Deep Learning for Computational Photography

Paramanand Chandramouli
University of Siegen

Computational Photography

- Create new functionalities beyond conventional imaging

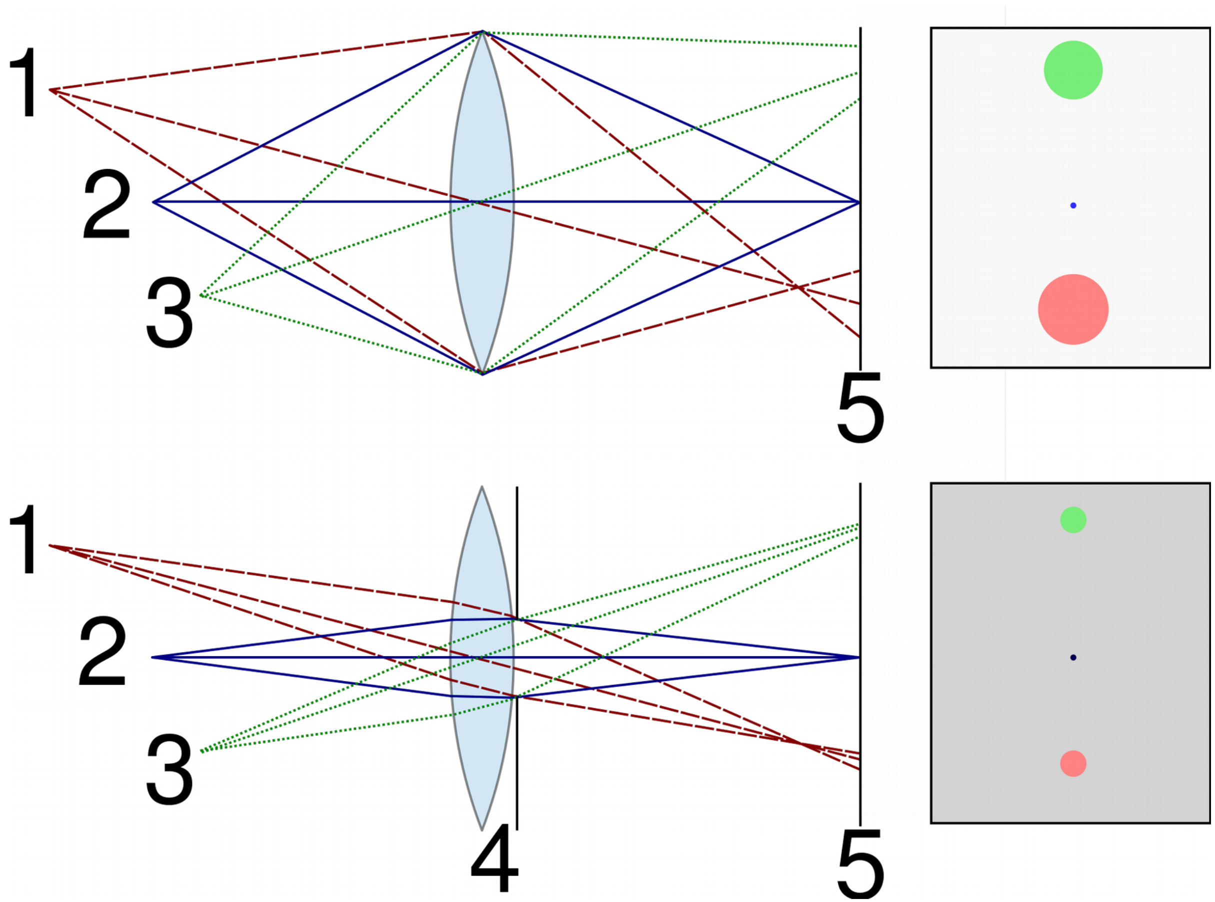
- Design modification

  Optics, Sensors, Shutter mechanism etc.

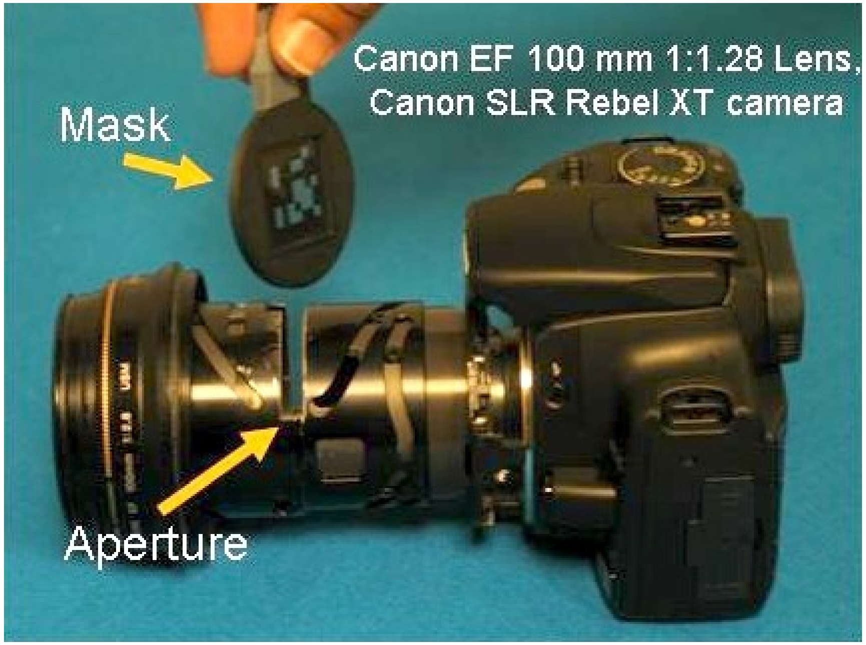
- Algorithms

  Inverse problems

Example- Extend the Depth of Field



Coded Aperture



Coded Aperture

Courtesy Ramesh Raskar MIT Media Labs

Coded Aperture

- Sharp images are rendered by solving inverse problems
- Would be much harder for a conventional aperture

Courtesy Ramesh Raskar MIT Media Labs

Transient Imaging

- Image the travel of light
- Use low-cost TOF sensor and algorithms

Heide et al. ``Low-budget Transient Imaging using Photonic Mixer Devices'' SIGGGRAPH 2013

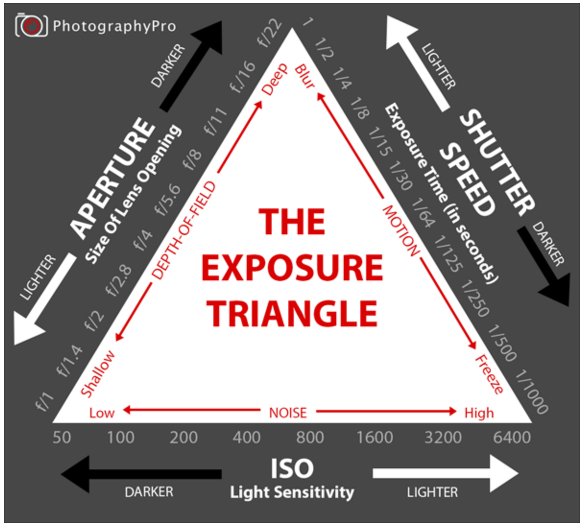
Image Restoration

Image Restoration

Xu and Jia ECCV 2010

Image Restoration

- Goal- Acquire a clean and sharp image
- Camera ISO gain denotes sensitivity of pixels- noise
- Exposure duration – Motion blur
- Aperture – Defocus



