Computer Vision Revision

Tianxiang and Fiseha

Exam Outline

- The Final exam is 60% of your total grade
  - Q1: Feature Detection and Stitching(20%)
  - Q2: Multiple Views and Motion (20%)
  - Q3: Visual Recognition (20%)

Q1: Feature Detection and Stitching (20%)

- Common features used in CV:
  - Colour features
  - Texture features
  - Shape features
  - Edge features
- Some common feature vectors
  - Colour histograms
  - Local binary patterns
  - Histograms of Gradient Orientations (HoG)

Image Feature Descriptor

- Common features used in CV:
  - Colour features
  - Texture features
  - Shape features
  - Edge features
- Some common feature vectors
  - Colour histograms
  - Local binary patterns
  - Histograms of Gradient Orientations (HoG)

Colour Features

- Colour correlates well with class identity.

- Human vision works hard to preserve colour constancy: presumably because colour is useful.

- Histograms
  - Are invariant to translation and rotation.
  - Change slowly as viewing direction changes.
  - Change slowly with object size.
  - Change slowly with occlusion.

- Colour histograms summarise target objects quite well, and should match a good range of images.

5

Colour Histograms

- Histogram
  - X-axis: bins of intensity (colour) value intervals
  - Y-axis: number of pixels whose value falls into those bins.
- Which colour space? – depend on colour models
  - RGB: red, green, blue channels.
  - YUV: Y (luma), U (chrominance), V (chrominance) channels.
- How many bins? 256 (0 – 255) or 32 (0-7, 8-15, …)

6

Texture Features

- Colour is a property of a single pixel, texture features capture the frequency with which patterns of colour/grey level appear.
- E.g. Local Binary Patterns (LBP)
  - For each pixel *p*, create an 8-bit number $b_1\, b_2\, b_3\, b_4\, b_5\, b_6\, b_7\, b_8$, where $b_i=0$ if neighbor *i* has value less than or equal to *p*'s value and *1* otherwise.

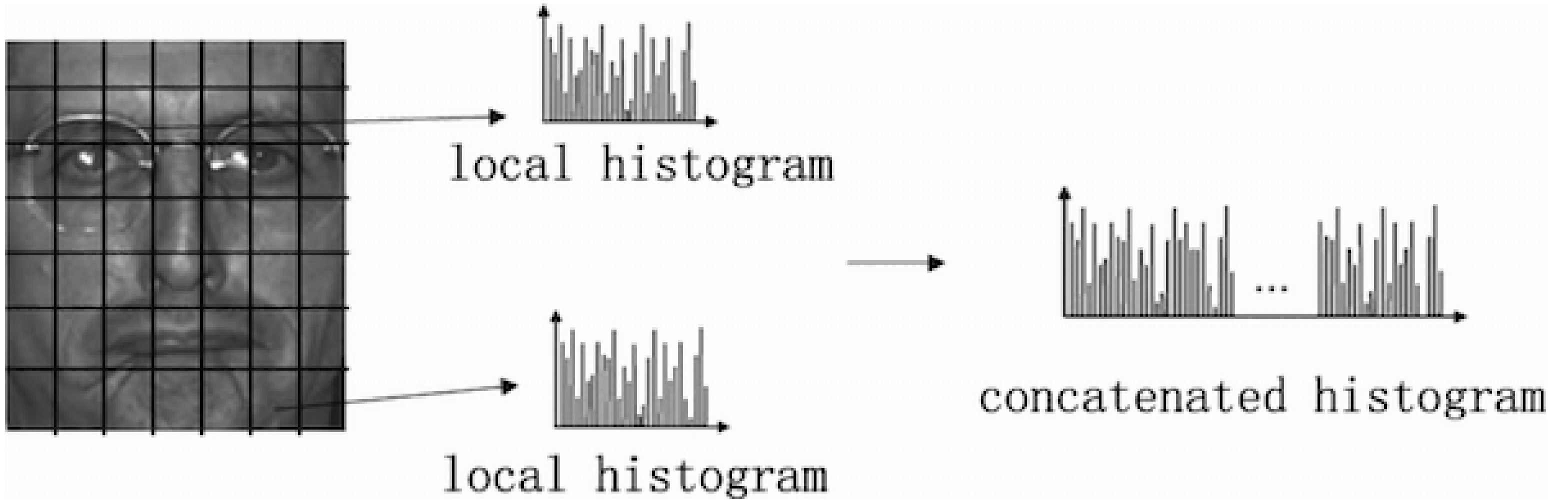
	1	2	3	
	100	101	103	
8	40	50	80	4
	50	60	90	5
	7	6		

$\longrightarrow$ $\boxed{1\ 1\ 1\ 1\ 1\ 1\ 0\ 0}$

Captures the pattern around one pixel

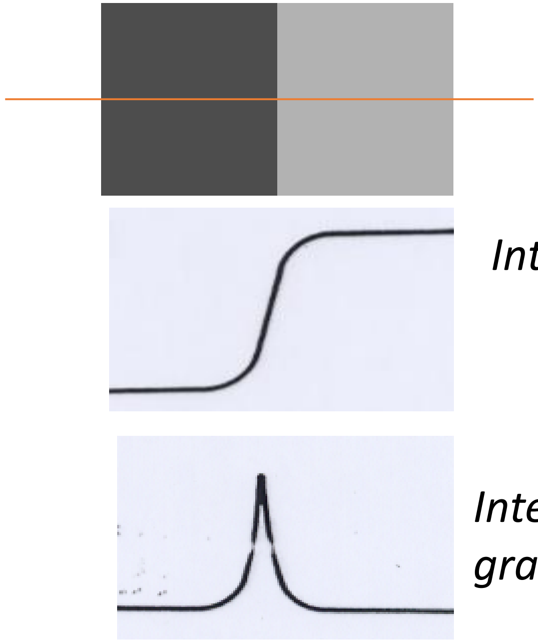
LBP Feature Vector

- Divide the patch into cells e.g. 16 x 16 pixels per cell.
- Compute the local patch description number of each pixel.
  - As described in previous slide.
- Histogram these numbers over each cell.
  - Usually a 256-d feature vector.
- Optionally normalize each histogram (so its bins sum to 1).
- Concatenate (normalized) histograms to make the feature vector.



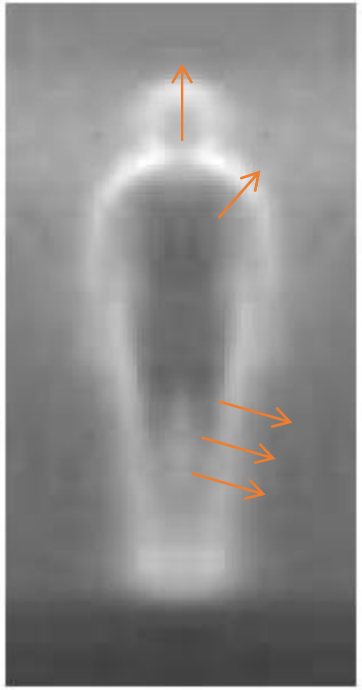
Shape Features

- Focus on **image gradient** measures:
  - The gradient of an image measures how it is changing.
  - The boundaries of objects are often associated with large gradients.
  - Distributions of gradients and gradient orientations reflect boundary shape (and internal boundaries between parts, surfaces, etc.).



Intensity

Intensity gradient



Mean gradient of a large set of person images

9

Derivative Filters

- Sobel Operators

G_x

-1	0	1
-2	0	2
-1	0	1

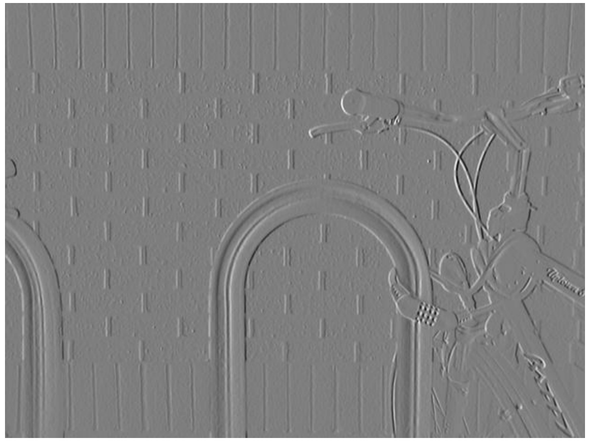
G_y

-1	-2	-1
0	0	0
1	2	1

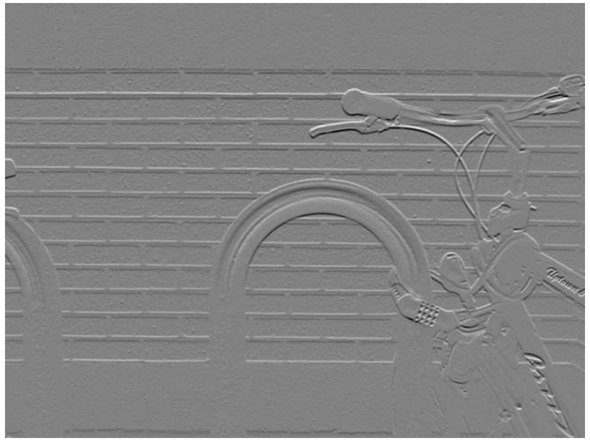
- Applied separately and results combined to estimate overall gradient magnitude.

10

Derivative Filters



G_x



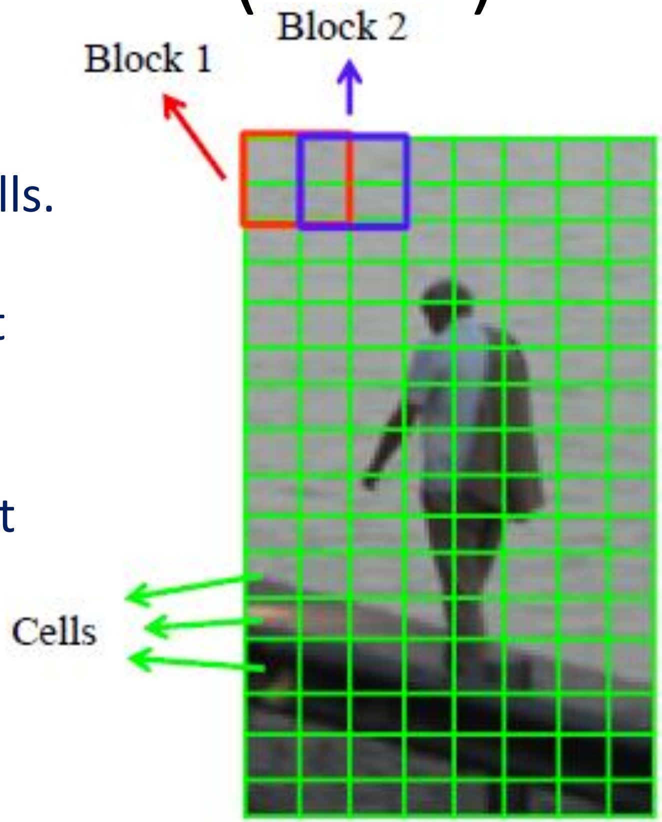
G_y

Oriented derivative filters only respond to edges in one direction.

Histogram of Oriented Gradients (HoG)

- Basic idea:
  - Local shape information often well described by the distribution of intensity gradients or edge directions
  - Convert the image (width*height*channels) into a feature vector, then apply the classification algorithms
  - The intent is to generalize the object in such a way that the same object (e.g., person) produces as close as possible to the same feature descriptor when viewed under different conditions

Histogram of Oriented Gradients (HoG)



- Divide the patch into small **cells**.
- Define slightly larger **blocks**, covering several cells.
- Compute gradient magnitude and orientation at each **pixel**.
- Compute a local **weighted histogram** of gradient orientations for each cell, weighting by some function of magnitude.
- Concatenate histogram entries to form a HoG vector for each block.
- Normalize vector values by dividing by some function of vector length.
  - For improved accuracy, the local histograms can be contrast-normalized by calculating a measure of the intensity across a larger region of the image, called a block, and then using this value to normalize all cells within the block.

