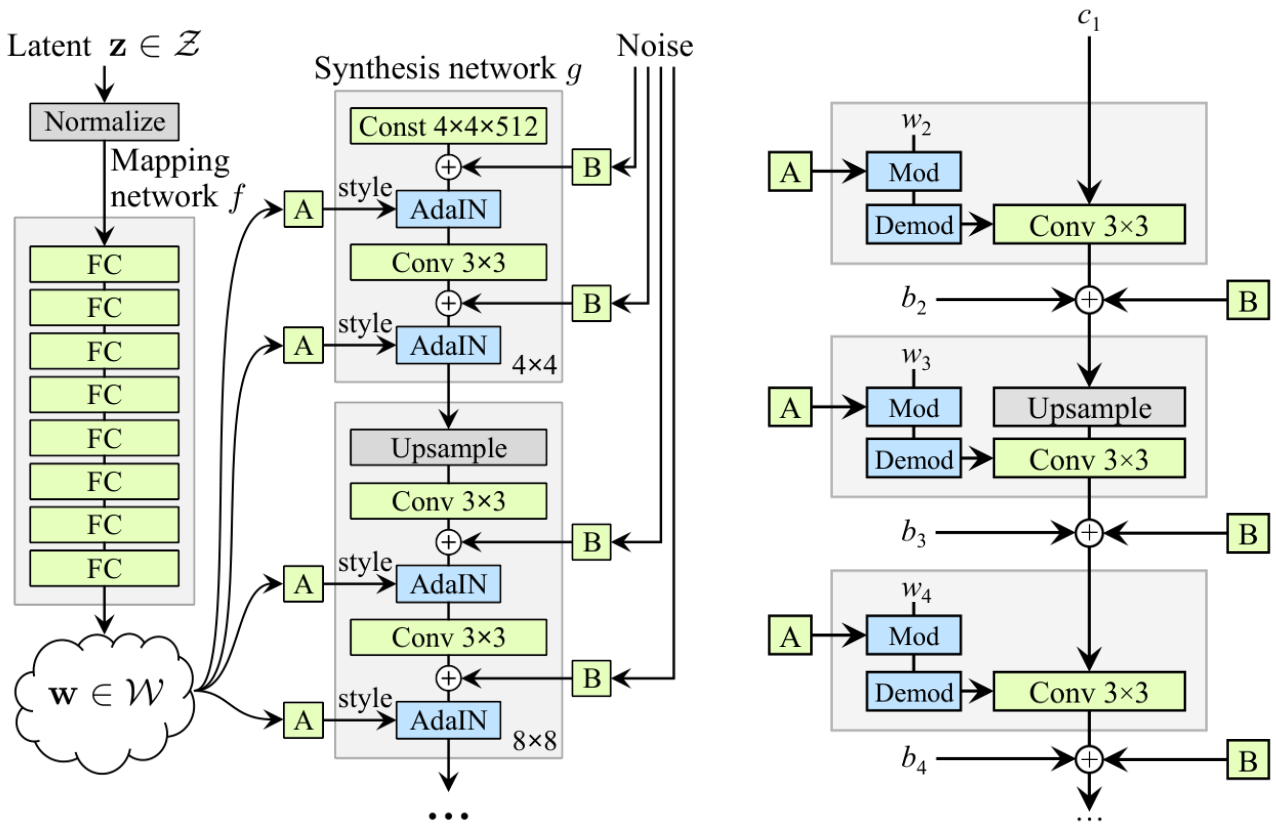
$$w'_{ijk} = s_i \cdot w_{ijk} \tag{8.32}$$

$$w''_{ijk} = \frac{w'_{ijk}}{\sqrt{\sum_{i,k} {w'_{ijk}}^2 + \epsilon}} \tag{8.33}$$

  - Lazy regularizer with PPL
  - Replacing progressive growing with skip connections and residual modules.
  - Finally, it trains faster
- StyleGAN3 suggests even more changes to deal with sticking texture artifacts (check bycloud and Andrew Melnik videos) [KAL+21]

8.6 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



(a) StyleGANv1's structure [KLA19]. **(b)** Weight demodulation [KLA+20].

Figure 8.8: StyleGAN architectures.

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.