- Image Reconstruction

  Denoising, Deblurring

  Demosaicing, Super-resolution

  High Dynamic Range Imaging

Image Super Resolution

Image Superresolution using Perceptual Loss

Johnson et al. ``Perceptual Losses for Real-Time Style Transfer and Super-Resolution'' ECCV 2016

- A low-res image is input to a feed-forward network
- Output of the network is a High-res image

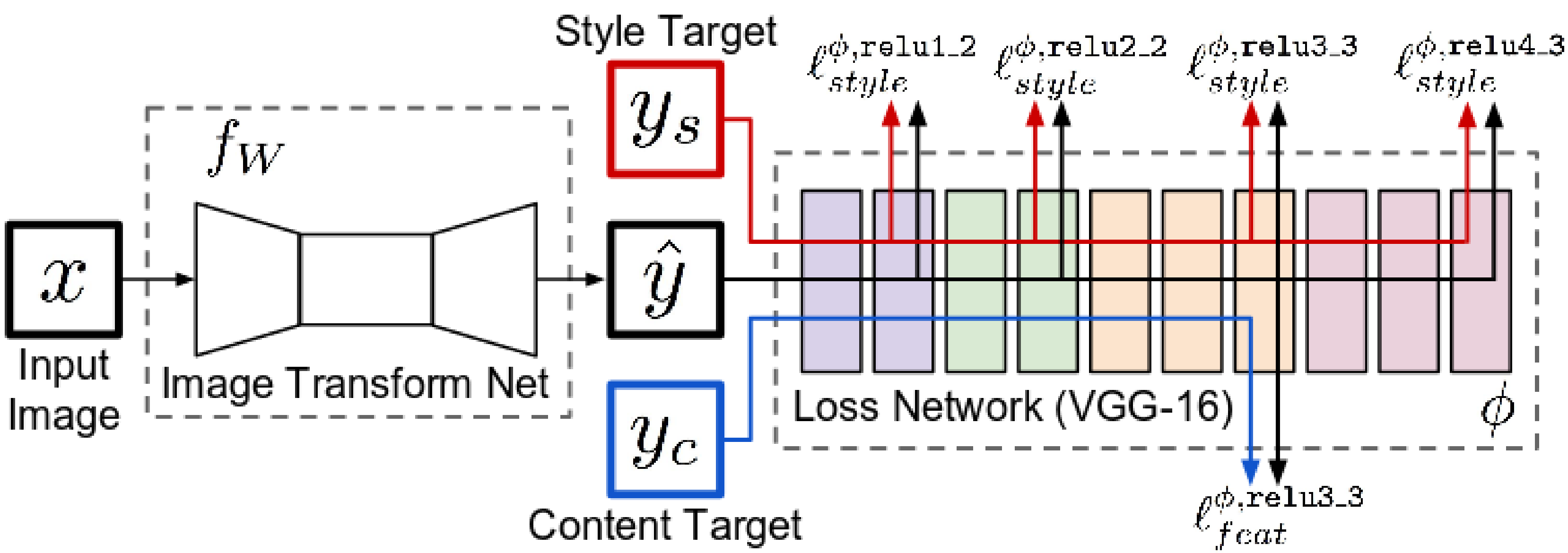
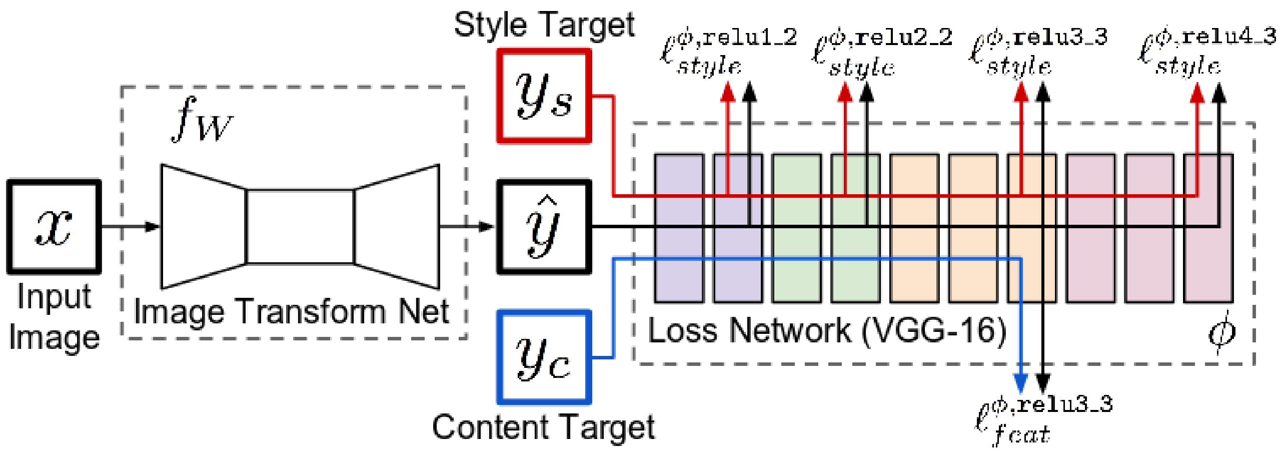


Image Superresolution using Perceptual Loss



$$W^* = \arg\min_W \mathbf{E}_{x,\{y_i\}} \left[\sum_{i=1} \lambda_i \ell_i(f_W(x), y_i) \right]$$

- Conventional approach: loss function is squared difference
- Perceptual loss is useful to generate better results

Perceptual Loss Functions

- Define loss function in terms of neural network
- VGG network which is trained for image classification
- Huge network with 138 million parameters
- Trained network is part of many Deep Learning softwares

VGGNet (2014)

Perceptual Loss Functions

- Feature reconstruction loss

$$\ell_{feat}^{\phi,j}(\hat{y}, y) = \frac{1}{C_j H_j W_j} \|\phi_j(\hat{y}) - \phi_j(y)\|_2^2$$

- Style Reconstruction loss (Deviation from our main topic!!)