13

Scale Invariant Feature Transform

- The method
  - Scale-space extrema detection (for **scale invariance**)
  - Keypoint localization (for **translation invariance**)
  - Orientation assignment (for **rotation/orientation invariance**)
  - Keypoint descriptor (for **illumination invariance**)

SIFT Overview

- Scale-space extrema detection
  - Search over all scales and image locations
  - Detect points that are invariant to scale and orientation
- Keypoint localization
  - A model is fit to determine the location and scale. Keypoints are selected based on measures of their stability
- Orientation assignment
  - Compute best orientation for each keypoint region
- Keypoint descriptor
  - Use local image gradients at selected scale and rotation to describe each keypoint region

15

SIFT: Invariance Properties

- To be robust to intensity value changes
  - Use gradient orientations
- To be scale invariant
  - Estimate the scale using scale space extrema detection
  - Calculate the gradient after Gaussian smoothing with this scale
- To be orientation invariant
  - Rotate the gradient orientations using the dominant orientation in a neighborhood
- To be Illumination invariant
  - Working in gradient space, so robust to $I = I + b$
  - Normalize vector to [0…1], robust to $I = \alpha I$ brightness changes
  - Clamp all vector values > 0.2 to 0.2, robust to “non-linear illumination effects”

Image Stitching

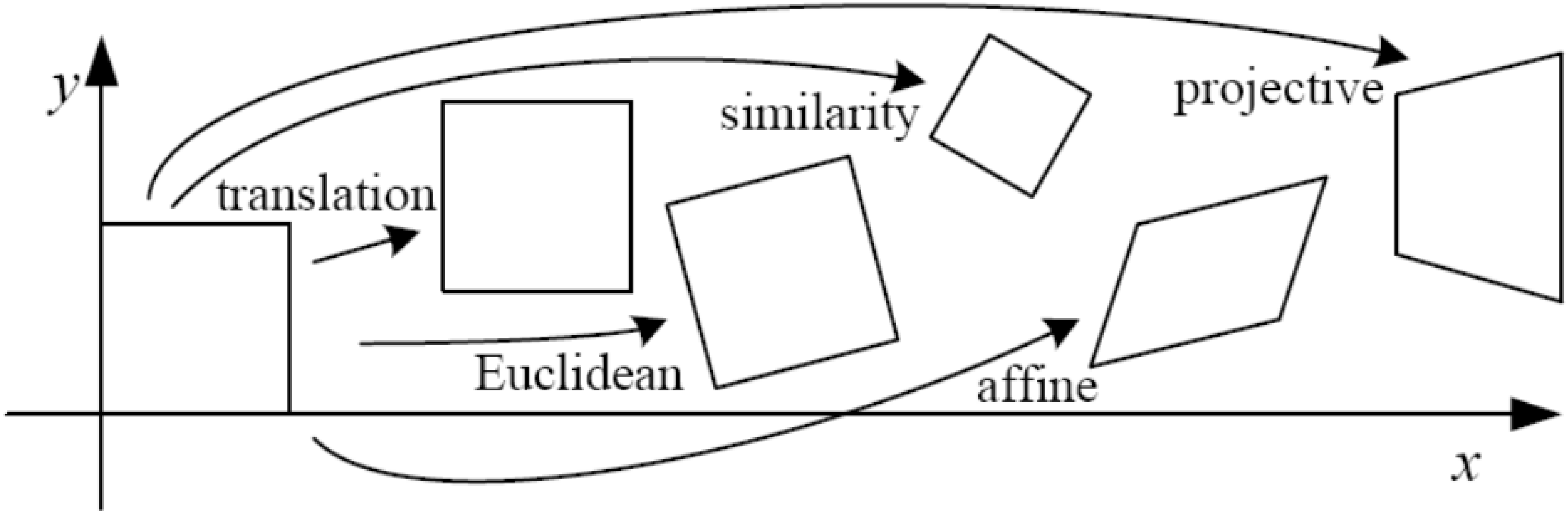
Combine two or more overlapping images to make one larger image

Image Stitching: The Idea

1. Take a sequence of images from the same position.
2. To stitch two images: compute transformation between second image and first.
3. Shift (warp) the second image to overlap with the first.
4. Blend the two together to create a mosaic.
5. If there are more images, repeat step 2 to 4.

18

Classification of 2D transformations



19

Finding the Transformation

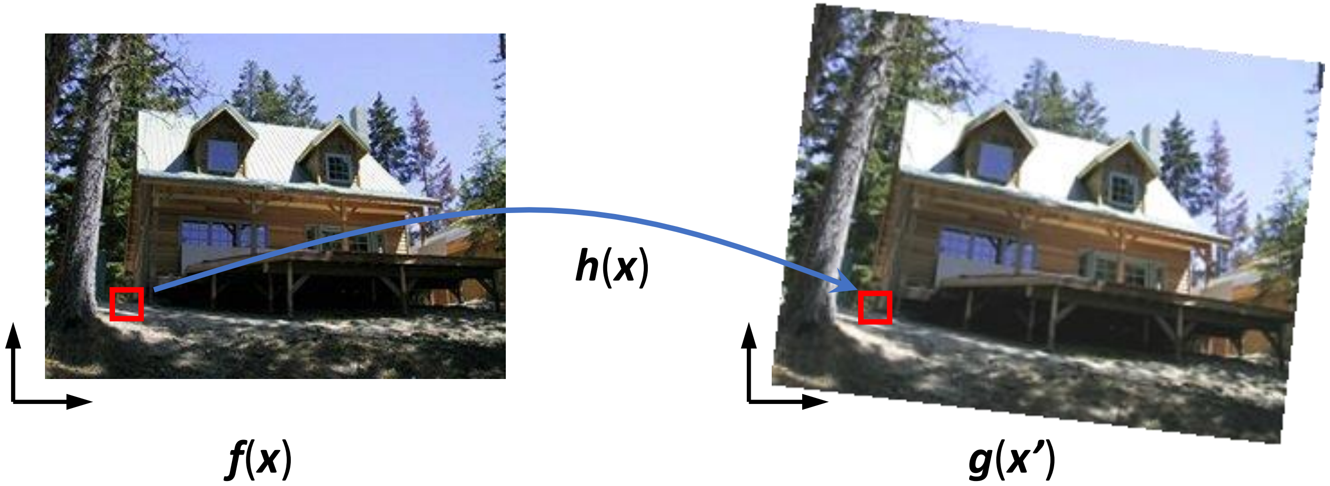
- Translation = 2 degrees of freedom
- Similarity = 4 degrees of freedom
- Affine = 6 degrees of freedom
- Projective (Homography) = 8 degrees of freedom
- "Least squares" solution.

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

20

Image Warping

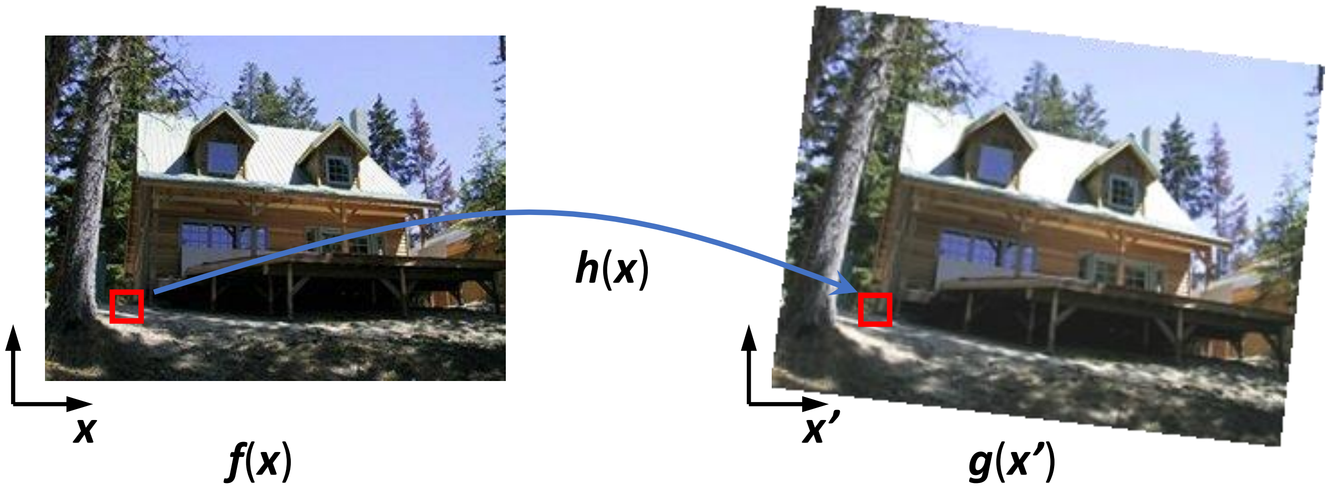
- Move pixels of an image
- Given a coordinate transform ***x'*** = ***h***(***x***) and a source image ***f***(***x***),
- We compute a transformed image ***g***(***x'***) = ***f***(***h***(***x***)).
  - Change the domain of image function



21

Forward Warping

- Send each pixel $\boldsymbol{f}(\boldsymbol{x})$ --the RGB value-- to its corresponding location in the dest image: $\boldsymbol{x'} = \boldsymbol{h}(\boldsymbol{x})$ in $\boldsymbol{g}(\boldsymbol{x'})$.



22

Inverse Warping

- Get each pixel $\boldsymbol{g}(\boldsymbol{x'})$ --the RGB value-- from its corresponding location in the source image: $\boldsymbol{x} = \boldsymbol{h}^{-1}(\boldsymbol{x'})$ in $\boldsymbol{f}(\boldsymbol{x})$.