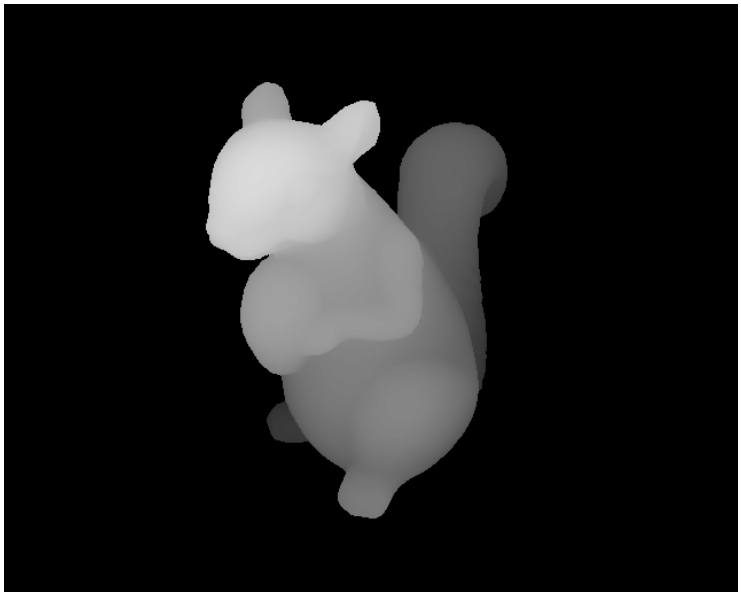


Figure 9.1: Example of a depth map [TSS+18].

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

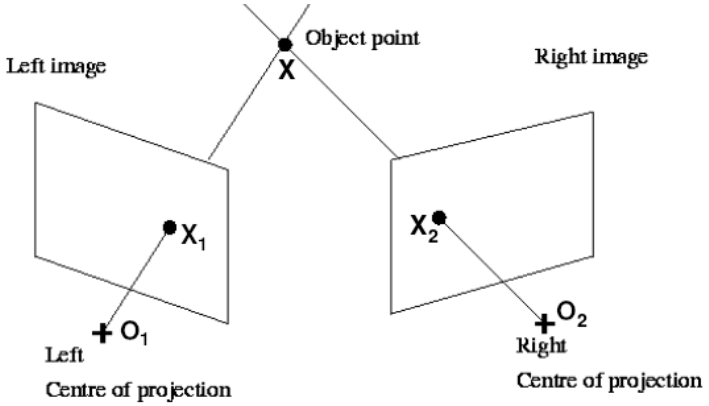


Figure 9.2: Example of triangulation (src). The lines connecting the camera poses with the correspondent points must intersect at the real object world space.

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
$\Rightarrow$ 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
• For **each pixel**, find correspondence • Easy when epipolar lines are scan lines (apply **rectification**)	• Only for a set of detected feature • Use feature description (Harris, SIFT??)
—— Pros ——	
• **Simple** process • **More depth** ⇒ useful for surface reconstruction	• **Efficiency** • Can have more reliable matches • Less sensitive to illumination ⇒ **robust**
—— Cons ——	
Problem with: • **texture-less regions** • different **viewpoints**	• Have to know enough to pick good features • **Sparse** information

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

16.2 Meta 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).

[HB17] X. Huang and S. Belongie. "Arbitrary style transfer in real-time with adaptive instance normalization". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 1501–1510.

[HRU+17] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. "GANs trained by a two time-scale update rule converge to a local Nash equilibrium". In: *Proc. of the Conf. on Neural Information Processing Systems (NeurIPS)* 30 (2017).

[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: *Proc. of the European Conference on Computer Vision (ECCV)*. Springer. 2016, pp. 694–711.

[KAL+21] T. Karras, M. Aittala, S. Laine, E. Härkönen, J. Hellsten, J. Lehtinen, and T. Aila. "Alias-free generative adversarial networks". In: *Proc. of the Conf. on Neural Information Processing Systems (NeurIPS)* 34 (2021), pp. 852–863.

[KLA19] T. Karras, S. Laine, and T. Aila. "A Style-Based Generator Architecture for Generative Adversarial Networks". In: *Proc. of the IEEE/CVF Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 4401–4410.

[KLA+20] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila. "Analyzing and Improving the Image Quality of StyleGAN". In: *Proc. of the IEEE/CVF Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 8110–8119.

[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.

[WXD+21] X. Wang, L. Xie, C. Dong, and Y. Shan. "Real-ESRGAN: Training Real-World Blind Super-Resolution with Pure Synthetic Data". In: *International Conference on Computer Vision Workshops (ICCVW)*. 2021.

[WYD+19] X. Wang, K. Yu, C. Dong, X. Tang, and C. C. Loy. "Deep network interpolation for continuous imagery effect transition". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 1692–1701.

[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.

[ZCN+19] X. Zhang, Q. Chen, R. Ng, and V. Koltun. "Zoom to learn, learn to zoom". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 3762–3770.

[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.