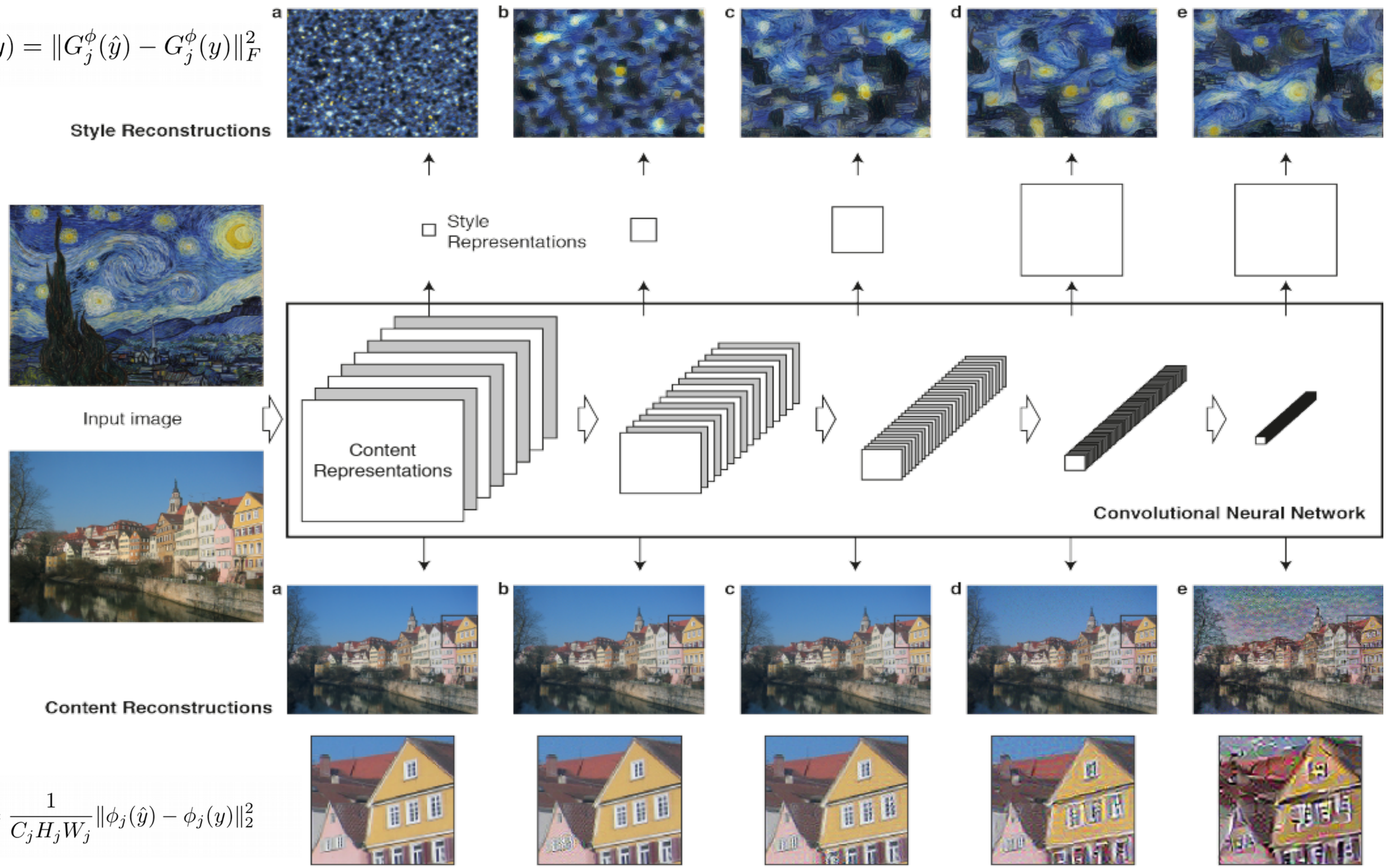
$$\ell_{style}^{\phi,j}(\hat{y}, y) = \|G_j^{\phi}(\hat{y}) - G_j^{\phi}(y)\|_F^2$$

- Gram matrix formed by covariance of features

$$G_j^{\phi}(x)_{c,c'} = \frac{1}{C_j H_j W_j} \sum_{h=1}^{H_j} \sum_{w=1}^{W_j} \phi_j(x)_{h,w,c} \phi_j(x)_{h,w,c'}$$

Perceptual Loss Functions

$$\ell_{style}^{\phi,j}(\hat{y}, y) = \|G_j^{\phi}(\hat{y}) - G_j^{\phi}(y)\|_F^2$$



$$\ell_{feat}^{\phi,j}(\hat{y}, y) = \frac{1}{C_j H_j W_j} \|\phi_j(\hat{y}) - \phi_j(y)\|_2^2$$

Gatys et al. 2015

Superresolution

- Trained with fixed upsampling
- Combined loss functions (TV + Feature loss)

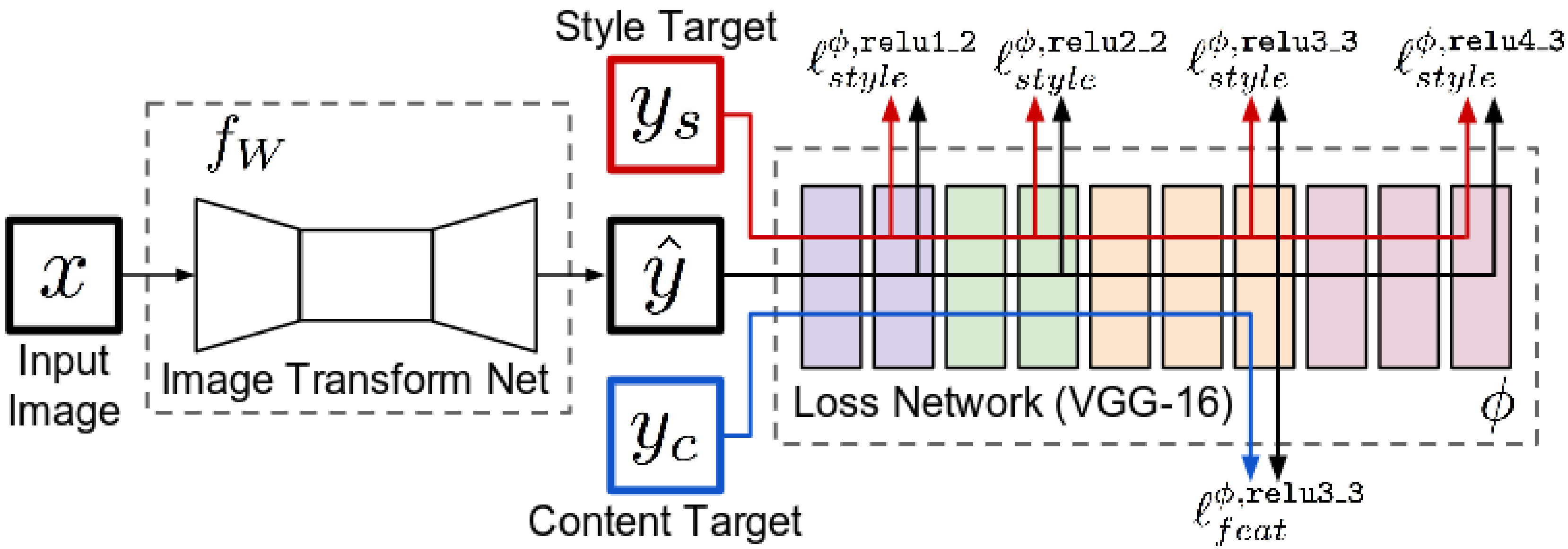
(Result from [Johnson et al. 2016])