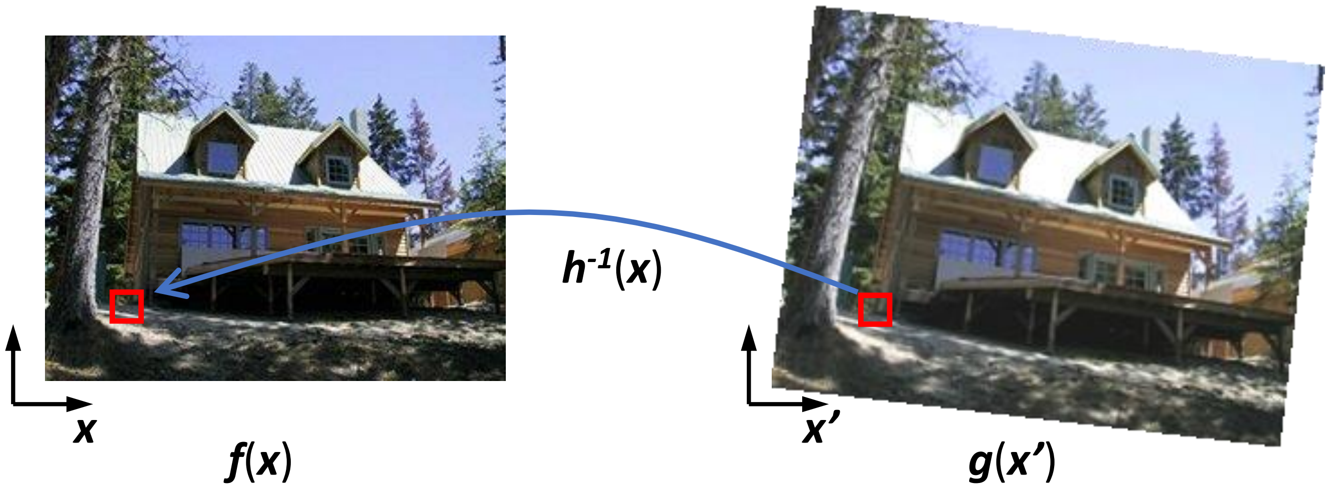
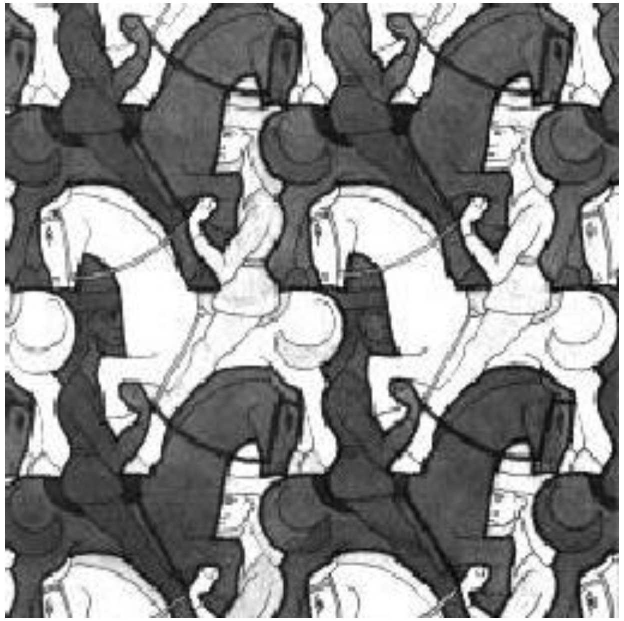
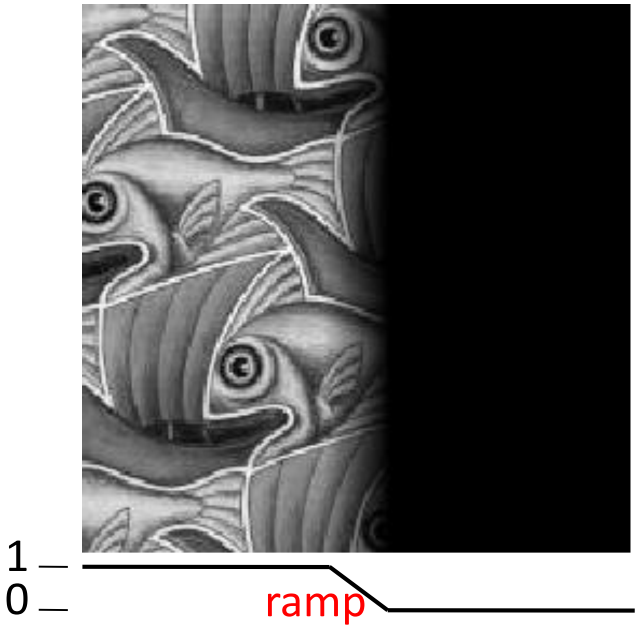


Image Blending

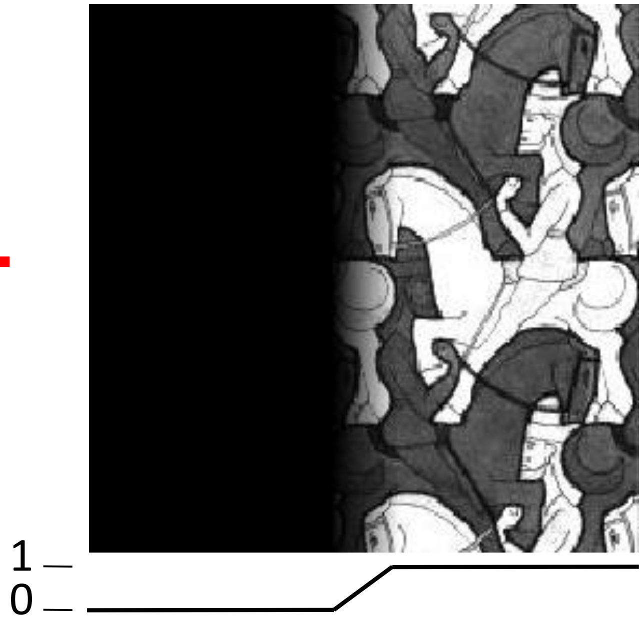


- Feathering with ramp, alpha blending.
- Pyramid blending.
- Multiband blending.

Feathering



+



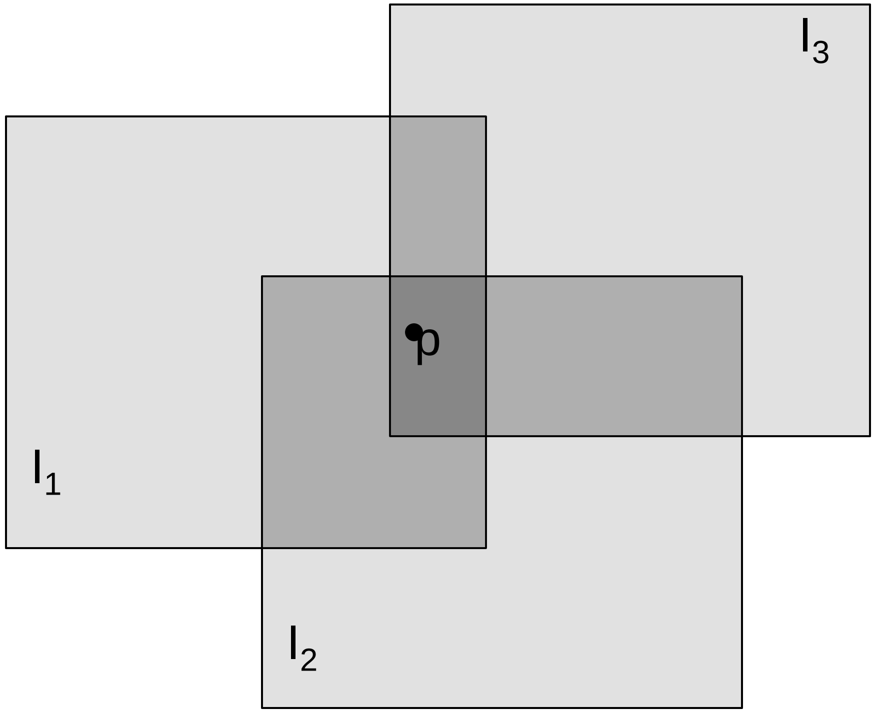
=

$$I_{blend} = c * I_{left} + (1 - c) * I_{right}$$

c can be 1 (object 1), and 0 (object 2), or somewhere in between

25

Alpha Blending



Encoding blend weights: $I(x,y) = (\alpha_1 R, \alpha_2 G, \alpha_3 B, \boldsymbol{\alpha})$

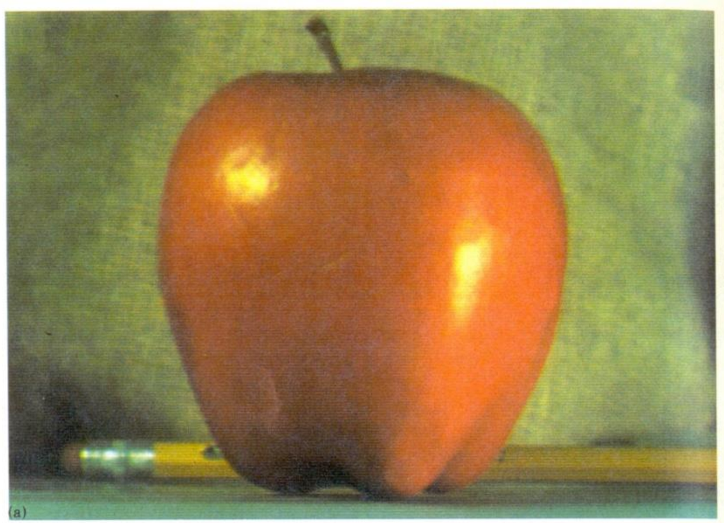
color at p = $$\frac{(\alpha_1 R_1,\ \alpha_1 G_1,\ \alpha_1 B_1) + (\alpha_2 R_2,\ \alpha_2 G_2,\ \alpha_2 B_2) + (\alpha_3 R_3,\ \alpha_3 G_3,\ \alpha_3 B_3)}{\alpha_1 + \alpha_2 + \alpha_3}$$

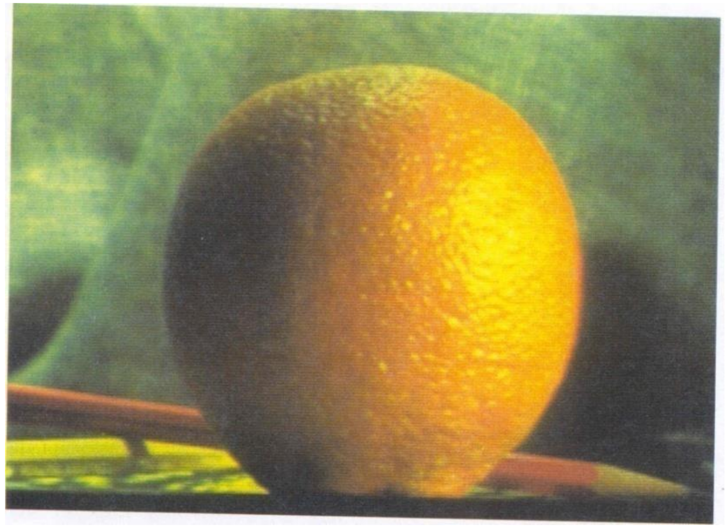
Implement this in two steps:

1. accumulate: add up the (α premultiplied) RGB values at each pixel.
2. normalize: divide each pixel's accumulated RGB by its α value.

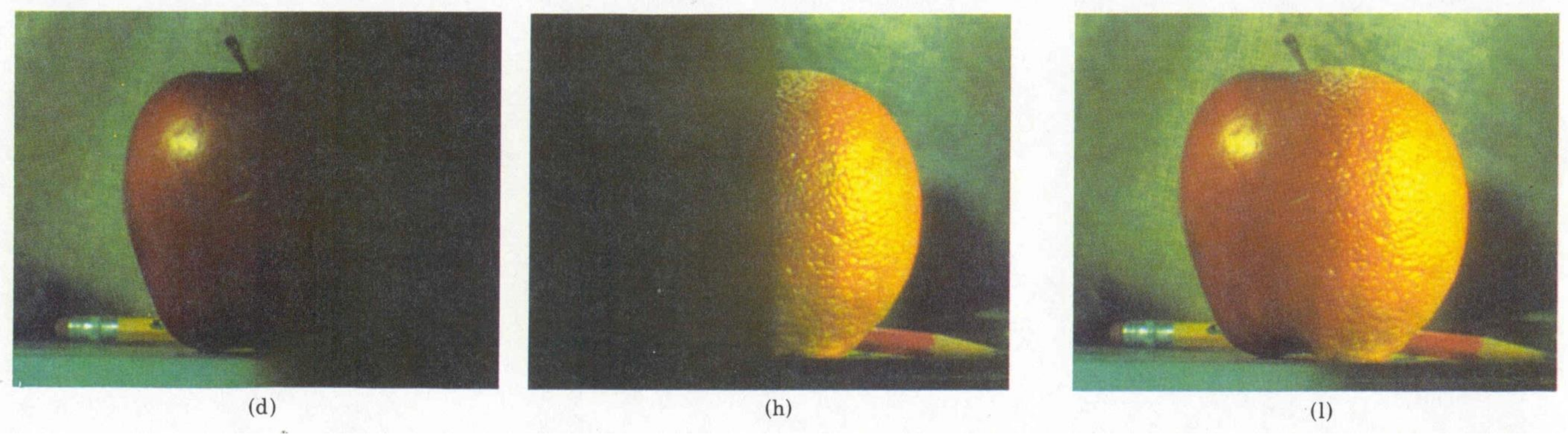
Pyramid Blending

apple





orange



Create a Laplacian pyramid, blend each level.

27

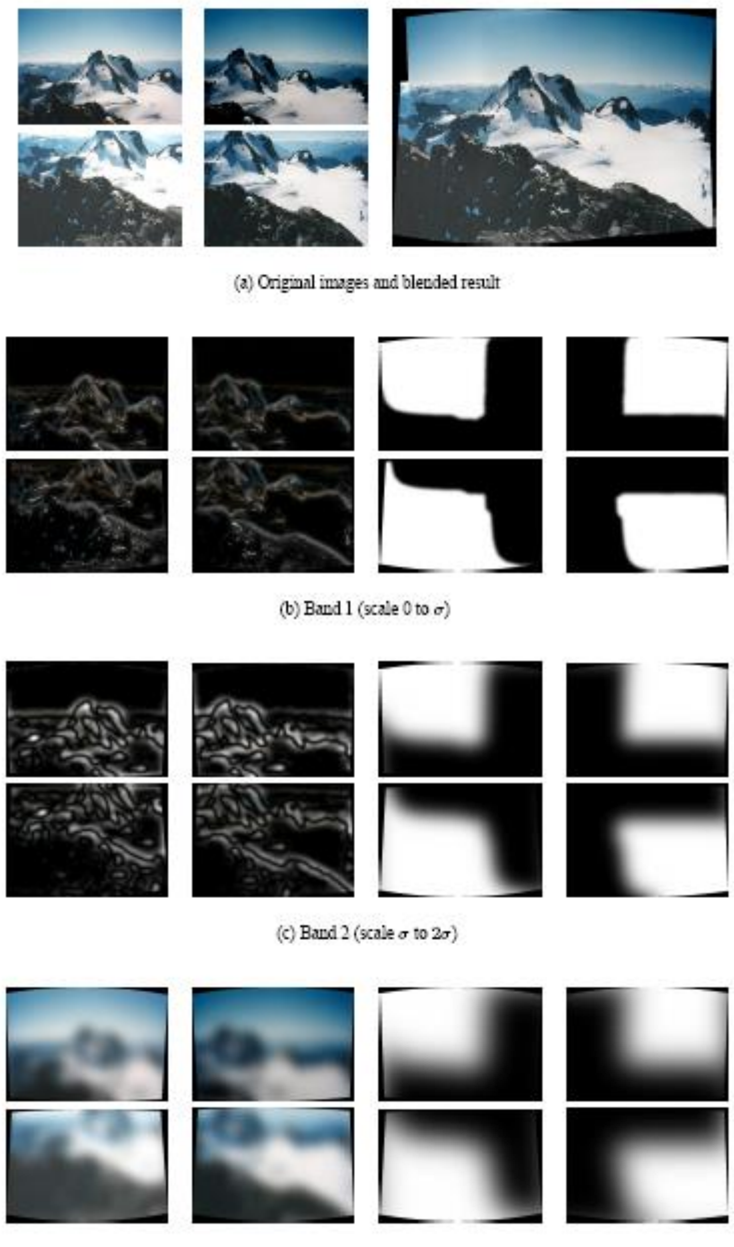
Multiband blending

Laplacian pyramids

1. Compute Laplacian pyramid of images and mask.

2. Create blended image at each level of pyramid.

3. Reconstruct complete image.



28

Q2: Multiple Views and Motion (20%)

- Image Formation, Camera and Camera Calibration
- Stereo Vision
- Optical Flow

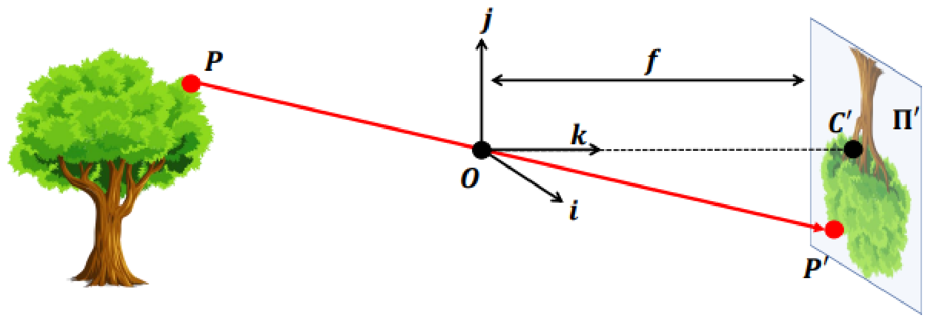
Image Formation, Camera and Camera Calibration

Outline

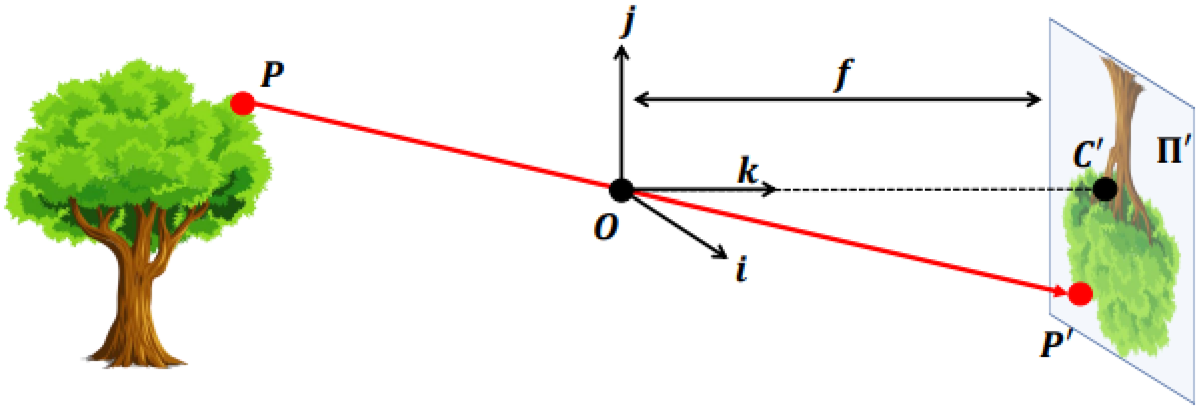
- Image Formation, Camera and Camera Calibration
- Stereo Vision

Camera Modeling: A formal construction of the pinhole camera model (perspective projection)

- Essential Components:
  - The film is commonly called the image or retinal plane:
    - The 2D plane where the projection of the 3D scene is captured, forming the image.
  - The aperture is referred to as the pinhole O or center of the camera.
    - The point through which all light rays from the 3D scene pass.
  - The focal length f.
    - The distance between the image plane and the pinhole O.
  - Camera Intrinsic
    - Parameters such as focal length, principal point (the intersection of the optical axis with the image plane), and skew (if the image axes are not perpendicular).



Camera Modeling

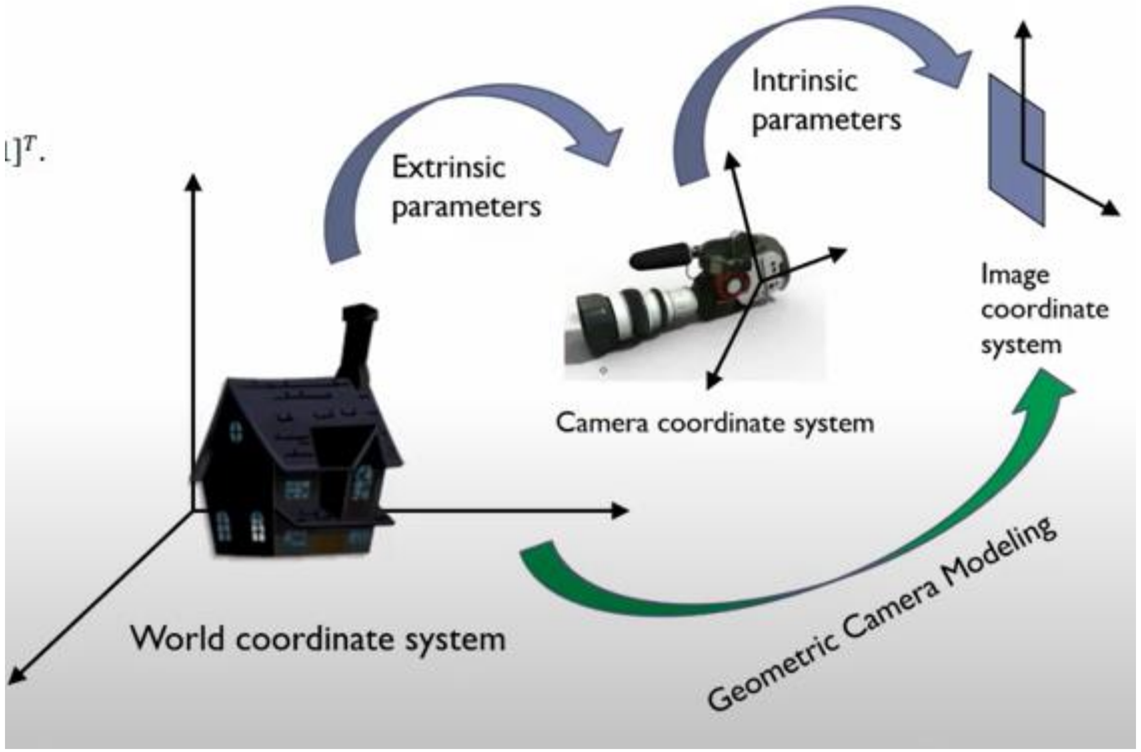


- Triangle $P'C'O$ is similar to the triangle formed by P, O and $(0, 0, Z)$. Therefore, using the law of similar triangles we find that:

$$\frac{f}{z} = \frac{P'}{P}$$

$$P' = \begin{bmatrix} x' & y' \end{bmatrix}^T = \begin{bmatrix} f\frac{x}{z} & f\frac{y}{z} \end{bmatrix}^T \qquad (1)$$

Geometric Camera Modeling



The Camera Matrix Model and Homogeneous Coordinates

Using homogeneous coordinates, we can formulate

$$P'_h = \begin{bmatrix} \alpha x + c_x z \\ \beta y + c_y z \\ z \end{bmatrix} = \begin{bmatrix} \alpha & 0 & c_x & 0 \\ 0 & \beta & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha & 0 & c_x & 0 \\ 0 & \beta & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} P_h \qquad (5)$$

Drop the h index, so any point $P\ or\ P'$ can be assumed to be in homogeneous coordinates

$$P' = \begin{bmatrix} x' \\ y' \\ z \end{bmatrix} = \begin{bmatrix} \alpha & 0 & c_x & 0 \\ 0 & \beta & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha & 0 & c_x & 0 \\ 0 & \beta & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} P = MP \qquad (6)$$

We can decompose this transformation a bit further into

$$P' = MP = \begin{bmatrix} \alpha & 0 & c_x \\ 0 & \beta & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} I & 0 \end{bmatrix} P = K \begin{bmatrix} I & 0 \end{bmatrix} P \qquad (7)$$

The matrix K is often referred to as the camera matrix.

The Camera Matrix Model and Homogeneous Coordinates

- The Complete Camera Matrix Model:
  - Most methods that we introduce in this class ignore distortion effects, therefore our class camera matrix K has 5 degrees of freedom:
    - 2 for focal length, 2 for offset, and 1 for skewness.

$$K = \begin{bmatrix} \alpha & -\alpha \cot\theta & c_x \\ 0 & \frac{\beta}{\sin\theta} & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{8}$$

These parameters are collectively known as the intrinsic parameters.