Superresolution

Ground Truth | Bicubic | Ours (ℓ_{pixel}) | Ours (ℓ_{feat})

Style Transfer

- Trained on about 40k content images (fixed style image)
- Combined loss functions



Style Transfer

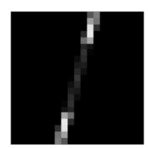
- Trained on about 40k content images (fixed style image)
- Combined loss functions

Motion Deblurring

Motion Deblurring- Blur Model

- Blur caused due to relative motion between camera and scene
- Uniform motion blur can be modeled by a convolution
- Point spread function (PSF) or blur kernel denotes the motion

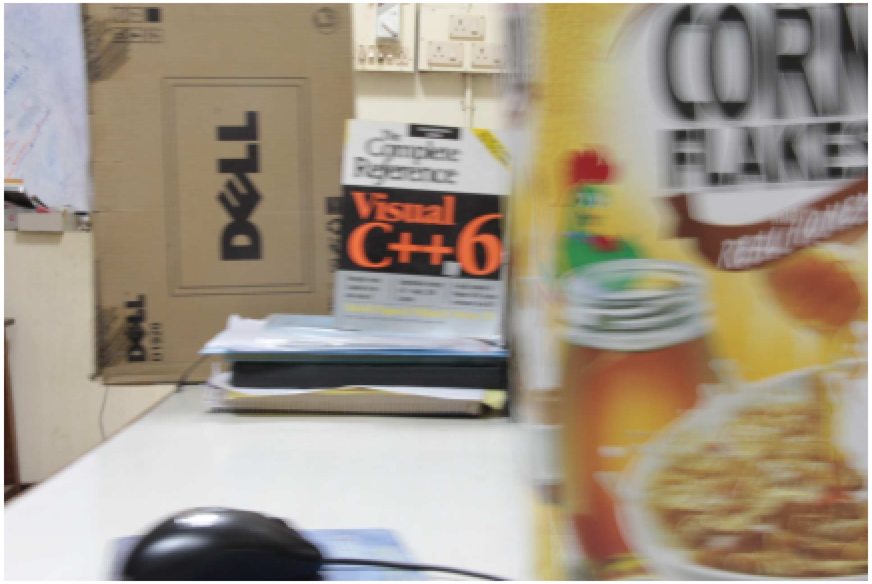
*



=

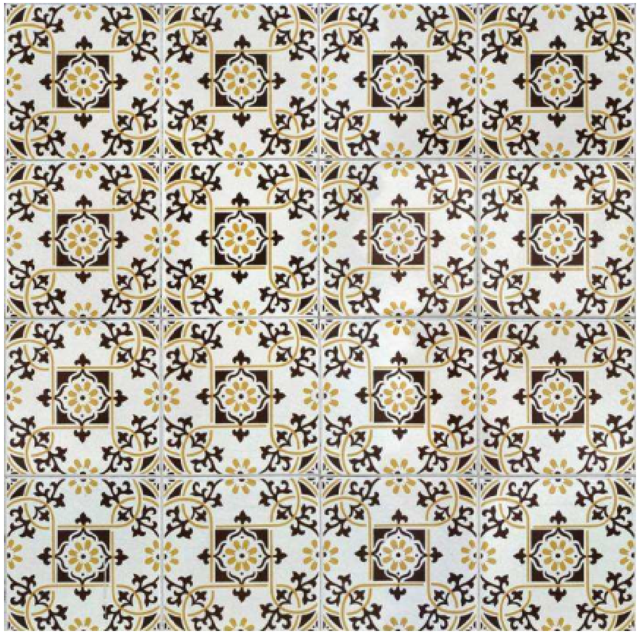
Space Variant Blur

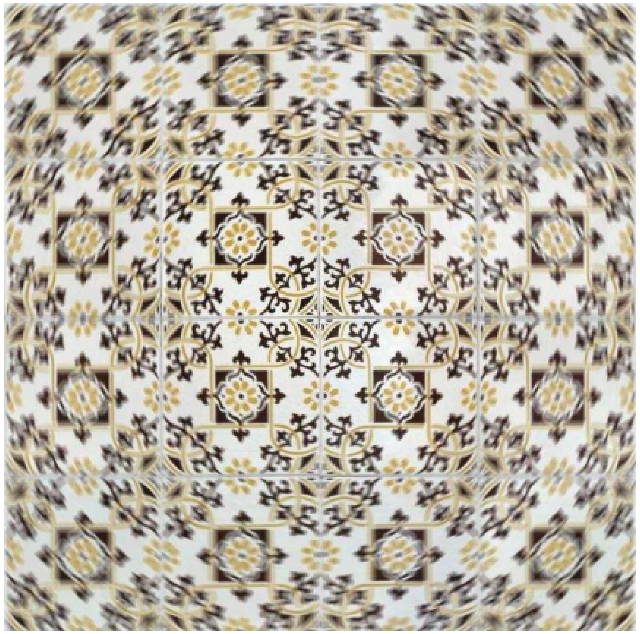
- Depth-dependent variation due to parallax
- PSF form remains same (camera translation)
- Difficult to model depth discontinuities