Cont. …

- Extrinsic Parameters:
  - given a point in a world reference system p_W,

$$P = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} P_w \qquad (9)$$

These parameters R and T are known as the extrinsic parameters because they are external to and do not depend on the camera.

Substituting Eq. 9 in equation (7) and simplifying gives

$$P' = MP = \begin{bmatrix} \alpha & 0 & c_x \\ 0 & \beta & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} I & 0 \end{bmatrix} P = K \begin{bmatrix} I & 0 \end{bmatrix} P \qquad (7)$$

$$P' = K \begin{bmatrix} R & T \end{bmatrix} P_w = MP_w \qquad (10)$$

The Camera Matrix Model and Homogeneous Coordinates

$$P' = K \begin{bmatrix} R & T \end{bmatrix} P_w = M P_w \tag{10}$$

- This completes the mapping from a 3D point P in an arbitrary world reference system to the image plane.
- To reiterate, we see that the full projection matrix M consists of the two types of parameters introduced above:
  - Intrinsic and extrinsic parameters.
  - All parameters contained in the camera matrix K are the intrinsic parameters, which change as the type of camera changes.
  - The extrinsic parameters include the rotation and translation, which do not depend on the camera's build.
  - Overall, we find that the 3 × 4 projection matrix M has 11 degrees of freedom:
    - 5 from the intrinsic camera matrix, 3 from extrinsic rotation, and 3 from extrinsic translation.

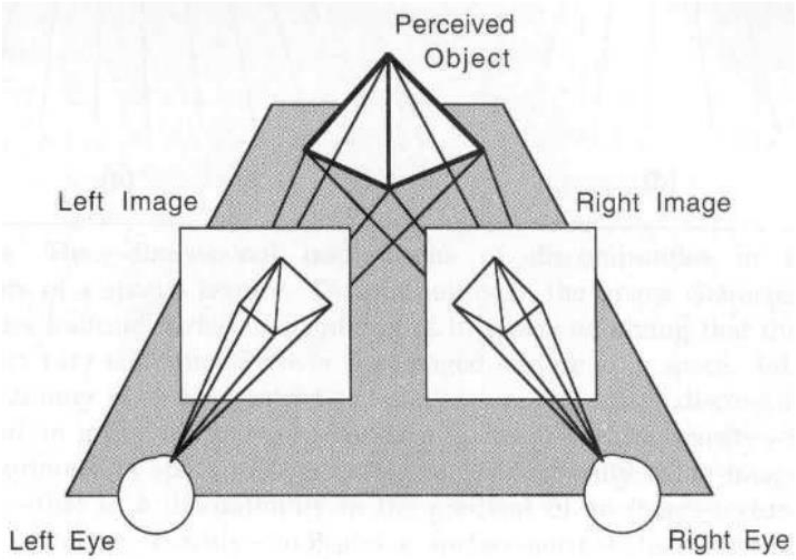
Where does Camera Model Leads?

- It also leads into camera calibration, which is usually done in factory settings to solve for the camera parameters before performing an industrial task..]
- We need it to understand stereo.
- And 3D reconstruction.

Stereo Vision

Simple (Calibrated) Stereo vision

- The recovery of the 3D structure of a scene using two or more images of the 3D scene, each acquired from a different viewpoint in space.
- The images can be obtained using multiple cameras or one moving camera.
- The term binocular vision is used when two cameras are employed.



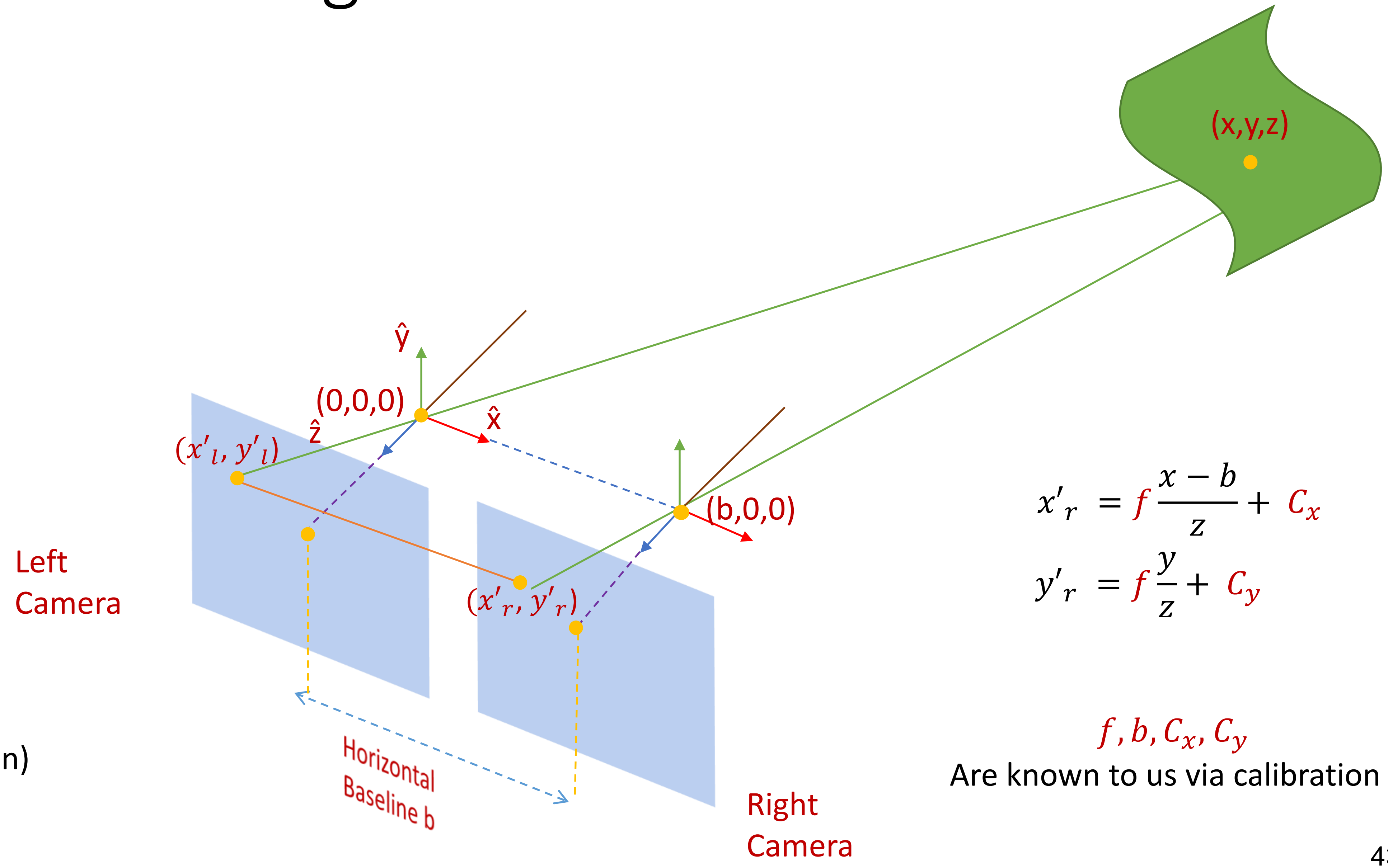
The two problems of stereo

- The correspondence problem.
  - Finding pairs of matched points such that each point in the pair is the projection of the same 3D point.
  - Triangulation depends crucially on the solution of the correspondence problem.
- The reconstruction problem.
  - Given the corresponding points, we can compute the disparity map.
  - The disparity map can be converted to a 3D map of the scene (i.e., recover the 3D structure) if the stereo geometry is known.

Triangulation using Two Cameras

$$x'_l = f\frac{x}{z} + C_x$$

$$y'_l = f\frac{y}{z} + C_y$$



$$x'_r = f\frac{x-b}{z} + C_x$$

$$y'_r = f\frac{y}{z} + C_y$$

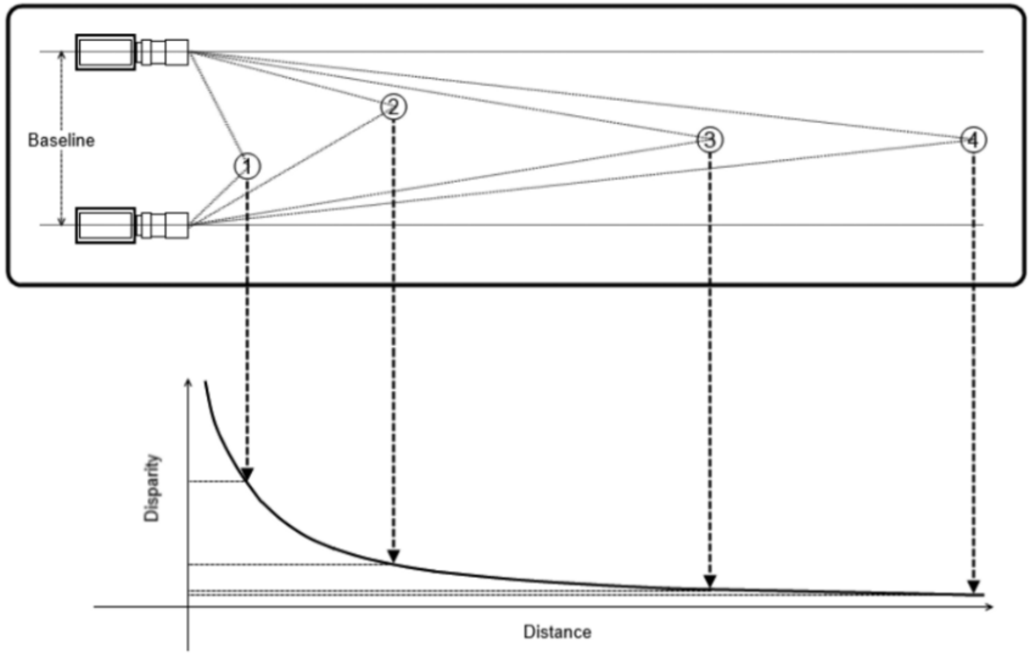
Stereo System
(Binocular Vision)

f, b, C_x, C_y
Are known to us via calibration

Simple Stereo: Depth and Disparity

$$(x'_l, y'_l) = \left(f\frac{x}{z} + C_x, y'_l \; = f\frac{y}{z} + C_y\right) \qquad (x'_r, y'_r) = \left(f\frac{x-b}{z} + C_x, f\frac{y}{z} + C_y\right)$$

$$x = \frac{b(x'_l - C_x)}{(x'_l - x'_r)} \qquad y = \frac{b(y' - C_y)}{(x'_l - x'_r)} \qquad \boxed{z = \frac{fb}{(x'_l - x'_r)}}$$



Where $(x'_l - x'_r)$ is called Disparity.

Depth z is inversely proportional to Disparity.
Disparity/Parallax is proportional to Baseline.

How we drive X,Y,Z?

$x'_l = f\frac{x}{z} + C_x$	$x'_r = f\frac{x-b}{z} + C_x$
$x = \frac{z}{f}(x'_l - C_x)$	$x = \frac{z}{f}(x'_r - C_x) + b$
$= \frac{z}{f}(x'_l - \cancel{C_x})$	$= \frac{z}{f}(x'_r - \cancel{C_x}) + b$

$$z = \frac{fb}{(x'_l - x'_r)}$$

Derivation of X,Y,

$x = \frac{z}{f}(x'_r - C_x) + b$	**Replce z with** $z = \dfrac{fb}{(x'_l - x'_r)}$
$x = \dfrac{b(x'_l - C_x)}{(x'_l - x'_r)}$	
$y = \dfrac{z}{f}(y' - C_y)$	**Replce z with** $z = \dfrac{fb}{(x'_l - x'_r)}$
$y = \dfrac{b(y' - C_y)}{(x'_l - x'_r)}$	

Basic Stereo Matching Algorithm/Compute depth map

1. **Rectify** the stereo images to align epipolar lines. (not required for basic stereo)
2. For each pixel in the left image:
   - **Find the corresponding pixel** in the right image along the scanline.
   - **Compute disparity** $d = x - x'$.
3. **Triangulate** to compute depth $z = \frac{f.B}{d}$
4. **Create a depth map** by storing depth values for all pixels.

Similarity Metrics for Template Matching:

- Similarity Metrics for Template Matching:
  - Find pixel$(k, l) \in L$ with Minimum Sum of Absolute Differences:

$$SAD(k, l) = \sum |E_l(i, j) - \ E_r\ (i + k, j + 1)|$$

  - Find pixel$(k, l) \in L$ with Minimum Sum of Squared Differences:

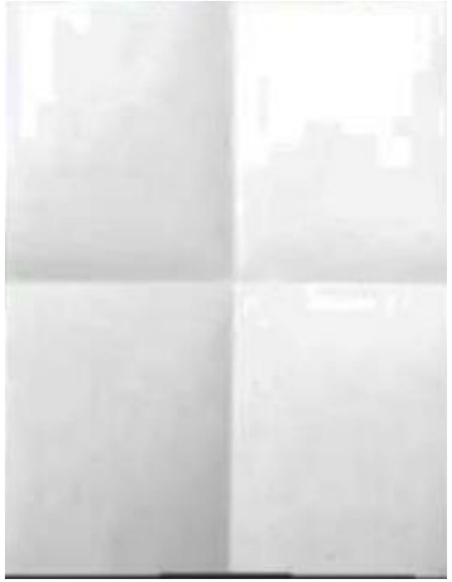
$$SSD(k, l) = \sum\nolimits_{(i,j)\in T} |E_l(i, j) - \ E_r\ (i + k, j + 1)|^2$$

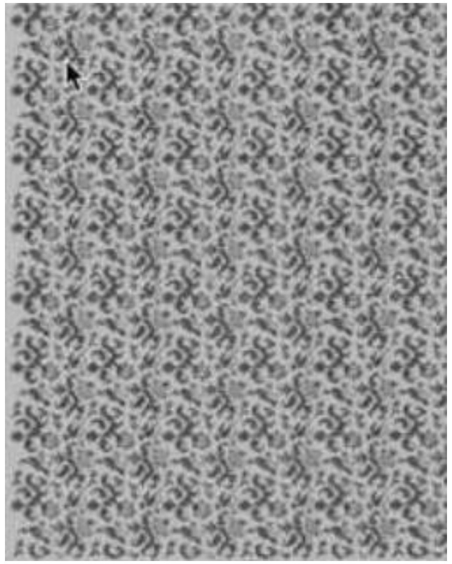
  - Find pixel$(k, l) \in L$ with Maximum of Normalized Cross-Correlation

$$\mathrm{NCC}(k, l) = \frac{\sum |E_l(i,j) - E_r\ (i+k, j+1)|}{\sqrt{\sum_{(i,j)\in T} E_l(i,j)^2\ \ \sum_{(i,j)\in T} E_r\ (i+k, j+1)^2}}$$

Issue with Stereo Matching/ What will cause errors?

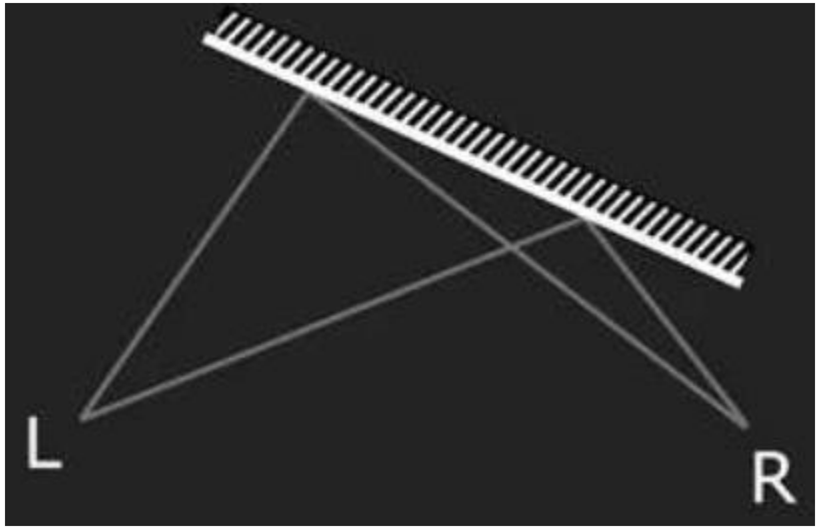
- Chose the image that have texture and non repetitive texture





Issue with Stereo Matching/ What will cause errors?

- Foreshortening effect makes matching challenging



Issue with Stereo Matching/ What will cause errors?

- Violations of brightness constancy (specular reflections)

Issue with Stereo Matching/ What will cause errors?

- Camera calibration errors
- Poor image resolution
- Occlusions
- Large motions
- Low-contrast image regions

Optical Flow

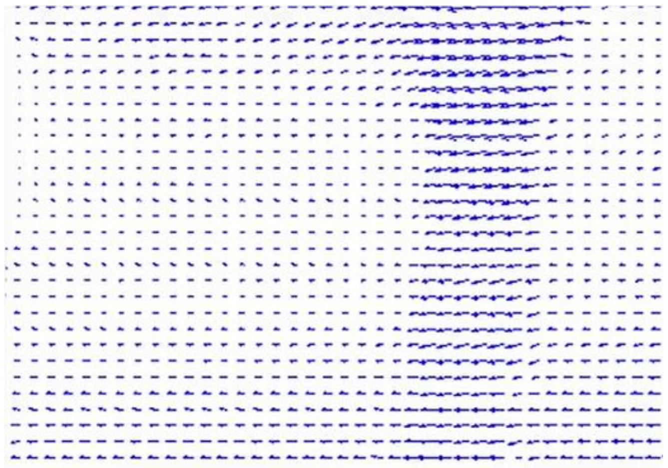
When is Optical Flow $\neq$ Motion Field?

- Optical flow is equal to motion field?
- Not all the time.

Optical Flow

- Motion of brightness patterns in the image

Image Sequence (2 frames)



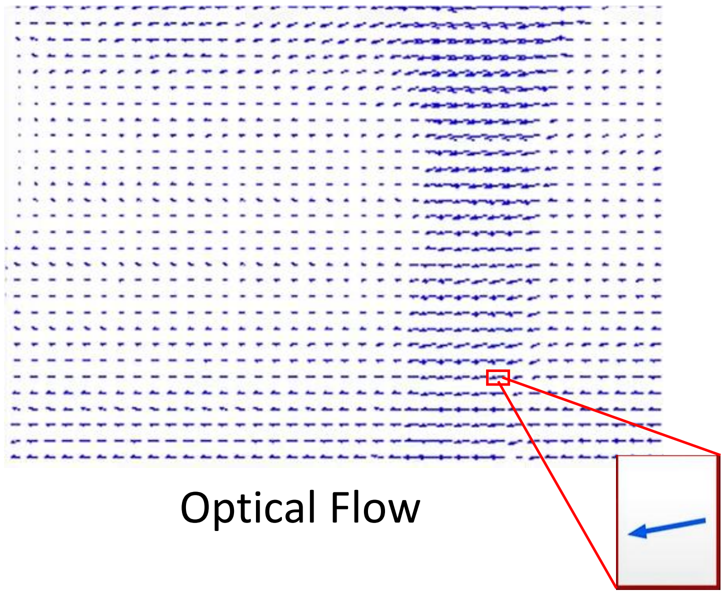
Optical Flow

- OF algorithm, Intended to develop algorithm which
  - Take the pattern, the brightness pattern in one image.
  - Observe where the brightness pattern ends up in the second image.
  - Generate the flow shown in the image

Optical Flow

- Motion of brightness patterns in the image

Image Sequence (2 frames)



Optical Flow

Velocity of brightness pattern

- The motion of brightness patterns is the optical flow.
- Each pixel you have a vector which tells you what the optical flow at that point.
  - The length of the vector tells you:
    - how fast it's moving and
  - Its direction tells you
    - Which direction it's moving in on the image plane.
- Ideally, optical flow is equal to motion field.

Optical Flow Constraint Equation

- Estimating optical flow?
  - It turns out it's a hard problem.
  - It's an under constraint problem.
- Drive a constraint equation is known as an optical flow constraint equation,
- Then develop an algorithm for estimating the optical flow at each point that uses the derived constrain equation.

Optical Flow Constraint Equation

$I(x + \delta x, y + \delta y, t + \delta t) = I(x, y, t)$ …(1) Brightens Assum.

$I(x + \delta x, y + \delta y, t + \delta t) = I(x, y, t) + I_x \delta x + I_y \delta y + I_t \delta t$…(2) Displacement Assum.

Subtract (1) from (2): $I_x \delta x + I_y \delta y + I_t \delta t = 0$

Divide by δt *and take limit as* $\delta t \rightarrow 0$: $I_x \frac{\delta x}{\delta t} + I_y \frac{\delta y}{\delta t} + I_t = 0$…

- replace $\frac{\delta x}{\delta t}$ with u component and
- $\frac{\delta y}{\delta t}$ with v component

Optical Flow Constrained Equation: $I_x u + I_y v + I_t = 0$ (u, v): *Optical Flow*

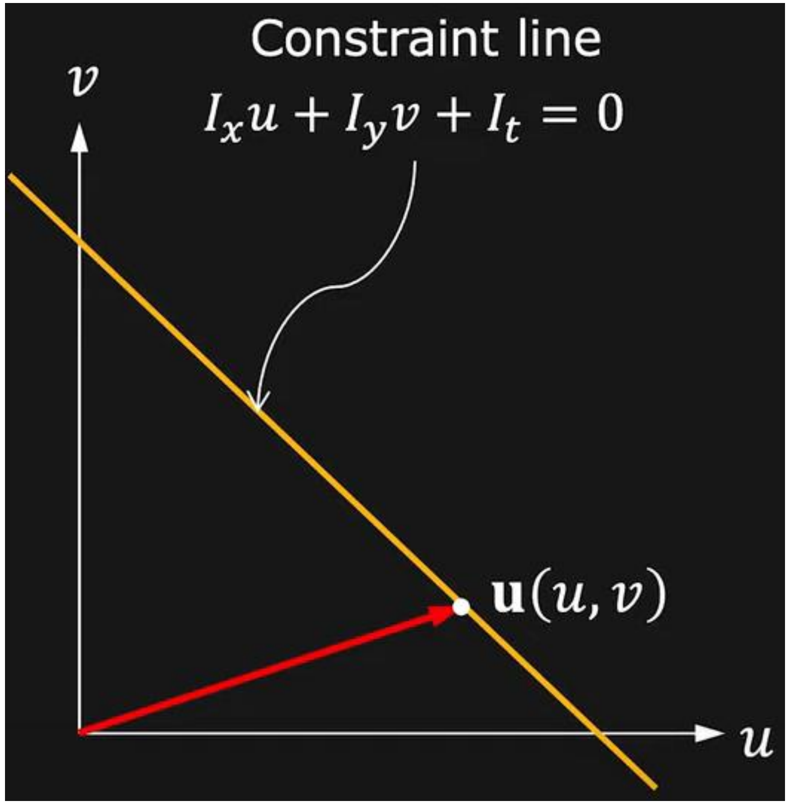
(I_x, I_y, I_t) Can be easily computed from two frames taken in **quick succession using finite differences.**