Space Variant Blur

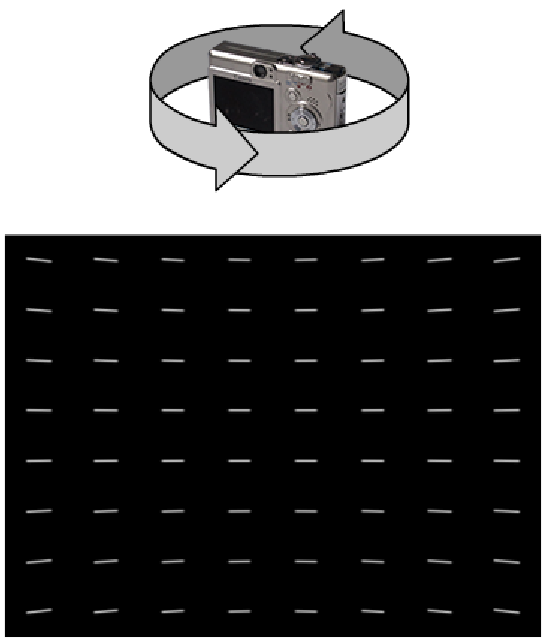
- Camera Rotation

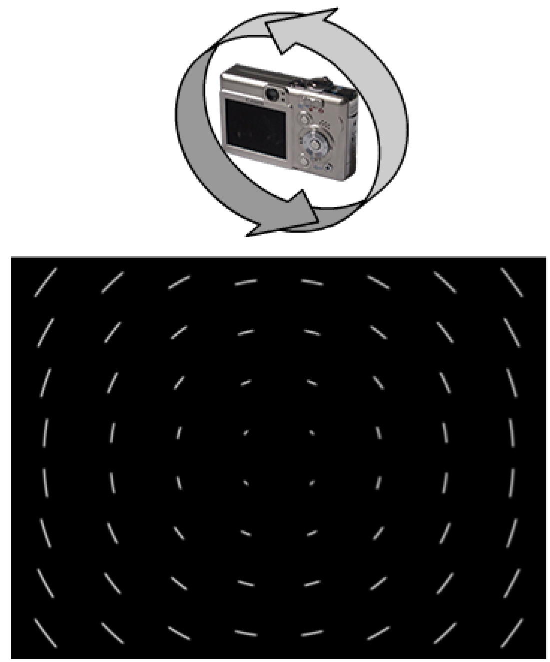




Space Variant Blur

- Apparent motion of scene points

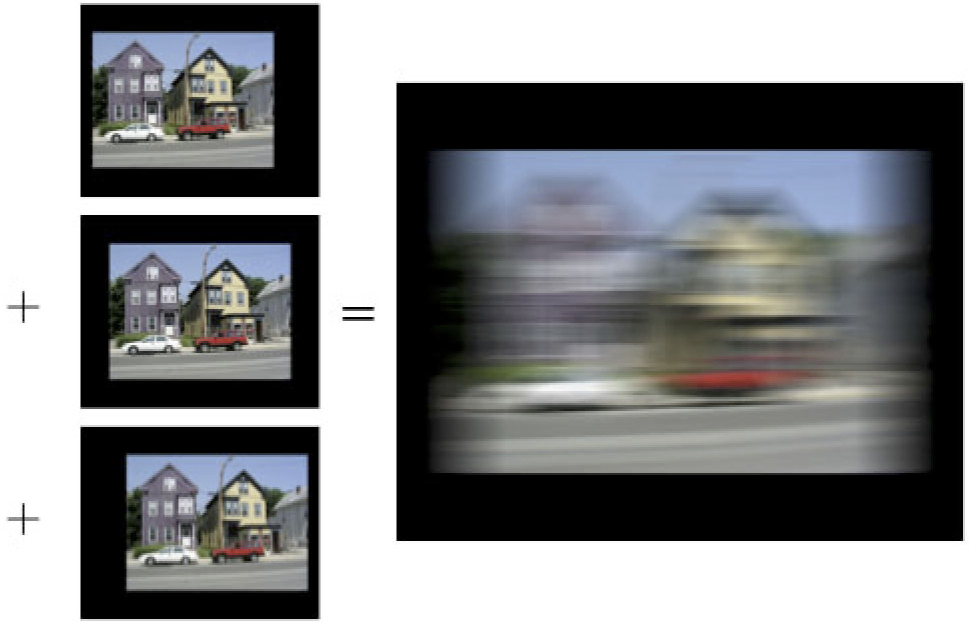




Whyte et al. ``Non-uniform Deblurring for Shaken Images'' CVPR 2010

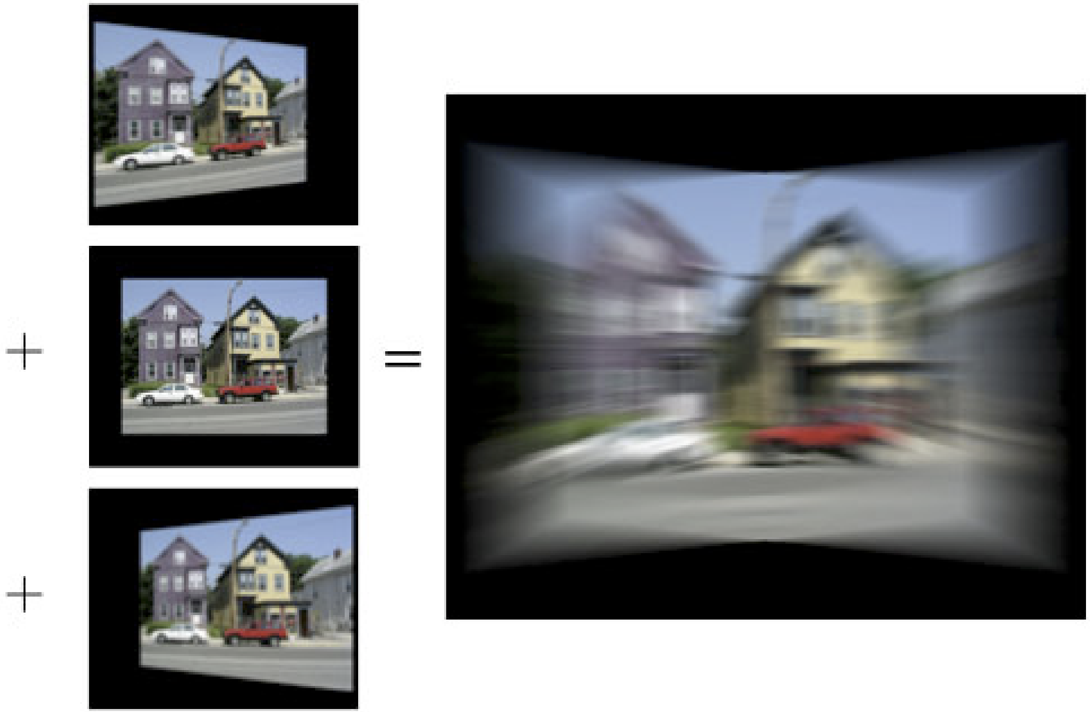
Space Variant Blur

- Camera Translation



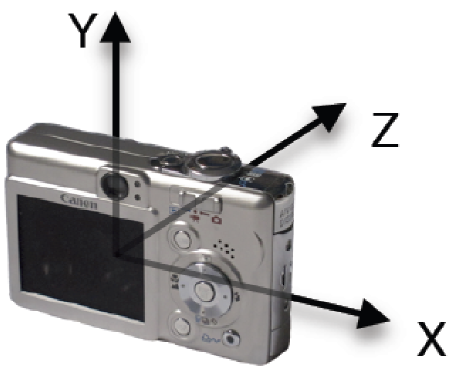
Space Variant Blur

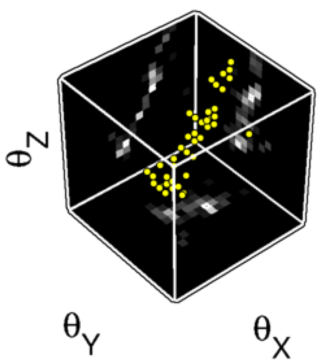
- Six DOF for camera motion – 6D blur
- Homography for a planar scene