Geometric Interpretation of OF Constraint Equation

- For any point (x, y) in the image, its optical flow (u, v) lies on the line

$$I_x u + I_y v + I_t \ = 0$$

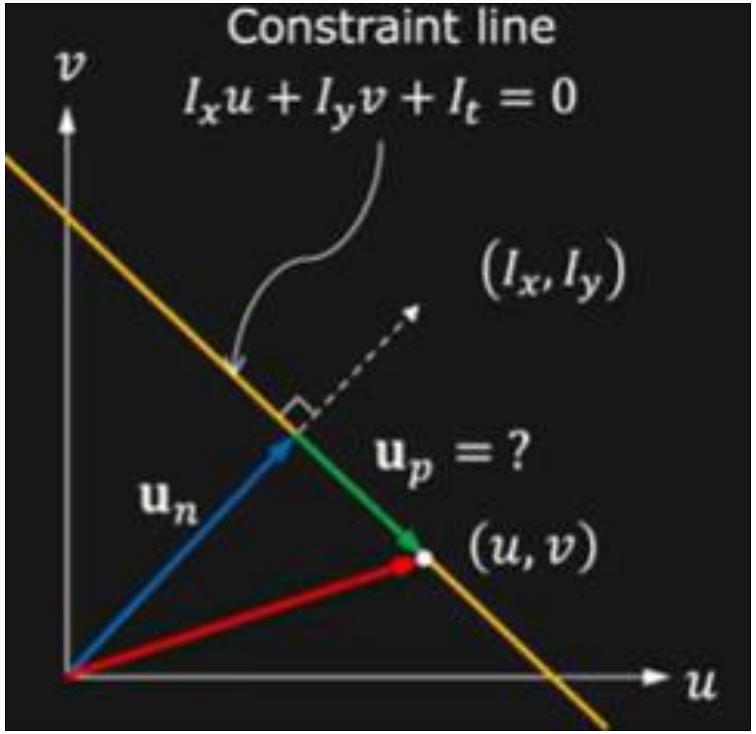
- We know that u, v lies on constraint line, but we don't know where exactly it lies.
  - This is what makes the optical flow estimation problem and under constrained problem.
  - So what we can do?



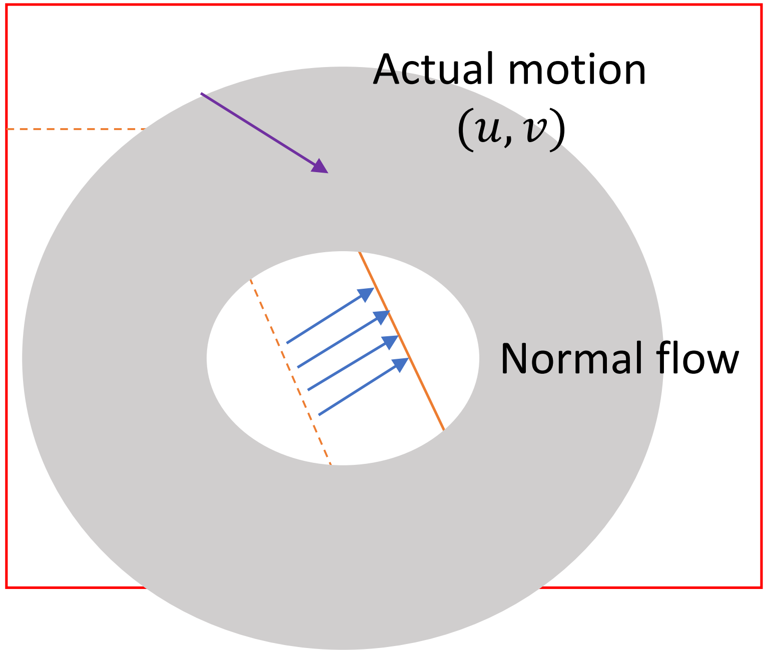
Parallel Flow

- We cannot determine u_p, the optical flow component parallel to the constraint line.

- The aforementioned issue is called the aperture problem – the motion of an edge as seen through an aperture is essentially ambiguous.



Aperture problem



- We are not able to measure the actual flow.
- We can only able to determine the normal flow.

[Aperature problem Demo](https://elvers.us/perception/aperture/)
https://elvers.us/perception/aperture/

Given two consecutive frames compute the motion vector(OF) for each pixel in the image using LK method

1) Compute Gradients: Calculate spatial gradients I_x *and* I_y(*x and y derivatives*) and temporal gradient I_t (frame difference).

2) Optical Flow Equation: For each pixel, set up the equation: $I_x u + I_y v = -I_t$ where u and v are the motion vectors in the I_x *and* I_y directions, respectively.

3) Local Window System: For each pixel, gather the equations from its local window (e.g., 5×5) and form the system:

$$\mathrm{A}\begin{bmatrix} u \\ v \end{bmatrix} = b \,,$$

where A contains the spatial gradients I_x *and* I_y for all pixels in the window, and b contains the negative temporal gradients $-I_t$.

4) Solve for Motion Vectors: Use least squares to solve for u *and* v:

$$\begin{bmatrix} u \\ v \end{bmatrix} = (A^T A)^{-1} A^T b$$

5) Iterate: Repeat for all pixels to compute the flow vectors.

Dense and Sparse Optical Flow

- Dense optical flow
  - Compute estimate for each pixel.
  - Higher accuracy at the cost of slow/expense computation.
- Sparse optical flow
  - Compute estimate for some interesting feature points (given by corners or SIFT).
  - Much less computation cost.

Sparse Dense

63

Q3: Visual Recognition (20%)

- Bag of Visual Words
- Introduction to Deep Learning and CNNs
- CNN Architectures
- GAN

Bag of Visual Words

What is Bag of Visual Word for?

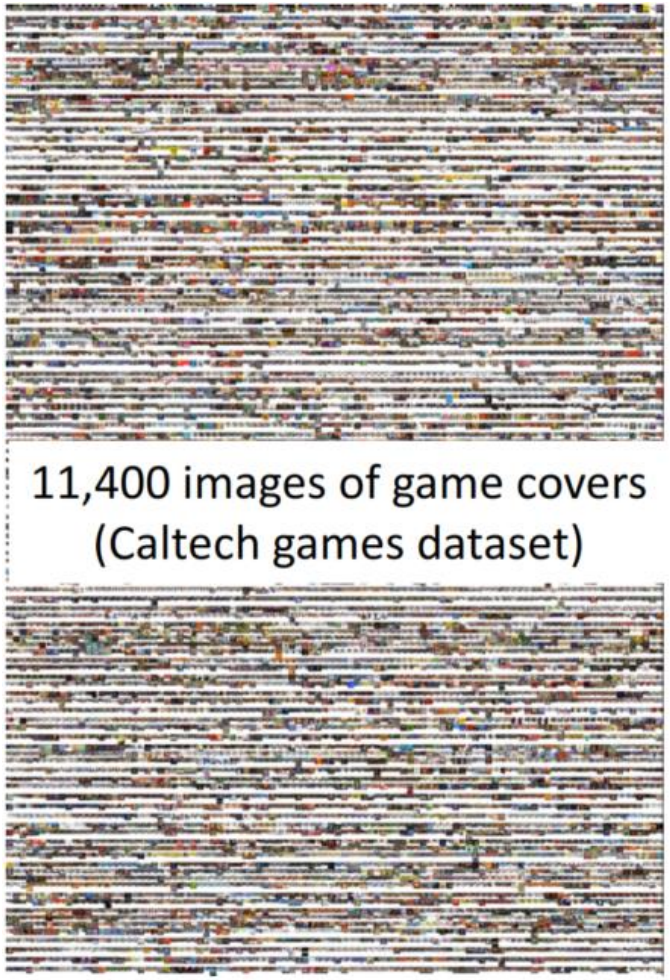
- Finding images in a database, which are similar to a given query image.
  - E.g. Google image search
- Computing image similarities
- Compact representation of images

Task Description

- Task: Find similar looking images
- Input:
  - Database of images
  - Dictionary
  - Query image(s)

- Output:
  - The N most similar database images to the query image

Large-scale image matching



- Bag-of-words models have been useful in matching an image to a large database of object instances.

How do I find this image in the database?

[Image courtesy: Fei-Fei Li]

Large-scale image search

- Build the database:
  - Extract features from the database images
  - Learn a vocabulary using k-means (typical k: 100,000)
  - Compute weights for each word
  - Create an inverted file mapping words →images

[Image courtesy: Fei-Fei Li]

Similarity Queries

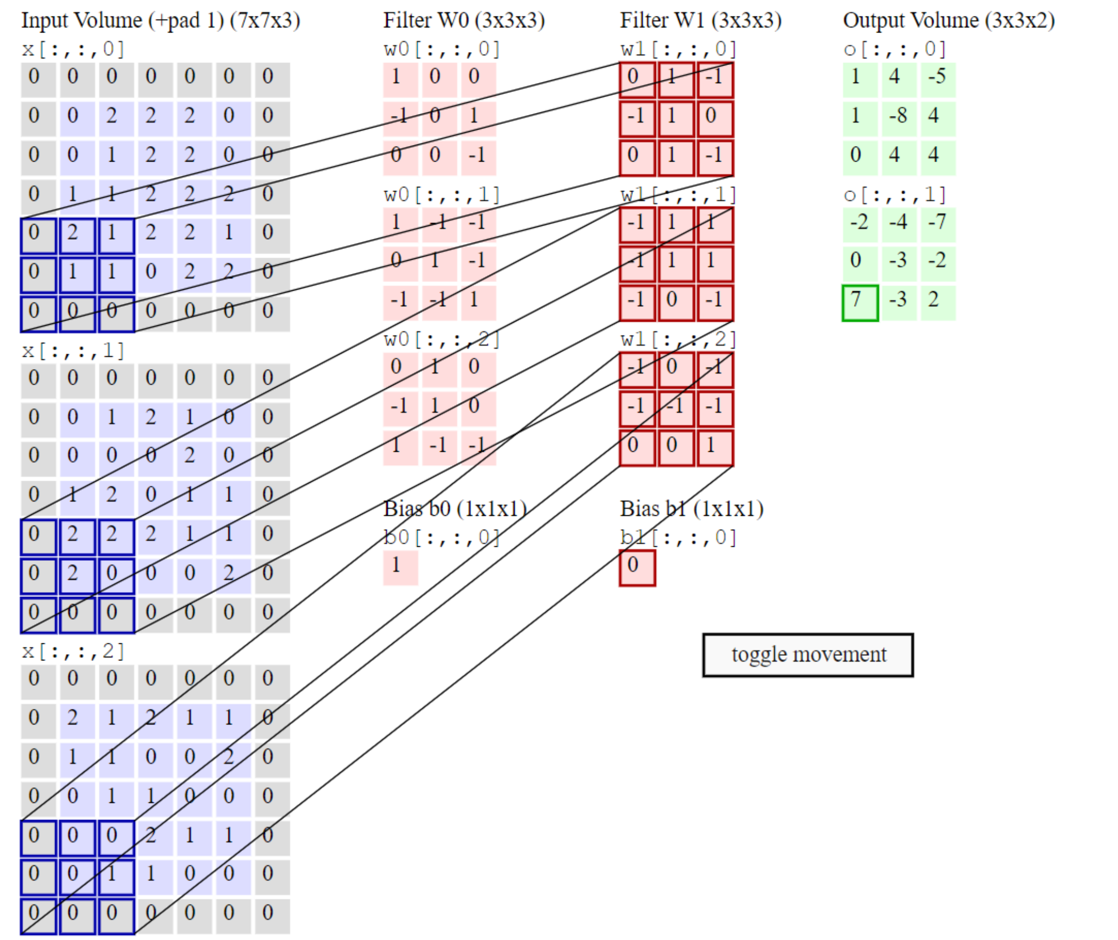
- Database stores TF-IDF weighted histograms for all database images
- Find similar images by
  - Extract features from query image
  - Assign features to visual words
  - Build TF-IDF histogram for query image
  - Return N most similar histograms from database under cosine distance