Space Variant Blur

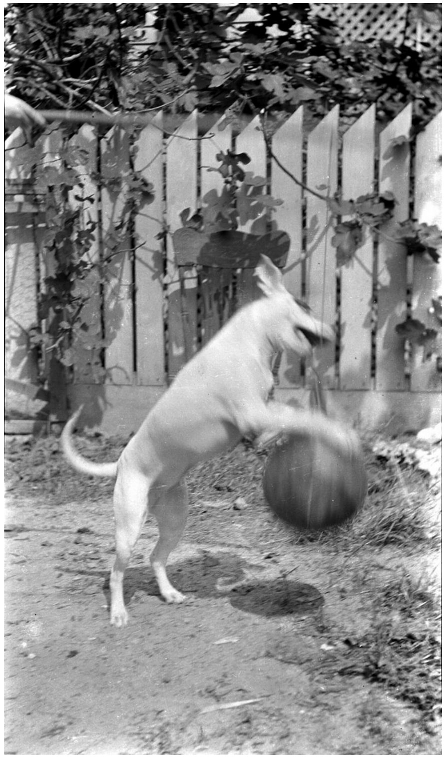
- Approximation are made in modeling 6DOF to 3DOF
- PSF denotes the time spent by the camera in a particular pose

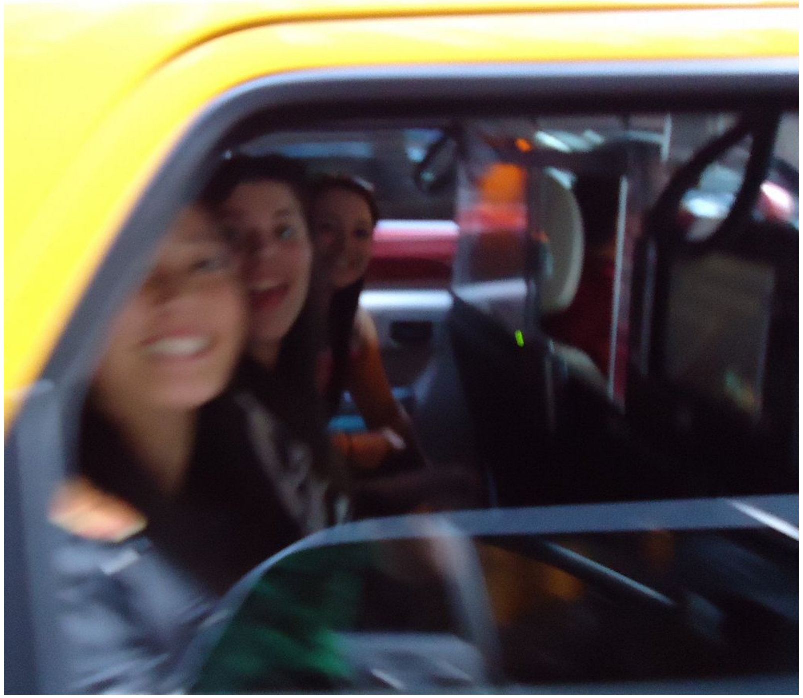




Object Motion

- Space-variant blur without any constraints





Blind Deconvolution

- Let us first consider constant blur scenario
- Blind deconvolution- both the image and PSF have to be estimated from a single blurred observation

$$f = k_0 * u_0 + n$$

- Image reconstruction problem but model is not known

$$\min_{u,k} \|k * u - f\|_2^2 + \lambda J(u) + \gamma G(k)$$

- Many solutions ranging from trivial solution to actual one
- If either one of the parameters is known then it would be a convex problem. Image estimation would still be ill-posed

Blind Deconvolution

- Common Tricks
  - Pyramid implementation
  - Work with gradient images
  - Solve for the PSF first
  - Bilateral filtering [Cho and Lee SIGGRAPH 2009]
- PAM algorithm [Perrone and Favaro CVPR 2014]

$$
\begin{aligned}
u^{t+1} &\leftarrow u^t - \epsilon_u \left(k^t_- \bullet (k^t \circ u^t - f) - \lambda \nabla \cdot \frac{\nabla u^t}{|\nabla u^t|} \right); \\
k^{t+1/3} &\leftarrow k^t - \epsilon_k \left(u^{t+1}_- \circ (k^t \circ u^{t+1} - f) \right); \\
k^{t+2/3} &\leftarrow \max\{k^{t+1/3}, 0\}; \\
k^{t+1} &\leftarrow \frac{k^{t+2/3}}{\|k^{t+2/3}\|_1};
\end{aligned}
$$

Blind Deconvolution

Blurry Input.

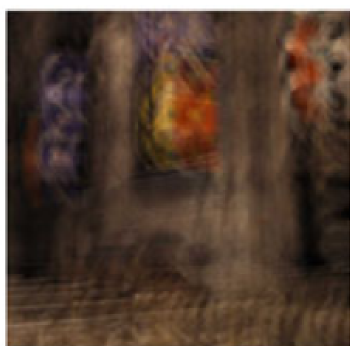
Restored image and blur with PAM algorithm.

Blind Deconvolution

- Camera rotations- some techniques use 3D PSF and similar optimization [Whyte et al. CVPR 2010]

Blurry Input
PSNR: 22.120.

Cho and Lee [4]
PSNR: 24.940.



Whyte *et al.* [40]
PSNR: 21.755.

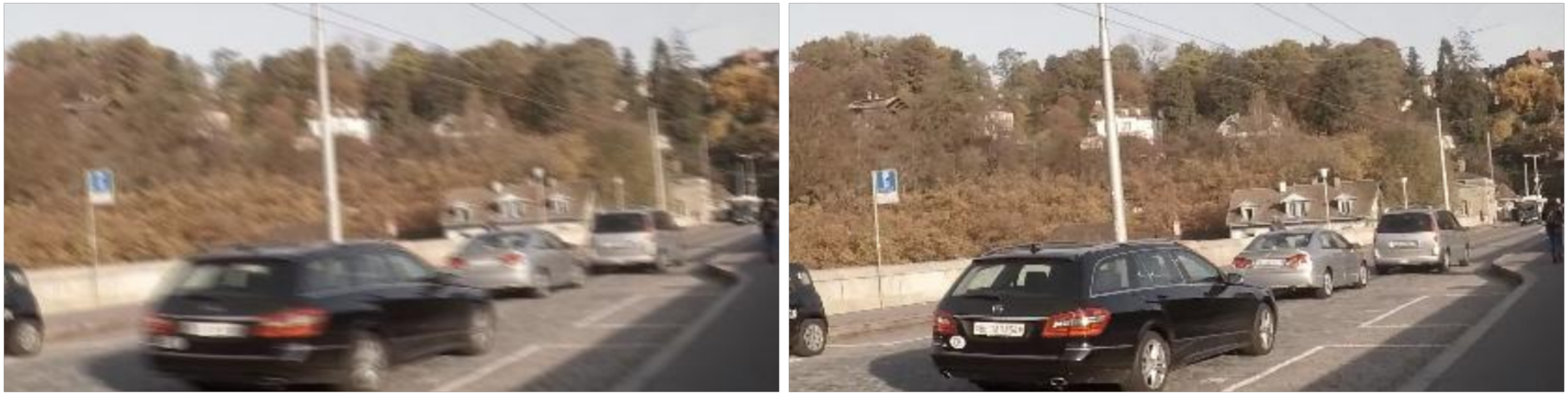
PAM
PSNR: 28.572.

Deep Learning for Blind Deconvolution

- [Schuler et al. ``Learning to Deblur'' PAMI 2015] Neural architecture designed to mimic the computational steps of BD iterations
- [Chakarabarti 2016] Estimate PSF and use conventional image deblurring
- Optimization-based methods have good performance

- Image deblurring (given PSF) – reconstruction
- Still ill-posed
- Unrolled optimization or Learning Proximal Operators

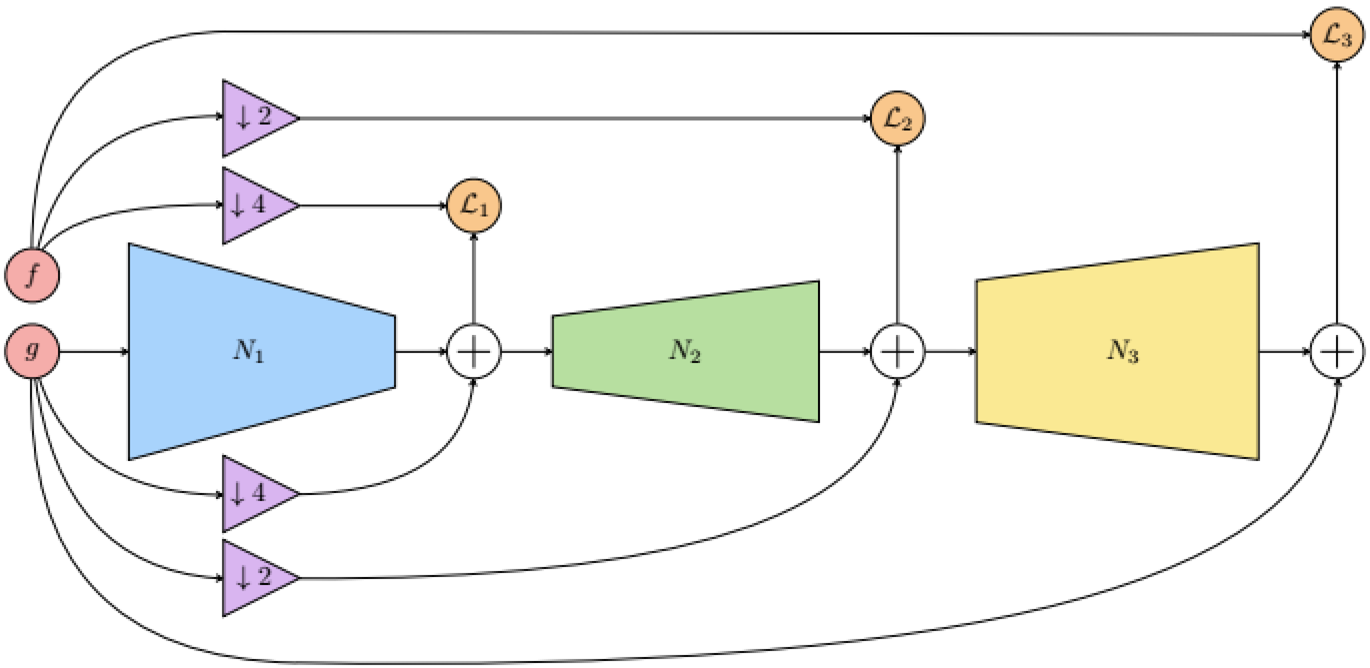
Deblurring Dynamic Scenes



Sample image pair from wild dataset

- Noroozi et al. ``Motion deblurring in the wild'' GCPR 2017
- Input is a blurry image and output is a sharp image
- No explicit PSF estimation
- Data itslelf represents occlusion, segmentation, camera shake and object motion
- To make data realistic, we estimate optical flow and consider sets where motion is not too high

Deblurring Dynamic Scenes



Network Architecture

- Multiscale residual architecture
- Total loss is sum of individual loss function

Deblurring Dynamic Scenes

Input

Method for uniform blur

DeblurNet

Deblurring Dynamic Scenes

- Convolutional model – input can be of any size
- Automatically keeps sharp regions sharp