Introduction to Deep Learning and CNNs

Outline

- Convolutional Neural Networks (CNNs)
  - Activation functions
  - Fully-Connected layer to at a final stage: making a decision
  - Convolution layers
  - Pooling layers
  - Normalization

A closer look at spatial dimensions



https://cs231n.github.io/convolutional-networks/

Convolution Summary

Input: C_{in} x H x W

Hyperparameters:

- **Kernel size**: K_H x K_W
- **Number filters**: C_{out}
- **Padding**: P
- **Stride**: S

Weight matrix: C_{out} x C_{in} x K_H x K_W
giving C_{out} filters of size C_{in} x K_H x K_W

Bias vector: C_{out}

Output size: C_{out} x H' x W' where:

$$H' = \frac{(H - K + 2p)}{S} + 1$$

$$W' = \frac{(W - K + 2p)}{S} + 1$$

Common settings:

$K_H = K_W$ (Small square filters)

P = (K – 1) / 2 ("Same" padding)

C_{in}, C_{out} = 32, 64, 128, 256 (powers of 2)

K = 3, P = 1, S = 1 (3x3 conv)

K = 5, P = 2, S = 1 (5x5 conv)

K = 1, P = 0, S = 1 (1x1 conv)

K = 3, P = 1, S = 2 (Downsample by 2)

Pooling Summary

Input: C x H x W

Hyperparameters:

- Kernel size: K
- Stride: S
- Pooling function (max, avg)

Output: C x H’ x W’ where

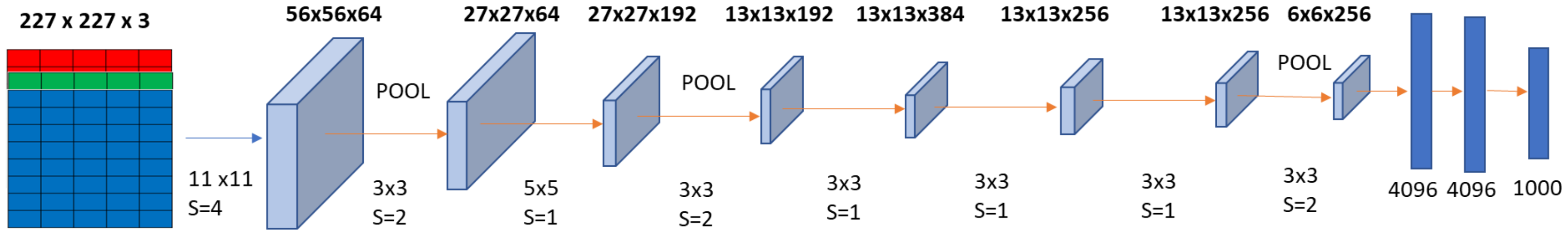
$$H' = \frac{(H - K)}{S} + 1$$

$$W' = \frac{(W - K)}{S} + 1$$

Learnable parameters: None!

Common settings:
max, K = 2, S = 2
max, K = 3, S = 2 (AlexNet)

AlexNet



	Input size		Layer				Output size				
Layer	C	H / W	filters	kernel	stride	pad	C	H / W	memory (KB)	params (k)	flop (M)
conv1	3	227	64	11	4	2	64	56	784	23	73
pool1	64	56		3	2	0	64	27	182	0	0
conv2	64	27	192	5	1	2	192	27	547	307	224
pool2	192	27		3	2	0	192	13	127	0	0
conv3	192	13	384	3	1	1	384	13	254	664	112
conv4	384	13	256	3	1	1	256	13	169	885	145
conv5	256	13	256	3	1	1	256	13	169	590	100
pool5	256	13		3	2	0	256	6	36	0	0
flatten	256	6					9216		36	0	0
fc6	9216		4096				4096		16	37,749	38

FC params = $C_{in} * C_{out} + C_{out}$
= 9216 * 4096 + 4096
= 37,725,832

FC flops = $C_{in} * C_{out}$
= 9216 * 4096
= 37,748,736

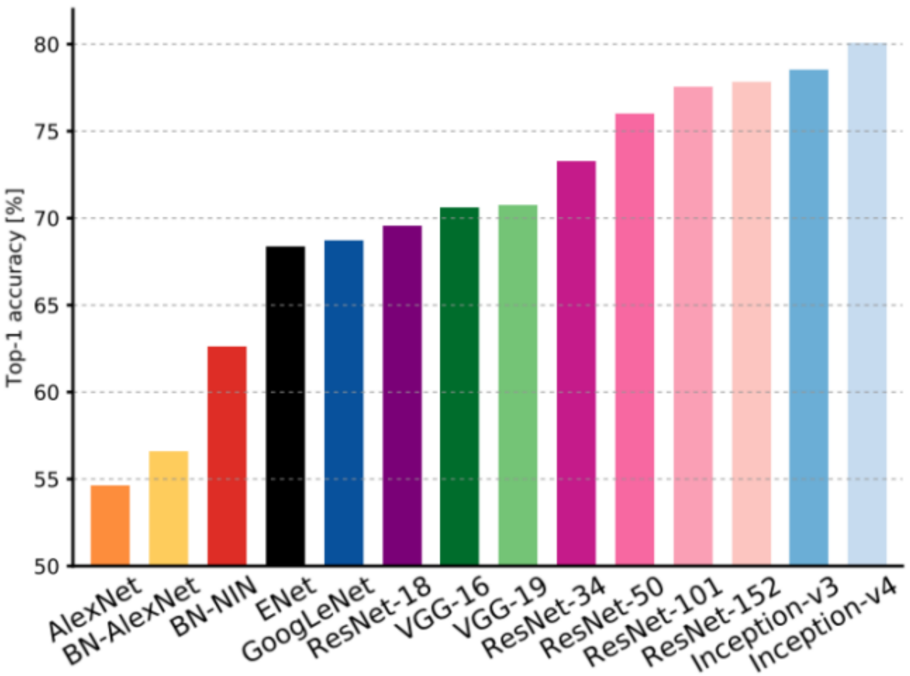
	Input size		Layer				Output size				
Layer	C	H / W	filters	kernel	stride	pad	C	H / W	memory (KB)	params (k)	flop (M)
conv1	3	227	64	11	4	2	64	56	784	23	73
pool1	64	56		3	2	0	64	27	182	0	0
conv2	64	27	192	5	1	2	192	27	547	307	224
pool2	192	27		3	2	0	192	13	127	0	0
conv3	192	13	384	3	1	1	384	13	254	664	112
conv4	384	13	256	3	1	1	256	13	169	885	145
conv5	256	13	256	3	1	1	256	13	169	590	100
pool5	256	13		3	2	0	256	6	36	0	0
flatten	256	6					9216		36	0	0
fc6	9216		4096				4096		16	37,749	38
fc7	4096		4096				4096		16	16,777	17
fc8	4096		1000				1000		4	4,096	4

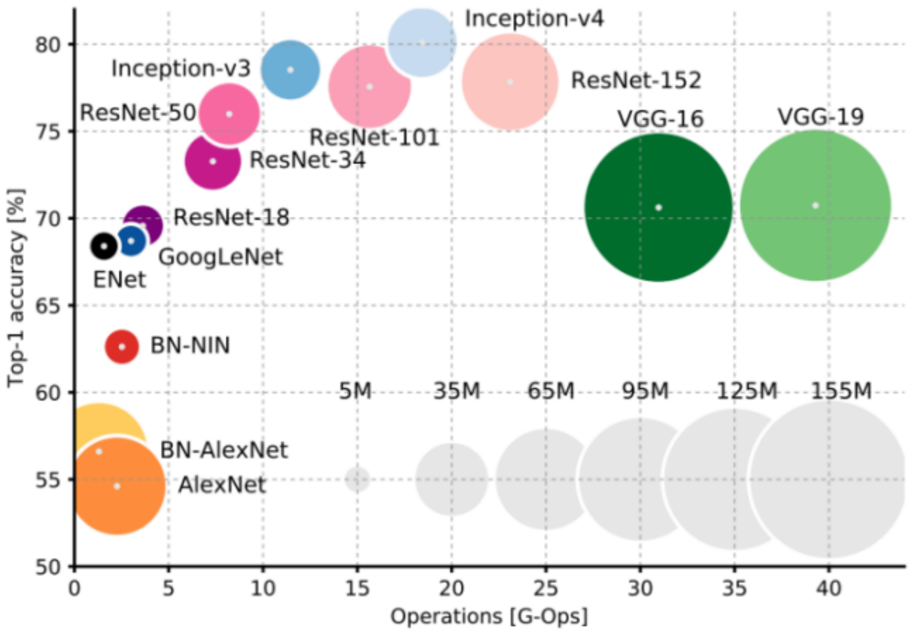
CNN Architectures

CNN Architectures Summary

- Early work (AlexNet -> ZFNet -> VGG) shows that **bigger networks work better**
- **GoogLeNet** one of the first to focus on **efficiency** (aggressive stem, 1x1 bottleneck convolutions, global avg pool instead of FC layers)
- **ResNet** showed us how to train **extremely deep networks, residual block.**
- After ResNet: **Efficient networks** became central: how can we improve the accuracy without increasing the complexity?
- Lots of **tiny networks** aimed at mobile devices: MobileNet, ShuffleNet, etc

Comparing Complexity



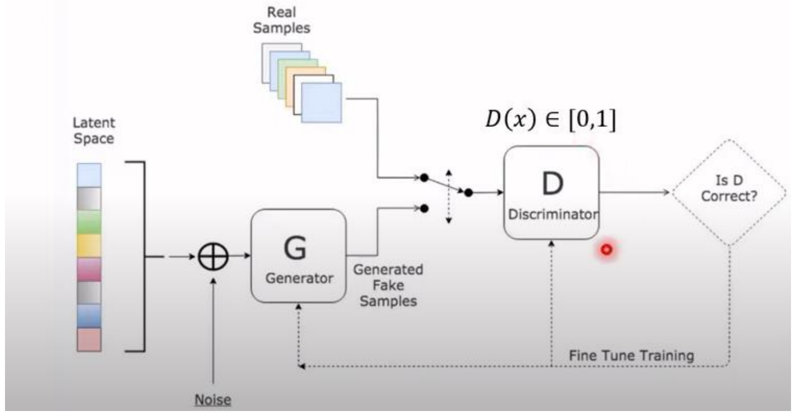


Canziani et al, “An analysis of deep neural network models for practical applications”, 2017

Generative Adversarial Networks

Training GANs: Two –player game

- **Generator network:** try to fool the discriminator by generating real-looking images
- **Discriminator network:** try to distinguish between real and fake images
- Train Jointly in **minimax game:**
  - Minimax objective function:



Discriminator outputs likelihood in (0,1) of real image

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log \underbrace{D_{\theta_d}(x)}_{\text{Discriminator output for real data x}} + \mathbb{E}_{z \sim p(z)} \log(1 - \underbrace{D_{\theta_d}(G_{\theta_g}(z))}_{\text{Discriminator output for generated fake data G(z)}}) \right]$$

- Where,
  - D(x) is the discriminator's estimate of the probability that real data instance x is real.
  - Ex is the expected value over all real data instances.
  - G(z) is the generator's output when given noise z.
  - D(G(z)) is the discriminator's estimate of the probability that a fake instance is real.
  - Ez is the expected value over all random inputs to the generator (in effect, the expected value over all generated fake instances G(z)).

From Goodfellow et al, 2014, Generative Adversarial Networks

Good Luck!