Input

Methods for dynamic scenes

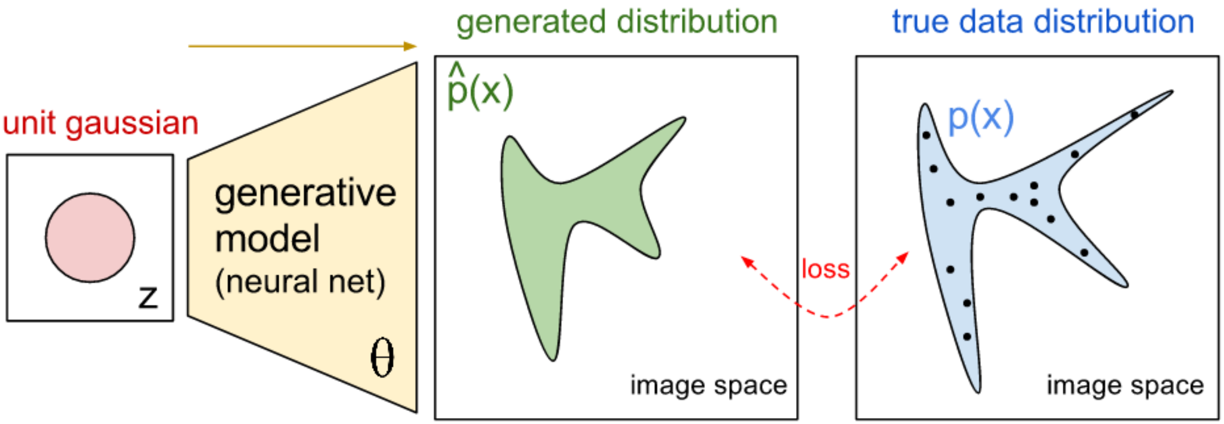
DeblurNet

DeblurGAN

Input DeblurGAN Ground Truth

DeblurGAN

- Deep Generative models – brief summary

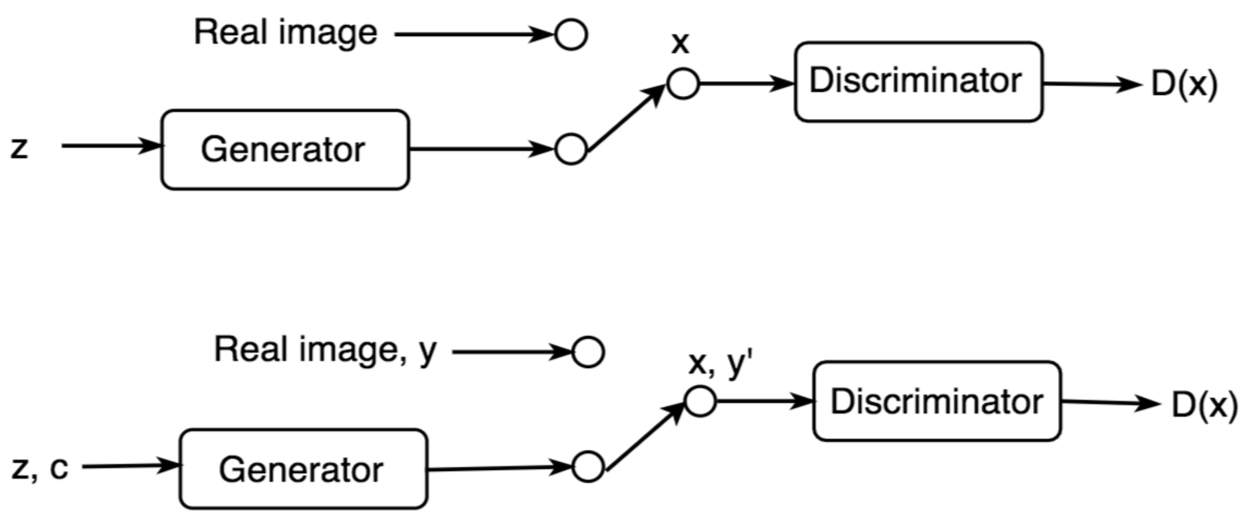


Courtesy: OpenAI Blog

- Goal: Enable computers to understand real world data !!
- A class of images are samples from a particular distribution
- Generative model: train to generate samples that are similar to the images in the dataset

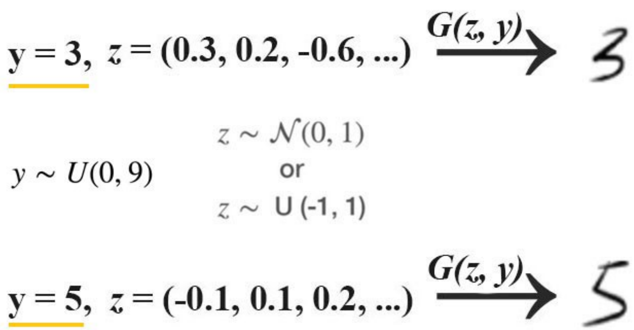
GAN and CGAN brief summary

- DeblurGAN is based on conditional GAN
- Generative adverserial network (GAN) a powerful generative model [Goodfellow et al. NIPS 2014]
- Has a generator and discriminator
- Conditional GAN takes additionally a label as input

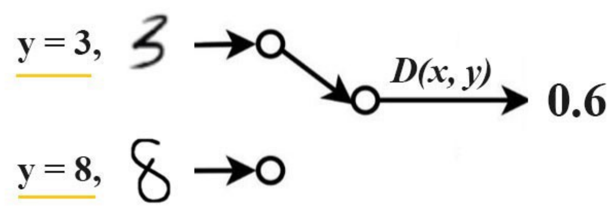


GAN and CGAN brief summary

- CGAN example: Hand-written digit generator
- Label is input to both Generator and Discriminator



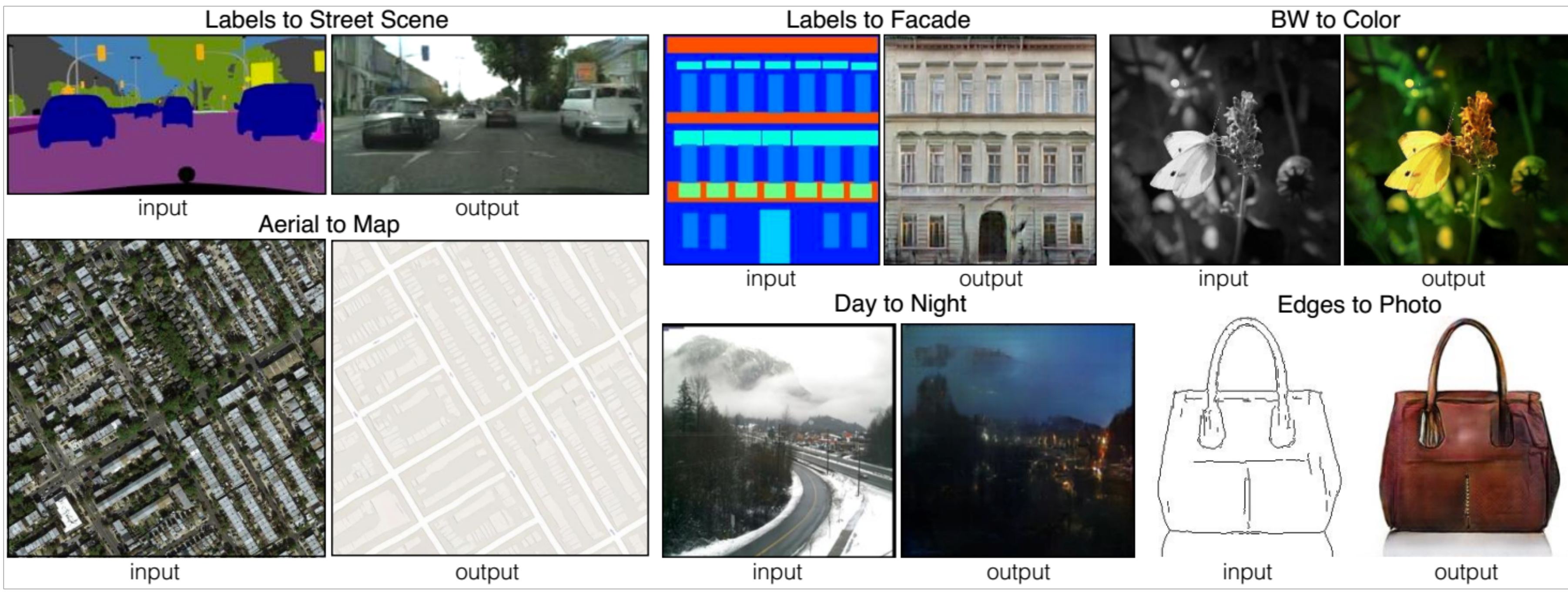
Generator



Discriminator

CGAN application

- Pix2Pix- [Isola et al. CVPR 2017]
- The latent variable has not much significance (in this case)



DeblurGAN

- DeblurGAN is CGAN trained on motion blurred/clean images
- Used real and synthetic data.
- Still not able to handle large blurs

Reconstruction using a Generative Prior

- Recent advances- very high quality generative models

  See StyleGAN [Karras et al. CVPR 2019]

- Caution! Hard to train
- Can generative models be used as prior to solve linear inverse problems?

$$y = Ax^* + \eta,$$

- Consider that the solution lies in the range of a generator
- Under certain conditions, one can solve for a latent variable [Bora et al. ICML 2018]

$$loss(z) = \|AG(z) - y\|^2$$

- Although the method was developed for compressed sensing, other tasks were also addressed

Thank You